

STATA BASE REFERENCE MANUAL

RELEASE 11



A Stata Press Publication
StataCorp LP
College Station, Texas



Copyright © 1985–2009 by StataCorp LP
All rights reserved
Version 11

Published by Stata Press, 4905 Lakeway Drive, College Station, Texas 77845

Typeset in T_EX

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN-10: 1-59718-066-1 (volumes 1–3)

ISBN-10: 1-59718-067-X (volume 1)

ISBN-10: 1-59718-068-8 (volume 2)

ISBN-10: 1-59718-069-6 (volume 3)

ISBN-13: 978-1-59718-066-5 (volumes 1–3)

ISBN-13: 978-1-59718-067-2 (volume 1)

ISBN-13: 978-1-59718-068-9 (volume 2)

ISBN-13: 978-1-59718-069-6 (volume 3)

This manual is protected by copyright. All rights are reserved. No part of this manual may be reproduced, stored in a retrieval system, or transcribed, in any form or by any means—electronic, mechanical, photocopy, recording, or otherwise—without the prior written permission of StataCorp LP unless permitted by the license granted to you by StataCorp LP to use the software and documentation. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document.

StataCorp provides this manual “as is” without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. StataCorp may make improvements and/or changes in the product(s) and the program(s) described in this manual at any time and without notice.

The software described in this manual is furnished under a license agreement or nondisclosure agreement. The software may be copied only in accordance with the terms of the agreement. It is against the law to copy the software onto DVD, CD, disk, diskette, tape, or any other medium for any purpose other than backup or archival purposes.

The automobile dataset appearing on the accompanying media is Copyright © 1979 by Consumers Union of U.S., Inc., Yonkers, NY 10703-1057 and is reproduced by permission from CONSUMER REPORTS, April 1979.

Stata and Mata are registered trademarks and NetCourse is a trademark of StataCorp LP.

Other brand and product names are registered trademarks or trademarks of their respective companies.

For copyright information about the software, type `help copyright` within Stata.

The suggested citation for this software is

StataCorp. 2009. *Stata: Release 11*. Statistical Software. College Station, TX: StataCorp LP.

Table of contents

intro	Introduction to base reference manual	1
about	Display information about your Stata	6
adoupdate	Update user-written ado-files	7
alpha	Compute interitem correlations (covariances) and Cronbach's alpha	11
ameans	Arithmetic, geometric, and harmonic means	18
anova	Analysis of variance and covariance	22
anova postestimation	Postestimation tools for anova	61
areg	Linear regression with a large dummy-variable set	78
areg postestimation	Postestimation tools for areg	84
asclogit	Alternative-specific conditional logit (McFadden's choice) model	86
asclogit postestimation	Postestimation tools for asclogit	97
asmprobit	Alternative-specific multinomial probit regression	104
asmprobit postestimation	Postestimation tools for asmprobit	128
asroprobit	Alternative-specific rank-ordered probit regression	138
asroprobit postestimation	Postestimation tools for asroprobit	151
BIC note	Calculating and interpreting BIC	159
binreg	Generalized linear models: Extensions to the binomial family	164
binreg postestimation	Postestimation tools for binreg	177
biprobit	Bivariate probit regression	181
biprobit postestimation	Postestimation tools for biprobit	189
bitest	Binomial probability test	192
bootstrap	Bootstrap sampling and estimation	197
bootstrap postestimation	Postestimation tools for bootstrap	219
boxcox	Box-Cox regression models	223
boxcox postestimation	Postestimation tools for boxcox	233
brier	Brier score decomposition	235
bsample	Sampling with replacement	241
bstat	Report bootstrap results	248
centile	Report centile and confidence interval	256
ci	Confidence intervals for means, proportions, and counts	262
clogit	Conditional (fixed-effects) logistic regression	274
clogit postestimation	Postestimation tools for clogit	290
cloglog	Complementary log-log regression	295
cloglog postestimation	Postestimation tools for cloglog	304
cnsreg	Constrained linear regression	307
cnsreg postestimation	Postestimation tools for cnsreg	313
constraint	Define and list constraints	316
copyright	Display copyright information	319
copyright lapack	LAPACK copyright notification	320
copyright scintilla	Scintilla copyright notification	321
copyright ttf2pt1	ttf2pt1 copyright notification	322
correlate	Correlations (covariances) of variables or coefficients	324
cumul	Cumulative distribution	332
cusum	Graph cumulative spectral distribution	336
db	Launch dialog	340
diagnostic plots	Distributional diagnostic plots	342
display	Substitute for a hand calculator	354

do	Execute commands from a file	355
doedit	Edit do-files and other text files	356
dotplot	Comparative scatterplots	357
dstdize	Direct and indirect standardization	364
dydx	Calculate numeric derivatives and integrals	380
<i>eform_option</i>	Displaying exponentiated coefficients	386
eivreg	Errors-in-variables regression	387
eivreg postestimation	Postestimation tools for eivreg	392
error messages	Error messages and return codes	394
estat	Postestimation statistics	395
estimates	Save and manipulate estimation results	403
estimates describe	Describe estimation results	407
estimates for	Repeat postestimation command across models	409
estimates notes	Add notes to estimation results	411
estimates replay	Redisplay estimation results	413
estimates save	Save and use estimation results	415
estimates stats	Model statistics	419
estimates store	Store and restore estimation results	421
estimates table	Compare estimation results	424
estimates title	Set title for estimation results	430
estimation options	Estimation options	431
exit	Exit Stata	434
exlogistic	Exact logistic regression	435
exlogistic postestimation	Postestimation tools for exlogistic	453
expoisson	Exact Poisson regression	457
expoisson postestimation	Postestimation tools for expoission	470
fracpoly	Fractional polynomial regression	472
fracpoly postestimation	Postestimation tools for fracpoly	487
frontier	Stochastic frontier models	491
frontier postestimation	Postestimation tools for frontier	505
fvrevar	Factor-variables operator programming command	507
fvset	Declare factor-variable settings	510
gllamm	Generalized linear and latent mixed models	516
glm	Generalized linear models	517
glm postestimation	Postestimation tools for glm	552
glogit	Logit and probit regression for grouped data	558
glogit postestimation	Postestimation tools for glogit, gprobit, blogit, and bprobit	569
gmm	Generalized method of moments estimation	571
gmm postestimation	Postestimation tools for gmm	628
grmeanby	Graph means and medians by categorical variables	632
hausman	Hausman specification test	635
heckman	Heckman selection model	644
heckman postestimation	Postestimation tools for heckman	660
heckprob	Probit model with sample selection	666
heckprob postestimation	Postestimation tools for heckprob	675
help	Display online help	680
hetprob	Heteroskedastic probit model	682
hetprob postestimation	Postestimation tools for hetprob	690
histogram	Histograms for continuous and categorical variables	693

hsearch	Search help files	703
inequality	Inequality measures	707
intreg	Interval regression	710
intreg postestimation	Postestimation tools for intreg	721
ivprobit	Probit model with continuous endogenous regressors	724
ivprobit postestimation	Postestimation tools for ivprobit	737
ivregress	Single-equation instrumental-variables regression	741
ivregress postestimation	Postestimation tools for ivregress	757
ivtobit	Tobit model with continuous endogenous regressors	774
ivtobit postestimation	Postestimation tools for ivtobit	785
jackknife	Jackknife estimation	789
jackknife postestimation	Postestimation tools for jackknife	801
kappa	Interrater agreement	802
kdensity	Univariate kernel density estimation	816
ksmirnov	Kolmogorov–Smirnov equality-of-distributions test	826
kwallis	Kruskal–Wallis equality-of-populations rank test	830
ladder	Ladder of powers	833
level	Set default confidence level	840
lincom	Linear combinations of estimators	842
linktest	Specification link test for single-equation models	849
lnskew0	Find zero-skewness log or Box–Cox transform	855
log	Echo copy of session to file	859
logistic	Logistic regression, reporting odds ratios	863
logistic postestimation	Postestimation tools for logistic	874
logit	Logistic regression, reporting coefficients	901
logit postestimation	Postestimation tools for logit	914
loneaway	Large one-way ANOVA, random effects, and reliability	919
lowess	Lowess smoothing	925
lpoly	Kernel-weighted local polynomial smoothing	931
lrtest	Likelihood-ratio test after estimation	941
lv	Letter-value displays	951
margins	Marginal means, predictive margins, and marginal effects	957
margins postestimation	Postestimation tools for margins	1008
matsize	Set the maximum number of variables in a model	1010
maximize	Details of iterative maximization	1012
mean	Estimate means	1019
mean postestimation	Postestimation tools for mean	1030
meta	Meta-analysis	1032
mfp	Multivariable fractional polynomial models	1033
mfp postestimation	Postestimation tools for mfp	1044
misstable	Tabulate missing values	1045
mkspline	Linear and restricted cubic spline construction	1053
ml	Maximum likelihood estimation	1059
mlogit	Multinomial (polytomous) logistic regression	1084
mlogit postestimation	Postestimation tools for mlogit	1098
more	The —more— message	1108
mprobit	Multinomial probit regression	1110
mprobit postestimation	Postestimation tools for mprobit	1118

mvreg	Multivariate regression	1121
mvreg postestimation	Postestimation tools for mvreg	1128
nbreg	Negative binomial regression	1130
nbreg postestimation	Postestimation tools for nbreg and gnbreg	1142
nestreg	Nested model statistics	1146
net	Install and manage user-written additions from the Internet	1152
net search	Search the Internet for installable packages	1170
netio	Control Internet connections	1174
news	Report Stata news	1177
nl	Nonlinear least-squares estimation	1178
nl postestimation	Postestimation tools for nl	1198
nlcom	Nonlinear combinations of estimators	1200
nlogit	Nested logit regression	1211
nlogit postestimation	Postestimation tools for nlogit	1234
nlstur	Estimation of nonlinear systems of equations	1239
nlstur postestimation	Postestimation tools for nlstur	1261
nptrend	Test for trend across ordered groups	1263
ologit	Ordered logistic regression	1267
ologit postestimation	Postestimation tools for ologit	1276
oneway	One-way analysis of variance	1280
oprobit	Ordered probit regression	1290
oprobit postestimation	Postestimation tools for oprobit	1296
orthog	Orthogonalize variables and compute orthogonal polynomials	1299
pcorr	Partial and semipartial correlation coefficients	1306
permute	Monte Carlo permutation tests	1309
pk	Pharmacokinetic (biopharmaceutical) data	1319
pkcollapse	Generate pharmacokinetic measurement dataset	1327
pkcross	Analyze crossover experiments	1330
pkequiv	Perform bioequivalence tests	1339
pkexamine	Calculate pharmacokinetic measures	1346
pkshape	Reshape (pharmacokinetic) Latin-square data	1352
pksumm	Summarize pharmacokinetic data	1360
poisson	Poisson regression	1365
poisson postestimation	Postestimation tools for poisson	1375
predict	Obtain predictions, residuals, etc., after estimation	1381
predictnl	Obtain nonlinear predictions, standard errors, etc., after estimation	1392
probit	Probit regression	1404
probit postestimation	Postestimation tools for probit	1416
proportion	Estimate proportions	1420
proportion postestimation	Postestimation tools for proportion	1425
prtest	One- and two-sample tests of proportions	1426
qc	Quality control charts	1431
qreg	Quantile regression	1446
qreg postestimation	Postestimation tools for qreg, iqreg, sqreg, and bsqreg	1466
query	Display system parameters	1468
ranksum	Equality tests on unmatched data	1474
ratio	Estimate ratios	1480
ratio postestimation	Postestimation tools for ratio	1489

reg3	Three-stage estimation for systems of simultaneous equations	1490
reg3 postestimation	Postestimation tools for reg3	1511
regress	Linear regression	1514
regress postestimation	Postestimation tools for regress	1539
regress postestimation time series	Postestimation tools for regress with time series	1584
#review	Review previous commands	1594
roc	Receiver operating characteristic (ROC) analysis	1595
rocfits	Fit ROC models	1616
rocfits postestimation	Postestimation tools for rocfits	1623
rologit	Rank-ordered logistic regression	1627
rologit postestimation	Postestimation tools for rologit	1643
rreg	Robust regression	1645
rreg postestimation	Postestimation tools for rreg	1652
runtest	Test for random order	1654
sampsi	Sample size and power determination	1660
saved results	Saved results	1671
scobit	Skewed logistic regression	1675
scobit postestimation	Postestimation tools for scobit	1684
sdtest	Variance-comparison tests	1687
search	Search Stata documentation	1694
serrbar	Graph standard error bar chart	1700
set	Overview of system parameters	1703
set_defaults	Reset system parameters to original Stata defaults	1714
set emptycells	Set what to do with empty cells in interactions	1716
set seed	Specify initial value of random-number seed	1717
signrank	Equality tests on matched data	1719
simulate	Monte Carlo simulations	1725
sj	Stata Journal and STB installation instructions	1732
sktest	Skewness and kurtosis test for normality	1735
slogit	Stereotype logistic regression	1740
slogit postestimation	Postestimation tools for slogit	1754
smooth	Robust nonlinear smoother	1758
spearman	Spearman's and Kendall's correlations	1766
spikeplot	Spike plots and rootograms	1775
ssc	Install and uninstall packages from SSC	1779
stem	Stem-and-leaf displays	1786
stepwise	Stepwise estimation	1790
suest	Seemingly unrelated estimation	1800
summarize	Summary statistics	1819
sunflower	Density-distribution sunflower plots	1828
sureg	Zellner's seemingly unrelated regression	1834
sureg postestimation	Postestimation tools for sureg	1842
swilk	Shapiro–Wilk and Shapiro–Francia tests for normality	1845
symmetry	Symmetry and marginal homogeneity tests	1849
table	Tables of summary statistics	1858
tabstat	Display table of summary statistics	1868
tabulate oneway	One-way tables of frequencies	1873
tabulate twoway	Two-way tables of frequencies	1881
tabulate, summarize()	One- and two-way tables of summary statistics	1897
test	Test linear hypotheses after estimation	1902

testnl	Test nonlinear hypotheses after estimation	1921
tetrachoric	Tetrachoric correlations for binary variables	1929
tobit	Tobit regression	1939
tobit postestimation	Postestimation tools for tobit	1946
total	Estimate totals	1951
total postestimation	Postestimation tools for total	1957
translate	Print and translate logs	1959
treatreg	Treatment-effects model	1969
treatreg postestimation	Postestimation tools for treatreg	1982
truncreg	Truncated regression	1985
truncreg postestimation	Postestimation tools for truncreg	1992
test	Mean-comparison tests	1995
update	Update Stata	2004
<i>vce_option</i>	Variance estimators	2010
view	View files and logs	2015
vwls	Variance-weighted least squares	2019
vwls postestimation	Postestimation tools for vwls	2025
which	Display location and version for an ado-file	2027
xi	Interaction expansion	2029
zinb	Zero-inflated negative binomial regression	2040
zinb postestimation	Postestimation tools for zinb	2046
zip	Zero-inflated Poisson regression	2048
zip postestimation	Postestimation tools for zip	2054
ztnb	Zero-truncated negative binomial regression	2056
ztnb postestimation	Postestimation tools for ztnb	2064
ztp	Zero-truncated Poisson regression	2067
ztp postestimation	Postestimation tools for ztp	2073
Author index		2075
Subject index		2087

Cross-referencing the documentation

When reading this manual, you will find references to other Stata manuals. For example,

[U] **26 Overview of Stata estimation commands**

[XT] **xtabond**

[D] **reshape**

The first example is a reference to chapter 26, *Overview of Stata estimation commands*, in the *User's Guide*; the second is a reference to the `xtabond` entry in the *Longitudinal-Data/Panel-Data Reference Manual*; and the third is a reference to the `reshape` entry in the *Data-Management Reference Manual*.

All the manuals in the Stata Documentation have a shorthand notation:

[GSM]	<i>Getting Started with Stata for Mac</i>
[GSU]	<i>Getting Started with Stata for Unix</i>
[GSW]	<i>Getting Started with Stata for Windows</i>
[U]	<i>Stata User's Guide</i>
[R]	<i>Stata Base Reference Manual</i>
[D]	<i>Stata Data-Management Reference Manual</i>
[G]	<i>Stata Graphics Reference Manual</i>
[XT]	<i>Stata Longitudinal-Data/Panel-Data Reference Manual</i>
[MI]	<i>Stata Multiple-Imputation Reference Manual</i>
[MV]	<i>Stata Multivariate Statistics Reference Manual</i>
[P]	<i>Stata Programming Reference Manual</i>
[SVY]	<i>Stata Survey Data Reference Manual</i>
[ST]	<i>Stata Survival Analysis and Epidemiological Tables Reference Manual</i>
[TS]	<i>Stata Time-Series Reference Manual</i>
[I]	<i>Stata Quick Reference and Index</i>
[M]	<i>Mata Reference Manual</i>

Detailed information about each of these manuals may be found online at

<http://www.stata-press.com/manuals/>

Title

intro — Introduction to base reference manual

Description

This entry describes the organization of the reference manuals.

Remarks

The complete list of reference manuals is as follows:

- [R] *Stata Base Reference Manual*
 Volume 1, A–H
 Volume 2, I–P
 Volume 3, Q–Z
- [D] *Stata Data-Management Reference Manual*
- [G] *Stata Graphics Reference Manual*
- [XT] *Stata Longitudinal-Data/Panel-Data Reference Manual*
- [MI] *Stata Multiple-Imputation Reference Manual*
- [MV] *Stata Multivariate Statistics Reference Manual*
- [P] *Stata Programming Reference Manual*
- [SVY] *Stata Survey Data Reference Manual*
- [ST] *Stata Survival Analysis and Epidemiological Tables Reference Manual*
- [TS] *Stata Time-Series Reference Manual*
- [I] *Stata Quick Reference and Index*

- [M] *Mata Reference Manual*

When we refer to “reference manuals”, we mean all manuals listed above.

When we refer to the reference manuals, we mean all manuals listed above except the *Mata Reference Manual*.

When we refer to the *Base Reference Manual*, we mean just the three-volume *Base Reference Manual*, known as [R].

When we refer to the specialty manuals, we mean all the manuals listed above except [R] and [I], the *Stata Quick Reference and Index*.

Detailed information about each of these manuals can be found online at

<http://www.stata-press.com/manuals/>

Arrangement of the reference manuals

Each manual contains the following sections:

- [Table of contents](#).

At the beginning of volume 1 of [R], the *Base Reference Manual*, is a table of contents for the three volumes.

- Cross-referencing the documentation.

This section lists all the manuals and explains how they are cross-referenced.

- Introduction.

This entry—usually called `intro`—provides an overview of the manual. In the specialty manuals, this introduction suggests entries that you might want to read first and provides information about new features.

Each specialty manual contains an overview of the commands described in it.

- Entries.

Entries are arranged in alphabetical order. Most entries describe Stata commands, but some entries discuss concepts, and others provide overviews.

Entries that describe estimation commands are followed by an entry discussing postestimation commands that are available for use after the estimation command. For example, the **xtlogit** entry in the [XT] manual is followed by the **xtlogit postestimation** entry.

- Index.

At the end of each manual is an index. The index for the entire three-volume *Base Reference Manual* is found at the end of the third volume.

The *Quick Reference and Index*, [I], contains a [combined index](#) for all the manuals and a [subject table of contents](#) for all the manuals and the *User's Guide*. It also contains quick-reference information on many subjects, such as the estimation commands.

To find information and commands quickly, use Stata's `search` command; see [\[R\] search](#) (see the entry `search` in the [R] manual). You can broaden your search to the Internet by using `search, all` to find commands and extensions written by Stata users.

Arrangement of each entry

Entries in the Stata reference manuals generally contain the following sections, which are explained below:

[Syntax](#)

[Menu](#)

[Description](#)

[Options](#)

[Remarks](#)

[Saved results](#)

[Methods and formulas](#)

[References](#)

[Also see](#)

Syntax

A command's syntax diagram shows how to type the command, indicates all possible options, and gives the minimal allowed abbreviations for all the items in the command. For instance, the syntax diagram for the `summarize` command is

```
summarize [ varlist ] [ if ] [ in ] [ weight ] [ , options ]
```

<i>options</i>	description
detail	display additional statistics
meanonly	suppress the display; calculate only the mean; programmer's option
format	use variable's display format
separator(#)	draw separator line after every # variables; default is <code>separator(5)</code>

Main

varlist may contain factor variables; see [U] 11.4.3 Factor variables.

varlist may contain time-series operators; see [U] 11.4.4 Time-series varlists.

by is allowed; see [D] `by`.

*aweight*s, *fweight*s, and *iweight*s are allowed. However, *iweight*s may not be used with the `detail` option; see [U] 11.1.6 `weight`.

Items in the typewriter-style font should be typed exactly as they appear in the diagram, although they may be abbreviated. Underlining indicates the shortest abbreviations where abbreviations are allowed. For instance, `summarize` may be abbreviated `su`, `sum`, `summ`, etc., or it may be spelled out completely. Items in the typewriter font that are not underlined may not be abbreviated.

Square brackets denote optional items. In the syntax diagram above, *varlist*, *if*, *in*, *weight*, and the *options* are optional.

The *options* are listed in a table immediately following the diagram, along with a brief description of each.

Items typed in *italics* represent arguments for which you are to substitute variable names, observation numbers, and the like.

The diagrams use the following symbols:

#	Indicates a literal number, e.g., 5; see [U] 12.2 Numbers.
[]	Anything enclosed in brackets is optional.
{ }	At least one of the items enclosed in braces must appear.
	The vertical bar separates alternatives.
%fmt	Any Stata format, e.g., %8.2f; see [U] 12.5 Formats: Controlling how data are displayed.
depvar	The dependent variable in an estimation command; see [U] 20 Estimation and postestimation commands.
exp	Any algebraic expression, e.g., (5+myvar)/2; see [U] 13 Functions and expressions.
filename	Any filename; see [U] 11.6 File-naming conventions.
indepvars	The independent variables in an estimation command; see [U] 20 Estimation and postestimation commands.
newvar	A variable that will be created by the current command; see [U] 11.4.2 Lists of new variables.

<i>numlist</i>	A list of numbers; see [U] 11.1.8 numlist .
<i>oldvar</i>	A previously created variable; see [U] 11.4.1 Lists of existing variables .
<i>options</i>	A list of options; see [U] 11.1.7 options .
<i>range</i>	An observation range, e.g., 5/20; see [U] 11.1.4 in range .
<i>"string"</i>	Any string of characters enclosed in double quotes; see [U] 12.4 Strings .
<i>varlist</i>	A list of variable names; see [U] 11.4 varlists . If <i>varlist</i> allows factor variables, a note to that effect will be shown below the syntax diagram; see [U] 11.4.3 Factor variables . If <i>varlist</i> allows time-series operators, a note to that effect will be shown below the syntax diagram; see [U] 11.4.4 Time-series varlists .
<i>varname</i>	A variable name; see [U] 11.3 Naming conventions .
<i>weight</i>	A [<i>wgttype=exp</i>] modifier; see [U] 11.1.6 weight and [U] 20.18 Weighted estimation .
<i>xvar</i>	The variable to be displayed on the horizontal axis.
<i>yvar</i>	The variable to be displayed on the vertical axis.

The *Syntax* section will indicate whether factor variables or time-series operators may be used with a command. `summarize` allows factor variables and time-series operators.

If a command allows prefix commands, this will be indicated immediately following the table of options. `summarize` allows by.

If a command allows weights, the types of weights allowed will be specified, with the default weight listed first. `summarize` allows `aweight`s, `fweight`s, and `iweight`s, and if the type of weight is not specified, the default is `aweight`s.

Menu

A menu indicates how the dialog box for the command may be accessed using the menu system.

Description

Following the syntax diagram is a brief description of the purpose of the command.

Options

If the command allows any options, they are explained here, and for dialog users the location of the options in the dialog is indicated. For instance, in the **logistic** entry in this manual, the *Options* section looks like this:

Model
...
SE/Robust
...
Reporting
...
Maximization
...

Remarks

The explanations under *Description* and *Options* are exceedingly brief and technical; they are designed to provide a quick summary. The remarks explain in English what the preceding technical jargon means. Examples are used to illustrate the command.

Saved results

Commands are classified as e-class, r-class, s-class, or n-class, according to whether they save calculated results in `e()`, `r()`, `s()`, or not at all. These results can then be used in subroutines by other programs (ado-files). Such saved results are documented here; see [U] [18.8 Accessing results calculated by other programs](#) and [U] [18.9 Accessing results calculated by estimation commands](#).

Methods and formulas

The techniques and formulas used in obtaining the results are described here as tersely and technically as possible. If a command is implemented as an ado-file, that is indicated here.

References

Published sources are listed that either were directly referenced in the preceding text or might be of interest.

Also see

Other manual entries relating to this entry are listed that might also interest you.

Also see

[U] [1.1 Getting Started with Stata](#)

Title

about — Display information about your Stata

Syntax

about

Menu

Help > About

Description

about displays information about your version of Stata.

Remarks

about displays information about the release number, flavor, serial number, and license for your Stata. If you are running Stata for Windows, information about memory is also displayed:

```
. about
Stata/SE 11.0 for Windows (32-bit)
Born 24 Aug 2009
Copyright (C) 1985-2009

Total physical memory:      506600 KB
Available physical memory:  83040 KB

11-user Stata network perpetual license:
  Serial number: 4011041234
  Licensed to:   Alan R. Riley
                 StataCorp
```

or

```
. about
Stata/IC 11.0 for Mac (64-bit Intel)
Born 24 Aug 2009
Copyright (C) 1985-2009

Single-user Stata perpetual license:
  Serial number: 30110512345
  Licensed to:   Chinh Nguyen
                 StataCorp
```

Also see

[R] [which](#) — Display location and version for an ado-file

[U] [5 Flavors of Stata](#)

Syntax

`adoupdate` [*pkglist*] [*, options*]

<i>options</i>	description
<code>update</code>	perform update; default is to list packages that have updates, but not to update them
<code>all</code>	include packages that might have updates; default is to list or update only packages that are known to have updates
<code>ssconly</code>	check only packages obtained from SSC; default is to check all installed packages
<code>dir(<i>dir</i>)</code>	check packages installed in <i>dir</i> ; default is to check those installed in PLUS
<code>verbose</code>	provide output to assist in debugging network problems

Description

User-written additions to Stata are called packages. These packages can add remarkable abilities to Stata. Packages are found and installed by using `ssc`, `findit`, and `net`.

User-written packages are updated by their developers, just as official Stata software is updated by StataCorp.

To determine whether your official Stata software is up to date, and to update it if it is not, you use `update`.

To determine whether your user-written additions are up to date, and to update them if they are not, you use `adoupdate`.

Options

`update` specifies that packages with updates be updated. The default is simply to list the packages that could be updated without actually performing the update.

The first time you `adoupdate`, do not specify this option. Once you see `adoupdate` work, you will be more comfortable with it. Then type

```
. adoupdate, update
```

The packages that can be updated will be listed and updated.

`all` is rarely specified. Sometimes, `adoupdate` cannot determine whether a package you previously installed has been updated. `adoupdate` can determine that the package is still available over the web but is unsure whether the package has changed. Usually, the package has not changed, but if you want to be certain that you are using the latest version, reinstall from the source.

Specifying `all` does this. Typing

```
. adoupdate, all
```

adds such packages to the displayed list as needing updating but does not update them. Typing

```
. adoupdate, update all
```

lists such packages and updates them.

`ssconly` is a popular option. Many packages are available from the Statistical Software Components (SSC) archive—often called the Boston College Archive—which is provided at <http://repec.org>. Many users find most of what they want there. See [R] `ssc` for more information on the SSC.

`ssconly` specifies that `adoupdate` check only packages obtained from that source. Specifying this option is popular because SSC always provides distribution dates, and so `adoupdate` can be certain whether an update exists.

`dir(dir)` specifies which installed packages be checked. The default is `dir(PLUS)`, and that is probably correct. If you are responsible for maintaining a large system, however, you may have previously installed packages in `dir(SITE)`, where they are shared across users. See [P] `sysdir` for an explanation of these directory codewords. You may also specify an actual directory name, such as `C:\mydir`.

`verbose` is specified when you suspect network problems. It provides more detailed output that may help you diagnose the problem.

Remarks

Do not confuse `adoupdate` with `update`. Use `adoupdate` to update user-written files. Use `update` to update the components (including ado-files) of the official Stata software. To use either command, you must be connected to the Internet.

Remarks are presented under the following headings:

Using adoupdate

Possible problem the first time you run adoupdate and the solution

Notes for developers

Using adoupdate

The first time you try `adoupdate`, type

```
. adoupdate
```

That is, do not specify the `update` option. `adoupdate` without `update` produces a report but does not update any files. The first time you run `adoupdate`, you may see messages such as

```
. adoupdate
(note: package utx was installed more than once; older copy removed)
(remaining output omitted)
```

Having the same packages installed multiple times is common; `adoupdate` cleans that up.

The second time you run `adoupdate`, pick one package to update. Suppose that the report indicates that package `st0008` has an update available. Type

```
. adoupdate st0008, update
```

You can specify one or many packages after the `adoupdate` command. You can even use wildcards such as `st*` to mean all packages that start with `st` or `st*8` to mean all packages that start with `st` and end with `8`. You can do that with or without the `update` option.

Finally, you can let adoupdate update all your user-written additions:

```
. adoupdate, update
```

Possible problem the first time you run adoupdate and the solution

The first time you run adoupdate, you might get many duplicate messages:

```
. adoupdate
(note: package ___ installed more than once; older copy removed)
(note: package ___ installed more than once; older copy removed)
(note: package ___ installed more than once; older copy removed)
...
(note: package ___ installed more than once; older copy removed)
(remaining output omitted)
```

Some users have hundreds of duplicates. You might even see the same package name repeated more than once:

```
(note: package stylus installed more than once; older copy removed)
(note: package stylus installed more than once; older copy removed)
```

That means that the package was duplicated twice.

Stata tolerates duplicates, and you did nothing wrong when you previously installed and updated packages. adoupdate, however, needs the duplicates removed, mainly so that it does not keep checking the same files.

The solution is to just let adoupdate run. adoupdate will run faster next time, when there are no (or just a few) duplicates.

Notes for developers

adoupdate reports whether an installed package is up to date by comparing its distribution date with that of the package available over the web.

If you are distributing software, include the line

```
d Distribution-Date: date
```

somewhere in your .pkg file. The capitalization of Distribution-Date does not matter, but include the hyphen and the colon as shown. Code the date in either of two formats:

all numeric:	yyyymmdd, for example, 20090701
Stata standard:	ddMONyyyy, for example, 01jul2009

Saved results

adoupdate saves the following in `r()`:

Macros

```
r(pkglist)  a space-separated list of package names that need updating (update not specified)
            or that were updated (update specified)
```

Methods and formulas

`adoupdate` is implemented as an ado-file.

Also see

[R] [ssc](#) — Install and uninstall packages from SSC

[R] [search](#) — Search Stata documentation

[R] [net](#) — Install and manage user-written additions from the Internet

[R] [update](#) — Update Stata

Title

alpha — Compute interitem correlations (covariances) and Cronbach's alpha

Syntax

alpha *varlist* [*if*] [*in*] [, *options*]

<i>options</i>	description
Options	
<u>asis</u>	take sign of each item as is
<u>casewise</u>	delete cases with missing values
<u>detail</u>	list individual interitem correlations and covariances
<u>generate</u> (<i>newvar</i>)	save the generated scale in <i>newvar</i>
<u>item</u>	display item-test and item-rest correlations
<u>label</u>	include variable labels in output table
<u>min</u> (#)	must have at least # observations for inclusion
<u>reverse</u> (<i>varlist</i>)	reverse signs of these variables
<u>std</u>	standardize items in the scale to mean 0, variance 1

by is allowed; see [\[D\]](#) **by**.

Menu

Statistics > Multivariate analysis > Cronbach's alpha

Description

alpha computes the interitem correlations or covariances for all pairs of variables in *varlist* and Cronbach's α statistic for the scale formed from them. At least two variables must be specified with alpha.

Options

Options

asis specifies that the sense (sign) of each item be taken as presented in the data. The default is to determine the sense empirically and reverse the scorings for any that enter negatively.

casewise specifies that cases with missing values be deleted listwise. The default is pairwise computation of covariances and correlations.

detail lists the individual interitem correlations and covariances.

generate(*newvar*) specifies that the scale constructed from *varlist* be stored in *newvar*. Unless **asis** is specified, the sense of items entering negatively is automatically reversed. If **std** is also specified, the scale is constructed by using standardized (mean 0, variance 1) values of the individual items. Unlike most Stata commands, **generate**() does not use casewise deletion. A score is created for every observation for which there is a response to at least one item (one variable in *varlist* is not missing). The summative score is divided by the number of items over which the sum is calculated.

`item` specifies that item-test and item-rest correlations and the effects of removing an item from the scale be displayed. `item` is valid only when more than two variables are specified in `varlist`.

`label` requests that the detailed output table be displayed in a compact format that enables the inclusion of variable labels.

`min(#)` specifies that only cases with at least `#` observations be included in the computations. `casewise` is a shorthand for `min(k)`, where `k` is the number of variables in `varlist`.

`reverse(varlist)` specifies that the signs (directions) of the variables (items) in `varlist` be reversed. Any variables specified in `reverse()` that are not also included in `alpha`'s `varlist` are ignored.

`std` specifies that the items in the scale be standardized (mean 0, variance 1) before summing.

Remarks

Cronbach's alpha (Cronbach 1951) assesses the reliability of a summative rating (Likert 1932) scale composed of the variables (called *items*) specified. The set of items is often called a *test* or *battery*. A scale is simply the sum of the individual item scores, reversing the scoring for statements that have negative correlations with the factor (e.g., attitude) being measured. Scales can be formed by using the raw item scores or standardized item scores.

The reliability α is defined as the square of the correlation between the measured scale and the underlying factor. If you think of a test as being composed of a random sample of items from a hypothetical domain of items designed to measure the same thing, α represents the expected correlation of one test with an alternative form containing the same number of items. The square root of α is the estimated correlation of a test with errorless true scores (Nunnally and Bernstein 1994, 235).

In addition to reporting α , `alpha` generates the summative scale from the items (variables) specified and automatically reverses the sense of any when necessary. Stata's decision can be overridden by specifying the `reverse(varlist)` option.

Because it concerns reliability in measuring an unobserved factor, α is related to factor analysis. The test should be designed to measure one factor, and, because the scale will be composed of an unweighted sum, the factor loadings should all contribute roughly equal information to the score. Both of these assumptions can be verified with `factor`; see [MV] `factor`. Equality of factor loadings can also be assessed by using the `item` option.

► Example 1

To illustrate `alpha`, we apply it, first without and then with the `item` option, to the automobile dataset after randomly introducing missing values:

```
. use http://www.stata-press.com/data/r11/automiss
(1978 Automobile Data)

. alpha price headroom rep78 trunk weight length turn displ, std
Test scale = mean(standardized items)
Reversed item: rep78

Average interitem correlation:      0.5251
Number of items in the scale:      8
Scale reliability coefficient:      0.8984
```

The scale derived from our somewhat arbitrarily chosen automobile items (variables) appears to be reasonable because the estimated correlation between it and the underlying factor it measures is $\sqrt{0.8984} \approx 0.9478$ and the estimated correlation between this battery of eight items and all other eight-item batteries from the same domain is 0.8984. Because the “items” are not on the same scale,

it is important that `std` was specified so that the scale and its reliability were based on the sum of standardized variables. We could obtain the scale in a new variable called `sc` with the `gen(sc)` option.

Though the scale appears reasonable, we include the `item` option to determine if all the items fit the scale:

```
. alpha price headroom rep78 trunk weight length turn displ, std item
Test scale = mean(standardized items)
```

Item	Obs	Sign	item-test correlation	item-rest correlation	average interitem correlation	alpha
price	70	+	0.5260	0.3719	0.5993	0.9128
headroom	66	+	0.6716	0.5497	0.5542	0.8969
rep78	61	-	0.4874	0.3398	0.6040	0.9143
trunk	69	+	0.7979	0.7144	0.5159	0.8818
weight	64	+	0.9404	0.9096	0.4747	0.8635
length	69	+	0.9382	0.9076	0.4725	0.8625
turn	66	+	0.8678	0.8071	0.4948	0.8727
displacement	63	+	0.8992	0.8496	0.4852	0.8684
Test scale					0.5251	0.8984

“Test” denotes the additive scale; here 0.5251 is the average interitem correlation, and 0.8984 is the alpha coefficient for a test scale based on all items.

“Obs” shows the number of nonmissing values of the items; “Sign” indicates the direction in which an item variable entered the scale; “-” denotes that the item was reversed. The remaining four columns in the table provide information on the effect of one item on the scale.

Column four gives the item-test correlations. Apart from the sign of the correlation for items that entered the scale in reversed order, these correlations are the same numbers as those computed by the commands

```
. alpha price headroom rep78 trunk weight length turn displ, std gen(sc)
. pwcorr sc price headroom rep78 trunk weight length turn displ
```

Typically, the item-test correlations should be roughly the same for all items. Item-test correlations may not be adequate to detect items that fit poorly because the poorly fitting items may distort the scale. Accordingly, it may be more useful to consider item-rest correlations (Nunnally and Bernstein 1994), i.e., the correlation between an item and the scale that is formed by all other items. The average interitem correlations (covariances if `std` is omitted) of all items, excluding one, are shown in column six. Finally, column seven gives Cronbach’s α for the test scale, which consists of all but the one item.

Here neither the `price` item nor the `rep78` item seems to fit well in the scale in all respects. The item-test and item-rest correlations of `price` and `rep78` are much lower than those of the other items. The average interitem correlation increases substantially by removing either `price` or `rep78`; apparently, they do not correlate strongly with the other items. Finally, we see that Cronbach’s α coefficient will increase from 0.8984 to 0.9128 if the `price` item is dropped, and it will increase from 0.8984 to 0.9143 if `rep78` is dropped. For well-fitting items, we would of course expect that α decreases by shortening the test.

➤ Example 2

The variable names for the automobile data are reasonably informative. This may not always be true; items in batteries commonly used to measure personality traits, attitudes, values, etc., are usually named with indexed names such as `item12a`, `item12b`, etc. The `label` option forces `alpha` to produce the same statistical information in a more compact format that leaves room to include variable (item) labels. In this compact format, `alpha` excludes the number of nonmissing values of the items, displays the statistics using fewer digits, and uses somewhat cryptic headers:

```
. alpha price headroom rep78 trunk weight length turn displ, std item label detail
Test scale = mean(standardized items)
```

Items	S	it-cor	ir-cor	ii-cor	alpha	label
price	+	0.526	0.372	0.599	0.913	Price
headroom	+	0.672	0.550	0.554	0.897	Headroom (in.)
rep78	-	0.487	0.340	0.604	0.914	Repair Record 1978
trunk	+	0.798	0.714	0.516	0.882	Trunk space (cu. ft.)
weight	+	0.940	0.910	0.475	0.863	Weight (lbs.)
length	+	0.938	0.908	0.473	0.862	Length (in.)
turn	+	0.868	0.807	0.495	0.873	Turn Circle (ft.)
displacement	+	0.899	0.850	0.485	0.868	Displacement (cu. in.)
Test scale				0.525	0.898	mean(standardized items)

Interitem correlations (reverse applied) (obs=pairwise, see below)

	price	headroom	rep78	trunk
price	1.0000			
headroom	0.1174	1.0000		
rep78	-0.0479	0.1955	1.0000	
trunk	0.2748	0.6841	0.2777	1.0000
weight	0.5093	0.5464	0.3624	0.6486
length	0.4511	0.5823	0.3162	0.7404
turn	0.3528	0.4067	0.4715	0.5900
displacement	0.5537	0.5166	0.3391	0.6471
	weight	length	turn	displacement
weight	1.0000			
length	0.9425	1.0000		
turn	0.8712	0.8589	1.0000	
displacement	0.8753	0.8422	0.7723	1.0000

Pairwise number of observations

	price	headroom	rep78	trunk
price	70			
headroom	62	66		
rep78	59	54	61	
trunk	65	61	59	69
weight	60	56	52	60
length	66	61	58	64
turn	62	58	56	62
displacement	59	58	51	58
	weight	length	turn	displacement
weight	64			
length	60	69		
turn	57	61	66	
displacement	54	58	56	63

Because the `detail` option was also specified, the interitem correlation matrix was printed, together with the number of observations used for each entry (because these varied across the matrix). Note the negative sign attached to `rep78` in the output, indicating the sense in which it entered the scale.

Better-looking output with less-cryptic headers is produced if the linesize is set to a value of at least 100:

```
. set linesize 100
. alpha price headroom rep78 trunk weight length turn displ, std item label
Test scale = mean(standardized items)
```

Item			item-test	item-rest	interitem	alpha	Label
	Obs	Sign	corr.	corr.	corr.		
price	70	+	0.5260	0.3719	0.5993	0.9128	Price
headroom	62	+	0.6716	0.5497	0.5542	0.8969	Headroom (in.)
rep78	59	-	0.4874	0.3398	0.6040	0.9143	Repair Record 1978
trunk	65	+	0.7979	0.7144	0.5159	0.8818	Trunk space (cu. ft.)
weight	60	+	0.9404	0.9096	0.4747	0.8635	Weight (lbs.)
length	66	+	0.9382	0.9076	0.4725	0.8625	Length (in.)
turn	62	+	0.8678	0.8071	0.4948	0.8727	Turn Circle (ft.)
displacement	59	+	0.8992	0.8496	0.4852	0.8684	Displacement (cu. in.)
Test scale					0.5251	0.8984	mean(standardized items)



Users of `alpha` require some standard for judging values of α . We paraphrase [Nunnally and Bernstein \(1994, 265\)](#): In the early stages of research, modest reliability of 0.70 or higher will suffice; values in excess of 0.80 often waste time and funds. In contrast, where measurements on individuals are of interest, a reliability of 0.80 may not be nearly high enough. Even with a reliability of 0.90, the standard error of measurement is almost one-third as large as the standard deviation of test scores; a reliability of 0.90 is the minimum that should be tolerated, and a reliability of 0.95 should be considered the desirable standard.

Saved results

`alpha` saves the following in `r()`:

Scalars	
<code>r(alpha)</code>	scale reliability coefficient
<code>r(k)</code>	number of items in the scale
<code>r(cov)</code>	average interitem covariance
<code>r(rho)</code>	average interitem correlation if <code>std</code> is specified
Matrices	
<code>r(Alpha)</code>	scale reliability coefficient
<code>r(ItemTestCorr)</code>	item-test correlation
<code>r(ItemRestCorr)</code>	item-rest correlation
<code>r(MeanInterItemCov)</code>	average interitem covariance
<code>r(MeanInterItemCorr)</code>	average interitem correlation if <code>std</code> is specified

If the `item` option is specified, results are saved as row matrices for the `k` subscales when one variable is removed.

Methods and formulas

alpha is implemented as an ado-file.

Let x_i , $i = 1, \dots, k$, be the variables over which α is to be calculated. Let s_i be the sign with which x_i enters the scale. If `asis` is specified, $s_i = 1$ for all i . Otherwise, principal-factor analysis is performed on x_i , and the first factor's score is predicted; see [MV] [factor](#). s_i is -1 if correlation of the x_i and the predicted score is negative and $+1$ otherwise.

Let r_{ij} be the correlation between x_i and x_j , c_{ij} be the covariance, and n_{ij} be the number of observations used in calculating the correlation or covariance. The average correlation is

$$\bar{r} = \frac{\sum_{i=2}^k \sum_{j=1}^{i-1} s_i s_j n_{ij} r_{ij}}{\sum_{i=2}^k \sum_{j=1}^{i-1} n_{ij}}$$

and the average covariance similarly is

$$\bar{c} = \frac{\sum_{i=2}^k \sum_{j=1}^{i-1} s_i s_j n_{ij} c_{ij}}{\sum_{i=2}^k \sum_{j=1}^{i-1} n_{ij}}$$

Let c_{ii} denote the variance of x_i , and define the average variance as

$$\bar{v} = \frac{\sum_{i=1}^k n_{ii} c_{ii}}{\sum_{i=1}^k n_{ii}}$$

If `std` is specified, the scale reliability α is calculated as defined by the general form of the Spearman–Brown Prophecy Formula ([Nunnally and Bernstein 1994](#), 232; [Allen and Yen 1979](#), 85–88):

$$\alpha = \frac{k\bar{r}}{1 + (k-1)\bar{r}}$$

This expression corresponds to α under the assumption that the summative rating is the sum of the standardized variables ([Nunnally and Bernstein 1994](#), 234). If `std` is not specified, α is defined ([Nunnally and Bernstein 1994](#), 232 and 234) as

$$\alpha = \frac{k\bar{c}}{\bar{v} + (k-1)\bar{c}}$$

Let x_{ij} reflect the value of item i in the j th observation. If `std` is specified, the j th value of the scale computed from the k x_{ij} items is

$$S_j = \frac{1}{k_j} \sum_{i=1}^k s_i S(x_{ij})$$

where $S()$ is the function that returns the standardized (mean 0, variance 1) value if x_{ij} is not missing and returns zero if x_{ij} is missing. k_j is the number of nonmissing values in x_{ij} , $i = 1, \dots, k$. If `std` is not specified, $S()$ is the function that returns x_{ij} or returns missing if x_{ij} is missing.

Lee Joseph Cronbach (1916–2001) was an American psychometrician and educational psychologist who worked principally on measurement theory, program evaluation, and instruction. He taught and researched at the State College of Washington, the University of Chicago, the University of Illinois, and Stanford. Cronbach's initial paper on alpha led to a theory of test reliability, called generalizability theory, a comprehensive statistical model for identifying sources of measurement error.

Acknowledgment

This improved version of `alpha` was written by Jeroen Weesie, Department of Sociology, Utrecht University, The Netherlands.

References

- Acock, A. C. 2008. *A Gentle Introduction to Stata*. 2nd ed. College Station, TX: Stata Press.
- Allen, M. J., and W. M. Yen. 1979. *Introduction to Measurement Theory*. Monterey, CA: Brooks/Cole.
- Bleda, M.-J., and A. Tobías. 2000. [sg143: Cronbach's alpha one-sided confidence interval](#). *Stata Technical Bulletin* 56: 26–27. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 187–189. College Station, TX: Stata Press.
- Cronbach, L. J. 1951. Coefficient alpha and the internal structure of tests. *Psychometrika* 16: 297–334.
- Likert, R. A. 1932. A technique for the measurement of attitudes. *Archives of Psychology* 140: 5–55.
- Nunnally, J. C., and I. H. Bernstein. 1994. *Psychometric Theory*. 3rd ed. New York: McGraw–Hill.
- Shavelson, R. J., and G. Gleser. 2002. Lee J. Cronbach (1916–2001). *American Psychologist* 57: 360–361.
- Tarlov, A. R., J. E. Ware Jr., S. Greenfield, E. C. Nelson, E. Perrin, and M. Zubkoff. 1989. The medical outcomes study. An application of methods for monitoring the results of medical care. *Journal of the American Medical Association* 262: 925–930.
- Weesie, J. 1997. [sg66: Enhancements to the alpha command](#). *Stata Technical Bulletin* 35: 32–34. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 176–179. College Station, TX: Stata Press.

Also see

[\[MV\] factor](#) — Factor analysis

Title

ameans — Arithmetic, geometric, and harmonic means

Syntax

ameans [*varlist*] [*if*] [*in*] [*weight*] [, *options*]

<i>options</i>	description
Main	
<u>a</u> dd(#)	add # to each variable in <i>varlist</i>
<u>o</u> nly	add # only to variables with nonpositive values
<u>l</u> evel(#)	set confidence level; default is level(95)
by is allowed; see [D] by .	
aweights and fweights are allowed; see [U] 11.1.6 weight .	

Menu

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Arith./geometric/harmonic means

Description

ameans computes the arithmetic, geometric, and harmonic means, with their corresponding confidence intervals, for each variable in *varlist* or for all the variables in the data if *varlist* is not specified. **gmeans** and **hmeans** are synonyms for **ameans**.

If you simply want arithmetic means and corresponding confidence intervals, see [R] [ci](#).

Options

Main

add(#) adds the value # to each variable in *varlist* before computing the means and confidence intervals. This option is useful when analyzing variables with nonpositive values.

only modifies the action of the **add(#)** option so that it adds # only to variables with at least one nonpositive value.

level(#) specifies the confidence level, as a percentage, for confidence intervals. The default is level(95) or as set by **set level**; see [U] [20.7 Specifying the width of confidence intervals](#).

Remarks

► Example 1

We have a dataset containing 8 observations on a variable named `x`. The eight values are 5, 4, −4, −5, 0, 0, *missing*, and 7.

```
. amean x
```

Variable	Type	Obs	Mean	[95% Conf. Interval]	
x	Arithmetic	7	1	−3.204405	5.204405
	Geometric	3	5.192494	2.57899	10.45448
	Harmonic	3	5.060241	3.023008	15.5179

```
. amean x, add(5)
```

Variable	Type	Obs	Mean	[95% Conf. Interval]	
x	Arithmetic	7	6	1.795595	10.2044 *
	Geometric	6	5.477226	2.1096	14.22071 *
	Harmonic	6	3.540984	.	. *

(*) 5 was added to the variables prior to calculating the results.
Missing values in confidence intervals for harmonic mean indicate
that confidence interval is undefined for corresponding variables.
Consult Reference Manual for details.

The number of observations displayed for the arithmetic mean is the number of nonmissing observations. The number of observations displayed for the geometric and harmonic means is the number of nonmissing, positive observations. Specifying the `add(5)` option produces 3 more positive observations. The confidence interval for the harmonic mean is not reported; see [Methods and formulas](#) below.



Saved results

`ameans` saves the following in `r()`:

Scalars	
<code>r(N)</code>	number of nonmissing observations; used for arithmetic mean
<code>r(N_pos)</code>	number of nonmissing positive observations; used for geometric and harmonic means
<code>r(mean)</code>	arithmetic mean
<code>r(lb)</code>	lower bound of confidence interval for arithmetic mean
<code>r(ub)</code>	upper bound of confidence interval for arithmetic mean
<code>r(Var)</code>	variance of untransformed data
<code>r(mean_g)</code>	geometric mean
<code>r(lb_g)</code>	lower bound of confidence interval for geometric mean
<code>r(ub_g)</code>	upper bound of confidence interval for geometric mean
<code>r(Var_g)</code>	variance of $\ln x_i$
<code>r(mean_h)</code>	harmonic mean
<code>r(lb_h)</code>	lower bound of confidence interval for harmonic mean
<code>r(ub_h)</code>	upper bound of confidence interval for harmonic mean
<code>r(Var_h)</code>	variance of $1/x_i$

Methods and formulas

`ameans` is implemented as an ado-file.

See [Armitage, Berry, and Matthews \(2002\)](#) or [Snedecor and Cochran \(1989\)](#). For a history of the concept of the mean, see [Plackett \(1958\)](#).

When restricted to the same set of values (i.e., to positive values), the arithmetic mean ($\bar{x}$) is greater than or equal to the geometric mean, which in turn is greater than or equal to the harmonic mean. Equality holds only if all values within a sample are equal to a positive constant.

The arithmetic mean and its confidence interval are identical to those provided by `ci`; see [\[R\] ci](#).

To compute the geometric mean, `ameans` first creates $u_j = \ln x_j$ for all positive x_j . The arithmetic mean of the u_j and its confidence interval are then computed as in `ci`. Let $\bar{u}$ be the resulting mean, and let $[L, U]$ be the corresponding confidence interval. The geometric mean is then $\exp(\bar{u})$, and its confidence interval is $[\exp(L), \exp(U)]$.

The same procedure is followed for the harmonic mean, except that then $u_j = 1/x_j$. The harmonic mean is then $1/\bar{u}$, and its confidence interval is $[1/U, 1/L]$ if L is greater than zero. If L is not greater than zero, this confidence interval is not defined, and missing values are reported.

When weights are specified, `ameans` applies the weights to the transformed values, $u_j = \ln x_j$ and $u_j = 1/x_j$, respectively, when computing the geometric and harmonic means. For details on how the weights are used to compute the mean and variance of the u_j , see [\[R\] summarize](#). Without weights, the formula for the geometric mean reduces to

$$\exp\left\{\frac{1}{n} \sum_j \ln(x_j)\right\}$$

Without weights, the formula for the harmonic mean is

$$\frac{n}{\sum_j \frac{1}{x_j}}$$

Acknowledgments

This improved version of `ameans` is based on the `gmci` command ([Carlin, Vidmar, and Ramalheira 1998](#)) and was written by John Carlin, University of Melbourne, Australia; Suzanna Vidmar, University of Melbourne, Australia; and Carlos Ramalheira, Coimbra University Hospital, Portugal.

References

- Armitage, P., G. Berry, and J. N. S. Matthews. 2002. *Statistical Methods in Medical Research*. 4th ed. Oxford: Blackwell.
- Carlin, J., S. Vidmar, and C. Ramalheira. 1998. [sg75: Geometric means and confidence intervals](#). *Stata Technical Bulletin* 41: 23–25. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 197–199. College Station, TX: Stata Press.
- Keynes, J. M. 1911. The principal averages and the laws of error which lead to them. *Journal of the Royal Statistical Society* 74: 322–331.
- Plackett, R. L. 1958. Studies in the history of probability and statistics: VII. The principle of the arithmetic mean. *Biometrika* 45: 130–135.

- Snedecor, G. W., and W. G. Cochran. 1989. *Statistical Methods*. 8th ed. Ames, IA: Iowa State University Press.
- Stigler, S. M. 1985. Arithmetic means. In Vol. 1 of *Encyclopedia of Statistical Sciences*, ed. S. Kotz and N. L. Johnson, 126–129. New York: Wiley.

Also see

- [R] [ci](#) — Confidence intervals for means, proportions, and counts
- [R] [mean](#) — Estimate means
- [R] [summarize](#) — Summary statistics
- [SVY] [svy estimation](#) — Estimation commands for survey data

Syntax

`anova` *varname* [*termlist*] [*if*] [*in*] [*weight*] [*, options*]

where *termlist* is a factor-variable list (see [U] 11.4.3 Factor variables) with the following additional features:

- Variables are assumed to be categorical; use the `c.` factor-variable operator to override this.
- The `|` symbol (indicating nesting) may be used in place of the `#` symbol (indicating interaction).
- The `/` symbol is allowed after a term and indicates that the following term is the error term for the preceding terms.

<i>options</i>	description
Model	
<code>repeated(<i>varlist</i>)</code>	variables in <i>terms</i> that are repeated-measures variables
<code>partial</code>	use partial (or marginal) sums of squares
<code>sequential</code>	use sequential sums of squares
<code>noconstant</code>	suppress constant term
<code>dropemptycells</code>	drop empty cells from the design matrix
Adv. model	
<code>bse(<i>term</i>)</code>	between-subjects error term in repeated-measures ANOVA
<code>bseunit(<i>varname</i>)</code>	variable representing lowest unit in the between-subjects error term
<code>grouping(<i>varname</i>)</code>	grouping variable for computing pooled covariance matrix

`bootstrap`, `by`, `jackknife`, and `statsby` are allowed; see [U] 11.1.10 Prefix commands.
`aweight`s and `fweight`s are allowed; see [U] 11.1.6 `weight`.
See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Linear models and related > ANOVA/MANOVA > Analysis of variance and covariance

Description

The `anova` command fits analysis-of-variance (ANOVA) and analysis-of-covariance (ANCOVA) models for balanced and unbalanced designs, including designs with missing cells; for repeated-measures ANOVA; and for factorial, nested, or mixed designs.

The `regress` command (see [R] `regress`) will display the coefficients, standard errors, etc., of the regression model underlying the last run of `anova`.

If you want to fit one-way ANOVA models, you may find the `oneway` or `loneway` command more convenient; see [R] `oneway` and [R] `loneway`. If you are interested in MANOVA or MANCOVA, see [MV] `manova`.

Options

Model

`repeated(varlist)` indicates the names of the categorical variables in the *terms* that are to be treated as repeated-measures variables in a repeated-measures ANOVA or ANCOVA.

`partial` presents the ANOVA table using partial (or marginal) sums of squares. This setting is the default. Also see the `sequential` option.

`sequential` presents the ANOVA table using sequential sums of squares.

`noconstant` suppresses the constant term (intercept) from the ANOVA or regression model.

`dropemptycells` drops empty cells from the design matrix. If `c(emptycells)` is set to `keep` (see [\[R\] set emptycells](#)), this option temporarily resets it to `drop` before running the ANOVA model. If `c(emptycells)` is already set to `drop`, this option does nothing.

Adv. model

`bse(term)` indicates the between-subjects error term in a repeated-measures ANOVA. This option is needed only in the rare case when the `anova` command cannot automatically determine the between-subjects error term.

`bseunit(varname)` indicates the variable representing the lowest unit in the between-subjects error term in a repeated-measures ANOVA. This option is rarely needed because the `anova` command automatically selects the first variable listed in the between-subjects error term as the default for this option.

`grouping(varname)` indicates a variable that determines which observations are grouped together in computing the covariance matrices that will be pooled and used in a repeated-measures ANOVA. This option is rarely needed because the `anova` command automatically selects the combination of all variables except the first (or as specified in the `bseunit()` option) in the between-subjects error term as the default for grouping observations.

Remarks

Remarks are presented under the following headings:

[Introduction](#)
[One-way ANOVA](#)
[Two-way ANOVA](#)
[N-way ANOVA](#)
[Weighted data](#)
[ANCOVA](#)
[Nested designs](#)
[Mixed designs](#)
[Latin-square designs](#)
[Repeated-measures ANOVA](#)

Introduction

`anova` uses least squares to fit the linear models known as ANOVA or ANCOVA (henceforth referred to simply as ANOVA models).

If your interest is in one-way ANOVA, you may find the `oneway` command to be more convenient; see [\[R\] oneway](#).

ANOVA was pioneered by Fisher. It features prominently in his texts on statistical methods and his design of experiments (1925, 1935). Many books discuss ANOVA; see, for instance, [Altman \(1991\)](#); [van Belle et al. \(2004\)](#); [Cobb \(1998\)](#); [Snedecor and Cochran \(1989\)](#); or [Winer, Brown, and Michels \(1991\)](#). For a classic source, see [Scheffé \(1959\)](#). [Kennedy Jr. and Gentle \(1980\)](#) discuss ANOVA’s computing problems. [Edwards \(1985\)](#) is concerned primarily with the relationship between multiple regression and ANOVA. [Acock \(2008, chap. 9\)](#) and [Rabe-Hesketh and Everitt \(2007, chap. 4 and 5\)](#) illustrate their discussion with Stata output. Repeated-measures ANOVA is discussed in [Winer, Brown, and Michels \(1991\)](#); [Kuehl \(2000\)](#); and [Milliken and Johnson \(1984\)](#). Pioneering work in repeated-measures ANOVA can be found in [Box \(1954\)](#); [Geisser and Greenhouse \(1958\)](#); [Huynh and Feldt \(1976\)](#); and [Huynh \(1978\)](#).

One-way ANOVA

`anova`, entered without options, performs and reports standard ANOVA. For instance, to perform a one-way layout of a variable called `endog` on `exog`, you would type `anova endog exog`.

► Example 1

We run an experiment varying the amount of fertilizer used in growing apple trees. We test four concentrations, using each concentration in three groves of 12 trees each. Later in the year, we measure the average weight of the fruit.

If all had gone well, we would have had 3 observations on the average weight for each of the four concentrations. Instead, two of the groves were mistakenly leveled by a confused man on a large bulldozer. We are left with the following data:

```
. use http://www.stata-press.com/data/r11/apple
(Apple trees)
. list, abbrev(10) sepby(treatment)
```

	treatment	weight
1.	1	117.5
2.	1	113.8
3.	1	104.4
4.	2	48.9
5.	2	50.4
6.	2	58.9
7.	3	70.4
8.	3	86.9
9.	4	87.7
10.	4	67.3

To obtain one-way ANOVA results, we type

```
. anova weight treatment
```

	Number of obs = 10		R-squared = 0.9147		
	Root MSE = 9.07002		Adj R-squared = 0.8721		
Source	Partial SS	df	MS	F	Prob > F
Model	5295.54433	3	1765.18144	21.46	0.0013
treatment	5295.54433	3	1765.18144	21.46	0.0013
Residual	493.591667	6	82.2652778		
Total	5789.136	9	643.237333		

We find significant (at better than the 1% level) differences among the four concentrations.

Although the output is a usual ANOVA table, let's run through it anyway. Above the table is a summary of the underlying regression. The model was fit on 10 observations, and the root mean squared error (Root MSE) is 9.07. The R^2 for the model is 0.9147, and the adjusted R^2 is 0.8721.

The first line of the table summarizes the model. The sum of squares (Partial SS) for the model is 5295.5 with 3 degrees of freedom (df). This line results in a mean square (MS) of $5295.5/3 \approx 1765.2$. The corresponding F statistic is 21.46 and has a significance level of 0.0013. Thus the model appears to be significant at the 0.13% level.

The next line summarizes the first (and only) term in the model, `treatment`. Because there is only one term, the line is identical to that for the overall model.

The third line summarizes the residual. The residual sum of squares is 493.59 with 6 degrees of freedom, resulting in a mean squared error of 82.27. The square root of this latter number is reported as the Root MSE.

The model plus the residual sum of squares equals the total sum of squares, which is reported as 5789.1 in the last line of the table. This is the total sum of squares of `weight` after removal of the mean. Similarly, the model plus the residual degrees of freedom sum to the total degrees of freedom, 9. Remember that there are 10 observations. Subtracting 1 for the mean, we are left with 9 total degrees of freedom.

◀

□ Technical note

Rather than using the `anova` command, we could have performed this analysis by using the `oneway` command. [Example 1](#) in [\[R\] oneway](#) repeats this same analysis. You may wish to compare the output.

□

Type `regress` to see the underlying regression model corresponding to an ANOVA model fit using the `anova` command.

▷ Example 2

Returning to the apple tree experiment, we found that the fertilizer concentration appears to significantly affect the average weight of the fruit. Although that finding is interesting, we next want to know which concentration appears to grow the heaviest fruit. One way to find out is by examining the underlying regression coefficients.

```
. regress, baselevels
```

Source	SS	df	MS
Model	5295.54433	3	1765.18144
Residual	493.591667	6	82.2652778
Total	5789.136	9	643.237333

Number of obs = 10

F(3, 6) = 21.46

Prob > F = 0.0013

R-squared = 0.9147

Adj R-squared = 0.8721

Root MSE = 9.07

weight	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
treatment						
1	(base)					
2	-59.16667	7.405641	-7.99	0.000	-77.28762	-41.04572
3	-33.25	8.279758	-4.02	0.007	-53.50984	-12.99016
4	-34.4	8.279758	-4.15	0.006	-54.65984	-14.14016
_cons	111.9	5.236579	21.37	0.000	99.08655	124.7134

See [R] [regress](#) for an explanation of how to read this table. The `baselevels` option of `regress` displays a row indicating the base category for our categorical variable, `treatment`. In summary, we find that concentration 1, the base (omitted) group, produces significantly heavier fruits than concentration 2, 3, and 4; concentration 2 produces the lightest fruits; and concentrations 3 and 4 appear to be roughly equivalent.



➤ **Example 3**

We previously typed `anova weight treatment` to produce and display the ANOVA table for our apple tree experiment. Typing `regress` displays the regression coefficients. We can redisplay the ANOVA table by typing `anova` without arguments:

. anova

	Number of obs =	10	R-squared =	0.9147	
	Root MSE =	9.07002	Adj R-squared =	0.8721	
Source	Partial SS	df	MS	F	Prob > F
Model	5295.54433	3	1765.18144	21.46	0.0013
treatment	5295.54433	3	1765.18144	21.46	0.0013
Residual	493.591667	6	82.2652778		
Total	5789.136	9	643.237333		



Two-way ANOVA

You can include multiple explanatory variables with the `anova` command, and you can specify interactions by placing ‘#’ between the variable names. For instance, typing `anova y a b` performs a two-way layout of `y` on `a` and `b`. Typing `anova y a b a#b` performs a full two-way factorial layout. The shorthand `anova y a##b` does the same.

With the default partial sums of squares, when you specify interacted terms, the order of the terms does not matter. Typing `anova y a b a#b` is the same as typing `anova y b a b#a`.

► Example 4

The classic two-way factorial ANOVA problem, at least as far as computer manuals are concerned, is a two-way ANOVA design from [Afifi and Azen \(1979\)](#).

Fifty-eight patients, each suffering from one of three different diseases, were randomly assigned to one of four different drug treatments, and the change in their systolic blood pressure was recorded. Here are the data:

	Disease 1	Disease 2	Disease 3
Drug 1	42, 44, 36 13, 19, 22	33, 26, 33 21	31, -3, 25 25, 24
Drug 2	28, 23, 34 42, 13	34, 33, 31 36	3, 26, 28 32, 4, 16
Drug 3	1, 29, 19	11, 9, 7 1, -6	21, 1, 9 3
Drug 4	24, 9, 22 -2, 15	27, 12, 12 -5, 16, 15	22, 7, 25 5, 12

Let's assume that we have entered these data into Stata and stored the data as `systolic.dta`. Below we use the data, list the first 10 observations, summarize the variables, and tabulate the control variables:

```
. use http://www.stata-press.com/data/r11/systolic
(Systolic Blood Pressure Data)
. list in 1/10
```

	drug	disease	systolic
1.	1	1	42
2.	1	1	44
3.	1	1	36
4.	1	1	13
5.	1	1	19
6.	1	1	22
7.	1	2	33
8.	1	2	26
9.	1	2	33
10.	1	2	21

```
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
drug	58	2.5	1.158493	1	4
disease	58	2.017241	.8269873	1	3
systolic	58	18.87931	12.80087	-6	44

```
. tabulate drug disease
```

Drug Used	Patient's Disease			Total
	1	2	3	
1	6	4	5	15
2	5	4	6	15
3	3	5	4	12
4	5	6	5	16
Total	19	19	20	58

Each observation in our data corresponds to one patient, and for each patient we record drug, disease, and the increase in the systolic blood pressure, `systolic`. The tabulation reveals that the data are not balanced—there are not equal numbers of patients in each drug–disease cell. Stata does not require that the data be balanced. We can perform a two-way factorial ANOVA by typing

```
. anova systolic drug disease drug#disease
```

	Number of obs =	58	R-squared =	0.4560	
	Root MSE =	10.5096	Adj R-squared =	0.3259	
Source	Partial SS	df	MS	F	Prob > F
Model	4259.33851	11	387.212591	3.51	0.0013
drug	2997.47186	3	999.157287	9.05	0.0001
disease	415.873046	2	207.936523	1.88	0.1637
drug#disease	707.266259	6	117.87771	1.07	0.3958
Residual	5080.81667	46	110.452536		
Total	9340.15517	57	163.862371		

Although Stata’s `table` command does not perform ANOVA, it can produce useful summary tables of your data (see [\[R\] table](#)):

```
. table drug disease, c(mean systolic) row col f(%8.2f)
```

Drug Used	Patient’s Disease			
	1	2	3	Total
1	29.33	28.25	20.40	26.07
2	28.00	33.50	18.17	25.53
3	16.33	4.40	8.50	8.75
4	13.60	12.83	14.20	13.50
Total	22.79	18.21	15.80	18.88

These are simple means and are not influenced by our `anova` model. More useful is the `margins` command (see [\[R\] margins](#)) that provides marginal means and adjusted predictions. Because `drug` is the only significant factor in our ANOVA, we now examine the adjusted marginal means for `drug`.

```
. margins drug, asbalanced
Adjusted predictions                                Number of obs   =           58
Expression   : Linear prediction, predict()
at           : drug                               (asbalanced)
              disease                             (asbalanced)
```

	Delta-method				
	Margin	Std. Err.	z	P> z	[95% Conf. Interval]
drug					
1	25.99444	2.751008	9.45	0.000	20.60257 31.38632
2	26.55556	2.751008	9.65	0.000	21.16368 31.94743
3	9.744444	3.100558	3.14	0.002	3.667462 15.82143
4	13.54444	2.637123	5.14	0.000	8.375778 18.71311

These adjusted marginal predictions are not equal to the simple drug means (see the total column from the `table` command); they are based upon predictions from our ANOVA model. The `asbalanced` option of `margins` corresponds with the interpretation of the *F* statistic produced by ANOVA—each cell is given equal weight regardless of its sample size (see the following three technical notes). You

can omit the `asbalanced` option and obtain predictive margins that take into account the unequal sample sizes of the cells.

```
. margins drug
Predictive margins                                Number of obs   =           58
Expression   : Linear prediction, predict()
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
drug						
1	25.89799	2.750533	9.42	0.000	20.50704	31.28893
2	26.41092	2.742762	9.63	0.000	21.0352	31.78664
3	9.722989	3.099185	3.14	0.002	3.648697	15.79728
4	13.55575	2.640602	5.13	0.000	8.380261	18.73123

◀

□ Technical note

How do you interpret the significance of terms like `drug` and `disease` in unbalanced data? If you are familiar with SAS, the sums of squares and the F statistic reported by Stata correspond to SAS type III sums of squares. (Stata can also calculate sequential sums of squares, but we will postpone that topic for now.)

Let's think in terms of the following table:

	Disease 1	Disease 2	Disease 3	
Drug 1	μ_{11}	μ_{12}	μ_{13}	$\mu_{1\cdot}$
Drug 2	μ_{21}	μ_{22}	μ_{23}	$\mu_{2\cdot}$
Drug 3	μ_{31}	μ_{32}	μ_{33}	$\mu_{3\cdot}$
Drug 4	μ_{41}	μ_{42}	μ_{43}	$\mu_{4\cdot}$
	$\mu_{\cdot 1}$	$\mu_{\cdot 2}$	$\mu_{\cdot 3}$	$\mu_{\cdot\cdot}$

In this table, μ_{ij} is the mean increase in systolic blood pressure associated with drug i and disease j , while $\mu_{i\cdot}$ is the mean for drug i , $\mu_{\cdot j}$ is the mean for disease j , and $\mu_{\cdot\cdot}$ is the overall mean.

If the data are balanced, meaning that there are equal numbers of observations going into the calculation of each mean μ_{ij} , the row means, $\mu_{i\cdot}$, are given by

$$\mu_{i\cdot} = \frac{\mu_{i1} + \mu_{i2} + \mu_{i3}}{3}$$

In our case, the data are not balanced, but we define the $\mu_{i\cdot}$ according to that formula anyway. The test for the main effect of drug is the test that

$$\mu_{1\cdot} = \mu_{2\cdot} = \mu_{3\cdot} = \mu_{4\cdot}.$$

To be absolutely clear, the F test of the term `drug`, called the *main effect* of drug, is formally equivalent to the test of the three constraints:

$$\frac{\mu_{11} + \mu_{12} + \mu_{13}}{3} = \frac{\mu_{21} + \mu_{22} + \mu_{23}}{3}$$
$$\frac{\mu_{11} + \mu_{12} + \mu_{13}}{3} = \frac{\mu_{31} + \mu_{32} + \mu_{33}}{3}$$
$$\frac{\mu_{11} + \mu_{12} + \mu_{13}}{3} = \frac{\mu_{41} + \mu_{42} + \mu_{43}}{3}$$

In our data, we obtain a significant F statistic of 9.05 and thus reject those constraints. □

□ Technical note

Stata can display the symbolic form underlying the test statistics it presents, as well as display other test statistics and their symbolic forms; see *Obtaining symbolic forms* in [\[R\] anova postestimation](#). Here is the result of requesting the symbolic form for the main effect of `drug` in our data:

```
. test drug, symbolic
drug
      1  -(r2+r3+r4)
      2   r2
      3   r3
      4   r4
disease
      1   0
      2   0
      3   0
drug#disease
      1  1  -1/3 (r2+r3+r4)
      1  2  -1/3 (r2+r3+r4)
      1  3  -1/3 (r2+r3+r4)
      2  1   1/3 r2
      2  2   1/3 r2
      2  3   1/3 r2
      3  1   1/3 r3
      3  2   1/3 r3
      3  3   1/3 r3
      4  1   1/3 r4
      4  2   1/3 r4
      4  3   1/3 r4
_cons      0
```

This says exactly what we said in the previous technical note. □

□ Technical note

Saying that there is no main effect of a variable is not the same as saying that it has no effect at all. Stata’s ability to perform ANOVA on unbalanced data can easily be put to ill use.

For example, consider the following table of the probability of surviving a bout with one of two diseases according to the drug administered to you:

	Disease 1	Disease 2
Drug 1	1	0
Drug 2	0	1

If you have disease 1 and are administered drug 1, you live. If you have disease 2 and are administered drug 2, you live. In all other cases, you die.

This table has no main effects of either drug or disease, although there is a large interaction effect. You might now be tempted to reason that because there is only an interaction effect, you would be indifferent between the two drugs in the absence of knowledge about which disease infects you. Given an equal chance of having either disease, you reason that it does not matter which drug is administered to you—either way, your chances of surviving are 0.5.

You may not, however, have an equal chance of having either disease. If you knew that disease 1 was 100 times more likely to occur in the population, and if you knew that you had one of the two diseases, you would express a strong preference for receiving drug 1.

When you calculate the significance of main effects on unbalanced data, you must ask yourself why the data are unbalanced. If the data are unbalanced for random reasons and you are making predictions for a balanced population, the test of the main effect makes perfect sense. If, however, the data are unbalanced because the underlying populations are unbalanced and you are making predictions for such unbalanced populations, the test of the main effect may be practically—if not statistically—meaningless.

□

► Example 5

Stata can perform ANOVA not only on unbalanced populations, but also on populations that are so unbalanced that entire cells are missing. For instance, using our systolic blood pressure data, let's refit the model eliminating the drug 1–disease 1 cell. Because `anova` follows the same syntax as all other Stata commands, we can explicitly specify the data to be used by typing the `if` qualifier at the end of the `anova` command. Here we want to use the data that are not for drug 1 and disease 1:

```
. anova systolic drug##disease if !(drug==1 & disease==1)
```

	Number of obs =	52	R-squared =	0.4545	
	Root MSE =	10.1615	Adj R-squared =	0.3215	
Source	Partial SS	df	MS	F	Prob > F
Model	3527.95897	10	352.795897	3.42	0.0025
drug	2686.57832	3	895.526107	8.67	0.0001
disease	327.792598	2	163.896299	1.59	0.2168
drug#disease	703.007602	5	140.60152	1.36	0.2586
Residual	4233.48333	41	103.255691		
Total	7761.44231	51	152.185143		

Here we used `drug##disease` as a shorthand for `drug disease drug#disease`.

◀

□ Technical note

The test of the main effect of drug in the presence of missing cells is more complicated than that for unbalanced data. Our underlying tableau now has the following form:

	Disease 1	Disease 2	Disease 3	
Drug 1		μ_{12}	μ_{13}	
Drug 2	μ_{21}	μ_{22}	μ_{23}	$\mu_{2\cdot}$
Drug 3	μ_{31}	μ_{32}	μ_{33}	$\mu_{3\cdot}$
Drug 4	μ_{41}	μ_{42}	μ_{43}	$\mu_{4\cdot}$
		$\mu_{\cdot 2}$	$\mu_{\cdot 3}$	

The hole in the drug 1–disease 1 cell indicates that the mean is unobserved. Considering the main effect of drug, the test is unchanged for the rows in which all the cells are defined:

$$\mu_{2\cdot} = \mu_{3\cdot} = \mu_{4\cdot}.$$

The first row, however, requires special attention. Here we want the average outcome for drug 1, which is averaged only over diseases 2 and 3, to be equal to the average values of all other drugs averaged over those same two diseases:

$$\frac{\mu_{12} + \mu_{13}}{2} = \frac{(\mu_{22} + \mu_{23})/2 + (\mu_{32} + \mu_{33})/2 + (\mu_{42} + \mu_{43})/2}{3}$$

Thus the test contains three constraints:

$$\begin{aligned} \frac{\mu_{21} + \mu_{22} + \mu_{23}}{3} &= \frac{\mu_{31} + \mu_{32} + \mu_{33}}{3} \\ \frac{\mu_{21} + \mu_{22} + \mu_{23}}{3} &= \frac{\mu_{41} + \mu_{42} + \mu_{43}}{3} \\ \frac{\mu_{12} + \mu_{13}}{2} &= \frac{\mu_{22} + \mu_{23} + \mu_{32} + \mu_{33} + \mu_{42} + \mu_{43}}{6} \end{aligned}$$

□

Stata can calculate two types of sums of squares, *partial* and *sequential*. If you do not specify which sums of squares to calculate, Stata calculates partial sums of squares. The technical notes above have gone into great detail about the definition and use of partial sums of squares. Use the `sequential` option to obtain sequential sums of squares.

□ Technical note

Before we illustrate sequential sums of squares, consider one more feature of the partial sums. If you know how such things are calculated, you may worry that the terms must be specified in some particular order, that Stata would balk or, even worse, produce different results if you typed, say, `anova drug#disease drug disease` rather than `anova drug disease drug#disease`. We assure you that is not the case.

When you type a model, Stata internally reorganizes the terms, forms the cross-product matrix, inverts it, converts the result to an upper-Hermite form, and then performs the hypothesis tests. As a final touch, Stata reports the results in the same order that you typed the terms.

□

► Example 6

We wish to estimate the effects on systolic blood pressure of drug and disease by using sequential sums of squares. We want to introduce disease first, then drug, and finally, the interaction of drug and disease:

```
. anova systolic disease drug disease#drug, sequential
```

	Number of obs =	58	R-squared =	0.4560
	Root MSE =	10.5096	Adj R-squared =	0.3259

Source	Seq. SS	df	MS	F	Prob > F
Model	4259.33851	11	387.212591	3.51	0.0013
disease	488.639383	2	244.319691	2.21	0.1210
drug	3063.43286	3	1021.14429	9.25	0.0001
disease#drug	707.266259	6	117.87771	1.07	0.3958
Residual	5080.81667	46	110.452536		
Total	9340.15517	57	163.862371		

The F statistic on disease is now 2.21. When we fit this same model by using partial sums of squares, the statistic was 1.88.

◀

N-way ANOVA

You may include high-order interaction terms, such as a third-order interaction between the variables A, B, and C, by typing `A#B#C`.

► Example 7

We wish to determine the operating conditions that maximize yield for a manufacturing process. There are three temperature settings, two chemical supply companies, and two mixing methods under investigation. Three observations are obtained for each combination of these three factors.

```
. use http://www.stata-press.com/data/r11/manuf
(manufacturing process data)

. describe
```

Contains data from <http://www.stata-press.com/data/r11/manuf.dta>

obs:	36	manufacturing process data
vars:	4	2 Jan 2009 13:28
size:	288 (99.9% of memory free)	

variable name	storage type	display format	value label	variable label
temperature	byte	%9.0g	temp	machine temperature setting
chemical	byte	%9.0g	supplier	chemical supplier
method	byte	%9.0g	meth	mixing method
yield	byte	%9.0g		product yield

Sorted by:

We wish to perform a three-way factorial ANOVA. We could type

```
. anova yield temp chem temp#chem meth temp#meth chem#meth temp#chem#meth
```

but prefer to use the `##` factor-variable operator for brevity.

```
. anova yield temp##chem##meth
```

	Number of obs =	36	R-squared =	0.5474	
	Root MSE =	2.62996	Adj R-squared =	0.3399	
Source	Partial SS	df	MS	F	Prob > F
Model	200.75	11	18.25	2.64	0.0227
temperature	30.5	2	15.25	2.20	0.1321
chemical	12.25	1	12.25	1.77	0.1958
temperature#chemical	24.5	2	12.25	1.77	0.1917
method	42.25	1	42.25	6.11	0.0209
temperature#method	87.5	2	43.75	6.33	0.0062
chemical#method	.25	1	.25	0.04	0.8508
temperature#chemical#method	3.5	2	1.75	0.25	0.7785
Residual	166	24	6.9166667		
Total	366.75	35	10.4785714		

The interaction between temperature and method appears to be the important story in these data. A table of means for this interaction is given below.

```
. table method temp, c(mean yield) row col f(%8.2f)
```

mixing method	machine temperature setting			
	low	medium	high	Total
stir	7.50	6.00	6.00	6.50
fold	5.50	9.00	11.50	8.67
Total	6.50	7.50	8.75	7.58

Here our ANOVA is balanced (each cell has the same number of observations), and we obtain the same values as in the table above (but with additional information such as confidence intervals) by using the `margins` command. Because our ANOVA is balanced, using the `asbalanced` option with `margins` would not produce different results. We request the predictive margins for the two terms that appear significant in our ANOVA: `temperature#method` and `method`.


```
. margins temperature#method method
```

```
Predictive margins                                Number of obs   =           36
```

```
Expression   : Linear prediction, predict()
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
temperature#						
method						
1 1	7.5	1.073675	6.99	0.000	5.395636	9.604364
1 2	5.5	1.073675	5.12	0.000	3.395636	7.604364
2 1	6	1.073675	5.59	0.000	3.895636	8.104364
2 2	9	1.073675	8.38	0.000	6.895636	11.10436
3 1	6	1.073675	5.59	0.000	3.895636	8.104364
3 2	11.5	1.073675	10.71	0.000	9.395636	13.60436
method						
1	6.5	.6198865	10.49	0.000	5.285045	7.714955
2	8.666667	.6198865	13.98	0.000	7.451711	9.881622

We decide to use the folding method of mixing and a high temperature in our manufacturing process.



Weighted data

Like all estimation commands, `anova` can produce estimates on weighted data. See [U] [11.1.6 weight](#) for details on specifying the weight.

► Example 8

We wish to investigate the prevalence of byssinosis, a form of pneumoconiosis that can afflict workers exposed to cotton dust. We have data on 5,419 workers in a large cotton mill. We know whether each worker smokes, his or her race, and the dustiness of the work area. The variables are

```
smokes      smoker or nonsmoker in the last five years
race        white or other
workplace   1 (most dusty), 2 (less dusty), 3 (least dusty)
```

We wish to fit an ANOVA model explaining the prevalence of byssinosis according to a full factorial model of `smokes`, `race`, and `workplace`.

The data are unbalanced. Moreover, although we have data on 5,419 workers, the data are grouped according to the explanatory variables, along with some other variables, resulting in 72 observations. For each observation, we know the number of workers in the group (`pop`), the prevalence of byssinosis (`prob`), and the values of the three explanatory variables. Thus we wish to fit a three-way factorial model on grouped data.

We begin by showing a bit of the data, which are from [Higgins and Koch \(1977\)](#).

```
. use http://www.stata-press.com/data/r11/byssin
(Byssinosis incidence)

. describe

Contains data from http://www.stata-press.com/data/r11/byssin.dta
  obs:                72                Byssinosis incidence
  vars:                 5                19 Dec 2008 07:04
  size:              1,152 (99.9% of memory free)
```

variable name	storage type	display format	value label	variable label
smokes	int	%8.0g	smokes	Smokes
race	int	%8.0g	race	Race
workplace	int	%8.0g	workplace	Dustiness of workplace
pop	int	%8.0g		Population size
prob	float	%9.0g		Prevalence of byssinosis

```
Sorted by:

. list in 1/5, abbrev(10) divider
```

	smokes	race	workplace	pop	prob
1.	yes	white	most	40	.075
2.	yes	white	less	74	0
3.	yes	white	least	260	.0076923
4.	yes	other	most	164	.152439
5.	yes	other	less	88	0

The first observation in the data represents a group of 40 white workers who smoke and work in a “most” dusty work area. Of those 40 workers, 7.5% have byssinosis. The second observation represents a group of 74 white workers who also smoke but who work in a “less” dusty environment. None of those workers has byssinosis.

Almost every Stata command allows weights. Here we want to weight the data by `pop`. We can, for instance, make a table of the number of workers by their smoking status and race:

```
. tabulate smokes race [fw=pop]
```

Smokes	Race		Total
	other	white	
no	799	1,431	2,230
yes	1,104	2,085	3,189
Total	1,903	3,516	5,419

The `[fw=pop]` at the end of the `tabulate` command tells Stata to count each observation as representing `pop` persons. When making the tally, `tabulate` treats the first observation as representing 40 workers, the second as representing 74 workers, and so on.

Similarly, we can make a table of the dustiness of the workplace:

```
. tabulate workplace [fw=pop]
```

Dustiness of workplace	Freq.	Percent	Cum.
least	3,450	63.66	63.66
less	1,300	23.99	87.65
most	669	12.35	100.00
Total	5,419	100.00	

We can discover the average incidence of byssinosis among these workers by typing

```
. summarize prob [fw=pop]
```

Variable	Obs	Mean	Std. Dev.	Min	Max
prob	5419	.0304484	.0567373	0	.287037

We discover that 3.04% of these workers have byssinosis. Across all cells, the byssinosis rates vary from 0 to 28.7%. Just to prove that there might be something here, let's obtain the average incidence rates according to the dustiness of the workplace:

```
. table workplace smokes race [fw=pop], c(mean prob)
```

Dustiness of workplace	Race and Smokes			
	other		white	
	no	yes	no	yes
least	.0107527	.0101523	.0081549	.0162774
less	.02	.0081633	.0136612	.0143149
most	.0820896	.1679105	.0833333	.2295082

Let's now fit the ANOVA model.

```
. anova prob workplace smokes race workplace#smokes workplace#race
```

```
> smokes#race workplace#smokes#race [aweight=pop]
```

```
(sum of wgt is 5.4190e+03)
```

	Number of obs =	65	R-squared =	0.8300	
	Root MSE =	.025902	Adj R-squared =	0.7948	
Source	Partial SS	df	MS	F	Prob > F
Model	.173646538	11	.015786049	23.53	0.0000
workplace	.097625175	2	.048812588	72.76	0.0000
smokes	.013030812	1	.013030812	19.42	0.0001
race	.001094723	1	.001094723	1.63	0.2070
workplace#smokes	.019690342	2	.009845171	14.67	0.0000
workplace#race	.001352516	2	.000676258	1.01	0.3718
smokes#race	.001662874	1	.001662874	2.48	0.1214
workplace#smokes#race	.000950841	2	.00047542	0.71	0.4969
Residual	.035557766	53	.000670901		
Total	.209204304	64	.003268817		

Of course, if we want to see the underlying regression, we could type `regress`.

Above we examined simple means of the cells of `workplace#smokes#race`. Our ANOVA shows `workplace`, `smokes`, and their interaction as being the only significant factors in our model. We now examine the predictive marginal mean byssinosis rates for these terms.

```
. margins workplace#smokes workplace smokes
Predictive margins                                Number of obs   =          65
Expression   : Linear prediction, predict()
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
workplace#smokes						
1 1	.0090672	.0062319	1.45	0.146	-.003147	.0212814
1 2	.0141264	.0053231	2.65	0.008	.0036934	.0245595
2 1	.0158872	.009941	1.60	0.110	-.0035967	.0353711
2 2	.0121546	.0087353	1.39	0.164	-.0049663	.0292756
3 1	.0828966	.0182151	4.55	0.000	.0471957	.1185975
3 2	.2078768	.012426	16.73	0.000	.1835222	.2322314
workplace						
1	.0120701	.0040471	2.98	0.003	.0041379	.0200022
2	.0137273	.0065685	2.09	0.037	.0008533	.0266012
3	.1566225	.0104602	14.97	0.000	.1361208	.1771241
smokes						
1	.0196915	.0050298	3.91	0.000	.0098332	.0295498
2	.0358626	.0041949	8.55	0.000	.0276408	.0440844

Smoking combined with the most dusty workplace produces the highest byssinosis rates.



Ronald Aylmer Fisher (1890–1962) (Sir Ronald from 1952) studied mathematics at Cambridge. Even before he finished his studies, he had published on statistics. He worked as a statistician at Rothamsted Experimental Station (1919–1933), as professor of eugenics at University College London (1933–1943), as professor of genetics at Cambridge (1943–1957), and in retirement at the CSIRO Division of Mathematical Statistics in Adelaide. His many fundamental and applied contributions to statistics and genetics mark him as one of the greatest statisticians of all time, including original work on tests of significance, distribution theory, theory of estimation, fiducial inference, and design of experiments.

ANCOVA

You can include multiple explanatory variables with the `anova` command, but unless you explicitly state otherwise by using the `c.` factor-variable operator, all the variables are interpreted as *categorical variables*. Using the `c.` operator, you can designate variables as *continuous* and thus perform ANCOVA.

➤ Example 9

We have census data recording the death rate (`drate`) and median age (`age`) for each state. The dataset also includes the region of the country in which each state is located (`region`):

```
. use http://www.stata-press.com/data/r11/census2
(1980 Census data by state)
. summarize drate age region
```

Variable	Obs	Mean	Std. Dev.	Min	Max
drate	50	84.3	13.07318	40	107
age	50	29.5	1.752549	24	35
region	50	2.66	1.061574	1	4

age is coded in integral years from 24 to 35, and region is coded from 1 to 4, with 1 standing for the Northeast, 2 for the North Central, 3 for the South, and 4 for the West.

When we examine the data more closely, we discover large differences in the death rate across regions of the country:

```
. tabulate region, summarize(drate)
```

Census region	Summary of Death Rate		Freq.
	Mean	Std. Dev.	
NE	93.444444	7.0553368	9
N Cntrl	88.916667	5.5833899	12
South	88.3125	8.5457104	16
West	68.769231	13.342625	13
Total	84.3	13.073185	50

Naturally, we wonder if these differences might not be explained by differences in the median ages of the populations. To find out, we fit a regression model (via `anova`) of `drate` on `region` and `age`. In the `anova` example below, we treat `age` as a categorical variable.

```
. anova drate region age
```

		Number of obs = 50		R-squared = 0.7927	
		Root MSE = 6.7583		Adj R-squared = 0.7328	
Source	Partial SS	df	MS	F	Prob > F
Model	6638.86529	11	603.533208	13.21	0.0000
region	1320.00973	3	440.003244	9.63	0.0001
age	2237.24937	8	279.656171	6.12	0.0000
Residual	1735.63471	38	45.6745977		
Total	8374.5	49	170.908163		

We have the answer to our question: differences in median ages do not eliminate the differences in death rates across the four regions. The ANOVA table summarizes the two terms in the model, `region` and `age`. The `region` term contains 3 degrees of freedom, and the `age` term contains 8 degrees of freedom. Both are significant at better than the 1% level.

The `age` term contains 8 degrees of freedom. Because we did not explicitly indicate that `age` was to be treated as a continuous variable, it was treated as *categorical*, meaning that unique coefficients were estimated for each level of age. The only clue of this labeling is that the number of degrees of freedom associated with the `age` term exceeds 1. The labeling becomes more obvious if we review the regression coefficients:

```
. regress, baselevels
```

Source	SS	df	MS			
Model	6638.86529	11	603.533208			
Residual	1735.63471	38	45.6745977			
Total	8374.5	49	170.908163			

	Number of obs =	50
	F(11, 38) =	13.21
	Prob > F =	0.0000
	R-squared =	0.7927
	Adj R-squared =	0.7328
	Root MSE =	6.7583

	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
region						
1	(base)					
2	.4428387	3.983664	0.11	0.912	-7.621668	8.507345
3	-.2964637	3.934766	-0.08	0.940	-8.261981	7.669054
4	-13.37147	4.195344	-3.19	0.003	-21.8645	-4.878439
age						
24	(base)					
26	-15	9.557677	-1.57	0.125	-34.34851	4.348506
27	14.30833	7.857378	1.82	0.076	-1.598099	30.21476
28	12.66011	7.495513	1.69	0.099	-2.51376	27.83399
29	18.861	7.28918	2.59	0.014	4.104825	33.61717
30	20.87003	7.210148	2.89	0.006	6.273847	35.46621
31	29.91307	8.242741	3.63	0.001	13.22652	46.59963
32	27.02853	8.509432	3.18	0.003	9.802089	44.25498
35	38.925	9.944825	3.91	0.000	18.79275	59.05724
_cons	68.37147	7.95459	8.60	0.000	52.26824	84.47469

The `regress` command displayed the `anova` model as a regression table. We used the `baselevels` option to display the dropped level (or base) for each term.

If we want to treat `age` as a continuous variable, we must prepend `c.` to `age` in our `anova`.

```
. anova drate region c.age
```

	Number of obs =	50	R-squared =	0.7203	
	Root MSE =	7.21483	Adj R-squared =	0.6954	
Source	Partial SS	df	MS	F	Prob > F
Model	6032.08254	4	1508.02064	28.97	0.0000
region age	1645.66228	3	548.554092	10.54	0.0000
	1630.46662	1	1630.46662	31.32	0.0000
Residual	2342.41746	45	52.0537213		
Total	8374.5	49	170.908163		

The `age` term now has 1 degree of freedom. The regression coefficients are

```
. regress, baselevels
```

Source	SS	df	MS	Number of obs = 50		
Model	6032.08254	4	1508.02064	F(4, 45) = 28.97		
Residual	2342.41746	45	52.0537213	Prob > F = 0.0000		
				R-squared = 0.7203		
				Adj R-squared = 0.6954		
Total	8374.5	49	170.908163	Root MSE = 7.2148		

drate	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
region						
1	(base)					
2	1.792526	3.375925	0.53	0.598	-5.006935	8.591988
3	.6979912	3.18154	0.22	0.827	-5.70996	7.105942
4	-13.37578	3.723447	-3.59	0.001	-20.87519	-5.876377
age	3.922947	.7009425	5.60	0.000	2.511177	5.334718
_cons	-28.60281	21.93931	-1.30	0.199	-72.79085	15.58524

Although we started analyzing these data to explain the regional differences in death rate, let's focus on the effect of age for a moment. In our first model, each level of **age** had a unique death rate associated with it. For instance, the predicted death rate in a north central state with a median age of 28 was

$$0.44 + 12.66 + 68.37 \approx 81.47$$

whereas the predicted death rate from our current model is

$$1.79 + 3.92 \times 28 - 28.60 \approx 82.95$$

Our previous model had an R^2 of 0.7927, whereas our current model has an R^2 of 0.7203. This “small” loss of predictive power accompanies a gain of 7 degrees of freedom, so we suspect that the continuous-age model is as good as the discrete-age model.

◀

□ Technical note

There is enough information in the two ANOVA tables to attach a statistical significance to our suspicion that the loss of predictive power is offset by the savings in degrees of freedom. Because the continuous-age model is nested within the discrete-age model, we can perform a standard Chow test. For those of us who know such formulas off the top of our heads, the F statistic is

$$\frac{(2342.41746 - 1735.63471)/7}{45.6745977} = 1.90$$

There is, however, a better way.

We can find out whether our continuous model is as good as our discrete model by putting **age** in the model twice: once as a continuous variable and once as a categorical variable. The categorical variable will then measure deviations around the straight line implied by the continuous variable, and the F test for the significance of the categorical variable will test whether those deviations are jointly zero.

```
. anova drate region c.age age
```

Number of obs = 50R-squared = 0.7927
Root MSE = 6.7583Adj R-squared = 0.7328

Source	Partial SS	df	MS	F	Prob > F
Model	6638.86529	11	603.533208	13.21	0.0000
region	1320.00973	3	440.003244	9.63	0.0001
age	699.74137	1	699.74137	15.32	0.0004
age	606.782747	7	86.6832496	1.90	0.0970
Residual	1735.63471	38	45.6745977		
Total	8374.5	49	170.908163		

We find that the F test for the significance of the (categorical) `age` variable is 1.90, just as we calculated above. It is significant at the 9.7% level. If we hold to a 5% significance level, we cannot reject the null hypothesis that the effect of `age` is linear.



➤ Example 10

In our census data, we still find significant differences across the regions after controlling for the median age of the population. We might now wonder whether the regional differences are differences in level—independent of age—or are instead differences in the regional effects of age. Just as we can interact categorical variables with other categorical variables, we can interact categorical variables with continuous variables.

```
. anova drate region c.age region#c.age
```

Number of obs = 50R-squared = 0.7365
Root MSE = 7.24852Adj R-squared = 0.6926

Source	Partial SS	df	MS	F	Prob > F
Model	6167.7737	7	881.110529	16.77	0.0000
region	188.713602	3	62.9045339	1.20	0.3225
age	873.425599	1	873.425599	16.62	0.0002
region#age	135.691162	3	45.2303874	0.86	0.4689
Residual	2206.7263	42	52.5411023		
Total	8374.5	49	170.908163		

The `region#c.age` term in our model measures the differences in slopes across the regions. We cannot reject the null hypothesis that there are no such differences. The `region` effect is now “insignificant”. This status does not mean that there are no regional differences in death rates because each test is a *marginal* or *partial* test. Here, with `region#c.age` included in the model, `region` is being tested at the point where `age` is zero. Apart from this value not existing in the dataset, it is also a long way from the mean value of `age`, so the test of `region` at this point is meaningless (although it is valid if you acknowledge what is being tested).

To obtain a more sensible test of `region`, we can subtract the mean from the `age` variable and use this in the model.

```
. quietly summarize age
. generate mage = age - r(mean)
```



```
. anova drate region c.mage region#c.mage
```

	Number of obs =	50	R-squared =	0.7365	
	Root MSE =	7.24852	Adj R-squared =	0.6926	
Source	Partial SS	df	MS	F	Prob > F
Model	6167.7737	7	881.110529	16.77	0.0000
region	1166.14735	3	388.715783	7.40	0.0004
mage	873.425599	1	873.425599	16.62	0.0002
region#mage	135.691162	3	45.2303874	0.86	0.4689
Residual	2206.7263	42	52.5411023		
Total	8374.5	49	170.908163		

region is significant when tested at the mean of the age variable.

◀

Remember that we can specify interactions by typing *varname#varname*. We have seen examples of interacting categorical variables with categorical variables and, in the examples above, a categorical variable (*region*) with a continuous variable (*age* or *mage*).

We can also interact continuous variables with continuous variables. To include an *age*² term in our model, we could type *c.age#c.age*. If we also wanted to interact the categorical variable *region* with the *age*² term, we could type *region#c.age#c.age* (or even *c.age#region#c.age*).

Nested designs

In addition to specifying interaction terms, nested terms can also be specified in an ANOVA. A vertical bar is used to indicate nesting: *A|B* is read as *A* nested within *B*. *A|B|C* is read as *A* nested within *B*, which is nested within *C*. *A|B#C* is read as *A* is nested within the interaction of *B* and *C*. *A#B|C* is read as the interaction of *A* and *B*, which is nested within *C*.

Different error terms can be specified for different parts of the model. The forward slash is used to indicate that the next term in the model is the error term for what precedes it. For instance, *anova y A / B|A* indicates that the *F* test for *A* is to be tested by using the mean square from *B|A* in the denominator. Error terms (terms following the slash) are generally not tested unless they are themselves followed by a slash. Residual error is the default error term.

For example, consider *A / B / C*, where *A*, *B*, and *C* may be arbitrarily complex terms. Then *anova* will report *A* tested by *B* and *B* tested by *C*. If we add one more slash on the end to form *A / B / C /*, then *anova* will also report *C* tested by the residual error.

► Example 11

We have collected data from a manufacturer that is evaluating which of five different brands of machinery to buy to perform a particular function in an assembly line. Twenty assembly-line employees were selected at random for training on these machines, with four employees assigned to learn a particular machine. The output from each employee (operator) on the brand of machine for which he trained was measured during four trial periods. In this example, the operator is nested within machine. Because of sickness and employee resignations, the final data are not balanced. The following table gives the mean output and sample size for each machine and operator combination.

```
. use http://www.stata-press.com/data/r11/machine, clear
(machine data)
```

```
. table machine operator, c(mean output n output) col f(%8.2f)
```

five brands of machine	operator nested in machine				
	1	2	3	4	Total
1	9.15	9.48	8.27	8.20	8.75
	2	4	3	4	13
2	15.03	11.55	11.45	11.52	12.47
	3	2	2	4	11
3	11.27	10.13	11.13		10.84
	3	3	3		9
4	16.10	18.97	15.35	16.60	16.65
	3	3	4	3	13
5	15.30	14.35	10.43		13.63
	4	4	3		11

Assuming that `operator` is random (i.e., we wish to infer to the larger population of possible operators) and `machine` is fixed (i.e., only these five machines are of interest), the typical test for `machine` uses `operator` nested within `machine` as the error term. `operator` nested within `machine` can be tested by residual error. Our earlier warning concerning designs with either unplanned missing cells or unbalanced cell sizes, or both, also applies to interpreting the ANOVA results from this unbalanced nested example.

```
. anova output machine / operator|machine /
```

	Number of obs =	57	R-squared =	0.8661	
	Root MSE =	1.47089	Adj R-squared =	0.8077	
Source	Partial SS	df	MS	F	Prob > F
Model	545.822288	17	32.1071934	14.84	0.0000
machine	430.980792	4	107.745198	13.82	0.0001
operator machine	101.353804	13	7.79644648		
operator machine	101.353804	13	7.79644648	3.60	0.0009
Residual	84.3766582	39	2.16350406		
Total	630.198947	56	11.2535526		

`operator|machine` is preceded by a slash, indicating that it is the error term for the terms before it (here `machine`). `operator|machine` is also followed by a slash that indicates it should be tested with residual error. The output lists the `operator|machine` term twice, once as the error term for `machine`, and again as a term tested by residual error. A line is placed in the ANOVA table to separate the two. In general, a dividing line is placed in the output to separate the terms into groups that are tested with the same error term. The overall model is tested by residual error and is separated from the rest of the table by a blank line at the top of the table.

The results indicate that the machines are not all equal and that there are significant differences between operators.

► Example 12

Your company builds and operates sewage treatment facilities. You want to compare two particulate solutions during the particulate reduction step of the sewage treatment process. For each solution, two area managers are randomly selected to implement and oversee the change to the new treatment process in two of their randomly chosen facilities. Two workers at each of these facilities are trained to operate the new process. A measure of particulate reduction is recorded at various times during the month at each facility for each worker. The data are described below.

```
. use http://www.stata-press.com/data/r11/sewage
(Sewage treatment)

. describe

Contains data from http://www.stata-press.com/data/r11/sewage.dta
   obs:                64                Sewage treatment
  vars:                  5                9 May 2009 12:43
 size:                576 (99.9% of memory free)
```

variable name	storage type	display format	value label	variable label
particulate	byte	%9.0g		particulate reduction
solution	byte	%9.0g		2 particulate solutions
manager	byte	%9.0g		2 managers per solution
facility	byte	%9.0g		2 facilities per manager
worker	byte	%9.0g		2 workers per facility

Sorted by: solution manager facility worker

You want to determine if the two particulate solutions provide significantly different particulate reduction. You would also like to know if manager, facility, and worker are significant effects. solution is a fixed factor, whereas manager, facility, and worker are random factors.

In the following anova command, we use abbreviations for the variable names, which can sometimes make long ANOVA model statements easier to read.

```
. anova particulate s / m|s / f|m|s / w|f|m|s /, dropemptycells
```

	Number of obs =	64	R-squared =	0.6338	
	Root MSE =	12.7445	Adj R-squared =	0.5194	
Source	Partial SS	df	MS	F	Prob > F
Model	13493.6094	15	899.573958	5.54	0.0000
solution	7203.76563	1	7203.76563	17.19	0.0536
manager solution	838.28125	2	419.140625		
manager solution	838.28125	2	419.140625	0.55	0.6166
facility manager solution	3064.9375	4	766.234375		
facility manager solution	3064.9375	4	766.234375	2.57	0.1193
worker facility manager solution	2386.625	8	298.328125		
worker facility manager solution	2386.625	8	298.328125	1.84	0.0931
Residual	7796.25	48	162.421875		
Total	21289.8594	63	337.934276		

While `solution` is not declared significant at the 5% significance level, it is near enough to that threshold to warrant further investigation (see [example 3](#) in [\[R\] anova postestimation](#) for a continuation of the analysis of these data).



□ Technical note

Why did we use the `dropemptycells` option with the previous `anova`? By default, Stata retains empty cells when building the design matrix and currently treats `|` and `#` the same in how it determines the possible number of cells. Retaining empty cells in an ANOVA with nested terms can cause your design matrix to become too large. In [example 12](#), there are $1024 = 2 \times 4 \times 8 \times 16$ cells that are considered possible for the `worker|facility|manager|solution` term because the `worker`, `facility`, and `manager` variables are uniquely numbered. With the `dropemptycells` option, the `worker|facility|manager|solution` term requires just 16 columns in the design matrix (corresponding to the 16 unique workers).

Why did we not use the `dropemptycells` option in [example 11](#), where `operator` is nested in `machine`? If you look at the table presented at the beginning of that example, you will see that `operator` is compactly instead of uniquely numbered (you need both `operator` number and `machine` number to determine the `operator`). Here the `dropemptycells` option would have only reduced our design matrix from 26 columns down to 24 columns (because there were only 3 operators instead of 4 for machines 3 and 5).

We suggest that you specify `dropemptycells` when there are nested terms in your ANOVA. You could also use the `set emptycells drop` command to accomplish the same thing; see [\[R\] set](#).



Mixed designs

An ANOVA can consist of both nested and crossed terms. A split-plot ANOVA design provides an example.

▷ Example 13

Two reading programs and three skill-enhancement techniques are under investigation. Ten classes of first-grade students were randomly assigned so that five classes were taught with one reading program and another five classes were taught with the other. The 30 students in each class were divided into six groups with 5 students each. Within each class, the six groups were divided randomly so that each of the three skill-enhancement techniques was taught to two of the groups within each class. At the end of the school year, a reading assessment test was administered to all the students. In this split-plot ANOVA, the whole-plot treatment is the two reading programs, and the split-plot treatment is the three skill-enhancement techniques.

```
. use http://www.stata-press.com/data/r11/reading
(Reading experiment data)
```

```
. describe
```

```
Contains data from http://www.stata-press.com/data/r11/reading.dta
obs:                300                Reading experiment data
vars:                5                  9 Mar 2009 18:57
size:                2,700 (99.9% of memory free)  (_dta has notes)
```

variable name	storage type	display format	value label	variable label
score	byte	%9.0g		reading score
program	byte	%9.0g		reading program
class	byte	%9.0g		class nested in program
skill	byte	%9.0g		skill enhancement technique
group	byte	%9.0g		group nested in class and skill

Sorted by:

In this split-plot ANOVA, the error term for `program` is `class` nested within `program`. The error term for `skill` and the `program` by `skill` interaction is the `class` by `skill` interaction nested within `program`. Other terms are also involved in the model and can be seen below.

Our `anova` command is too long to fit on one line of this manual. Where we have chosen to break the command into multiple lines is arbitrary. If we were typing this command into Stata, we would just type along and let Stata automatically wrap across lines, as necessary.

```
. anova score prog / class|prog skill prog#skill / class#skill|prog
> / group|class#skill|prog /, dropemptycells
```

		Number of obs =		300	R-squared =		0.3738
		Root MSE =		14.6268	Adj R-squared =		0.2199
Source	Partial SS	df	MS	F	Prob > F		
Model	30656.5167	59	519.601977	2.43	0.0000		
program	4493.07	1	4493.07	8.73	0.0183		
class program	4116.61333	8	514.576667				
skill	1122.64667	2	561.323333	1.54	0.2450		
program#skill	5694.62	2	2847.31	7.80	0.0043		
class#skill program	5841.46667	16	365.091667				
class#skill program	5841.46667	16	365.091667	1.17	0.3463		
group class#skill program	9388.1	30	312.936667				
group class#skill program	9388.1	30	312.936667	1.46	0.0636		
Residual	51346.4	240	213.943333				
Total	82002.9167	299	274.257246				

The `program#skill` term is significant, as is the `program` term for these particular data. Let's look at the predictive margins for these two terms.

```
. margins, within(program skill)
Predictive margins                                Number of obs   =           300
Expression   : Linear prediction, predict()
within       : program skill
```

	Delta-method					
	Margin	Std. Err.	z	P> z	[95% Conf. Interval]	
program# skill						
1 1	68.16	2.068542	32.95	0.000	64.10573	72.21427
1 2	52.86	2.068542	25.55	0.000	48.80573	56.91427
1 3	61.54	2.068542	29.75	0.000	57.48573	65.59427
2 1	50.7	2.068542	24.51	0.000	46.64573	54.75427
2 2	56.54	2.068542	27.33	0.000	52.48573	60.59427
2 3	52.1	2.068542	25.19	0.000	48.04573	56.15427

```
. margins, within(program)
Predictive margins                                Number of obs   =           300
Expression   : Linear prediction, predict()
within       : program
```

	Delta-method					
	Margin	Std. Err.	z	P> z	[95% Conf. Interval]	
program						
1	60.85333	1.194273	50.95	0.000	58.5126	63.19407
2	53.11333	1.194273	44.47	0.000	50.7726	55.45407

Because our ANOVA involves nested terms, we used the `within()` option of `margins`; see [R] [margins](#).

`skill 2` produces a low score when combined with `program 1` and a high score when combined with `program 2`, demonstrating the interaction between the reading program and the skill-enhancement technique. From these tables, you might conclude that the first reading program and the first skill-enhancement technique perform best when combined. However, notice the overlapping confidence interval for the first reading program and the third skill-enhancement technique.

❑ Technical note

There are several valid ways to write complicated `anova` terms. In the reading experiment example (example 13), we had a term `group|class#skill|program`. This term can be read as `group` nested within both `class` and `skill` and further nested within `program`. You can also write this term as `group|class#skill#program` or `group|program#class#skill` or `group|skill#class|program`, etc. All variations will produce the same result. Some people prefer having only one ‘|’ in a term and would use `group|class#skill#program`, which is read as `group` nested within `class`, `skill`, and `program`.

Gertrude Mary Cox (1900–1978) was born on a farm near Dayton, Iowa. Initially intending to become superintendent of an orphanage, she enrolled at Iowa State College, where she majored in mathematics and attained the college’s first Master’s degree in statistics. She started a PhD in psychological statistics at Berkeley but returned to Iowa State after only two years to work with George W. Snedecor. Cox was put in charge of establishing a Computing Laboratory and began to teach design of experiments, the latter leading to her classic text with William G. Cochran. In 1940, Snedecor showed Cox his all-male list of suggestions to head a new statistics department at North Carolina State College and, at her urging, added her name. She was selected and built an outstanding department. Cox retired early to work at the new Research Triangle Institute between Raleigh and Chapel Hill. She consulted widely, served as editor of *Biometrics*, and was elected to the National Academy of Sciences.

Latin-square designs

You can use `anova` to analyze a Latin-square design. Consider the following example, published in [Snedecor and Cochran \(1989\)](#).

► Example 14

Data from a Latin-square design are as follows:

Row	Column 1	Column 2	Column 3	Column 4	Column 5
1	257(B)	230(E)	279(A)	287(C)	202(D)
2	245(D)	283(A)	245(E)	280(B)	260(C)
3	182(E)	252(B)	280(C)	246(D)	250(A)
4	203(A)	204(C)	227(D)	193(E)	259(B)
5	231(C)	271(D)	266(B)	334(A)	338(E)

In Stata, the data might appear as follows:

```
. use http://www.stata-press.com/data/r11/latinsq
. list
```

	row	c1	c2	c3	c4	c5
1.	1	257	230	279	287	202
2.	2	245	283	245	280	260
3.	3	182	252	280	246	250
4.	4	203	204	227	193	259
5.	5	231	271	266	334	338

Before `anova` can be used on these data, the data must be organized so that the outcome measurement is in one column. `reshape` is inadequate for this task because there is information about the treatments in the sequence of these observations. `pkshape` is designed to reshape this type of data; see [\[R\] pkshape](#).

```
. pkshape row row c1-c5, order(beacd daebc ebcda acdeb cdbae)
. list
```

	sequence	outcome	treat	carry	period
1.	1	257	1	0	1
2.	2	245	5	0	1
3.	3	182	2	0	1
4.	4	203	3	0	1
5.	5	231	4	0	1
6.	1	230	2	1	2
7.	2	283	3	5	2
8.	3	252	1	2	2
9.	4	204	4	3	2
10.	5	271	5	4	2
11.	1	279	3	2	3
12.	2	245	2	3	3
13.	3	280	4	1	3
14.	4	227	5	4	3
15.	5	266	1	5	3
16.	1	287	4	3	4
17.	2	280	1	2	4
18.	3	246	5	4	4
19.	4	193	2	5	4
20.	5	334	3	1	4
21.	1	202	5	4	5
22.	2	260	4	1	5
23.	3	250	3	5	5
24.	4	259	1	2	5
25.	5	338	2	3	5

```
. anova outcome sequence period treat
```

Number of obs = 25R-squared = 0.6536

Root MSE = 32.4901Adj R-squared = 0.3073

Source	Partial SS	df	MS	F	Prob > F
Model	23904.08	12	1992.00667	1.89	0.1426
sequence	13601.36	4	3400.34	3.22	0.0516
period	6146.16	4	1536.54	1.46	0.2758
treat	4156.56	4	1039.14	0.98	0.4523
Residual	12667.28	12	1055.60667		
Total	36571.36	24	1523.80667		

These methods will work with any type of Latin-square design, including those with replicated measurements. For more information, see [\[R\] pk](#), [\[R\] pkcross](#), and [\[R\] pkshape](#).



Repeated-measures ANOVA

One approach for analyzing repeated-measures data is to use multivariate ANOVA (MANOVA); see [MV] [manova](#). In this approach, the data are placed in wide form (see [D] [reshape](#)), and the repeated measures enter the MANOVA as dependent variables.

A second approach for analyzing repeated measures is to use `anova`. However, one of the underlying assumptions for the F tests in ANOVA is independence of observations. In a repeated-measures design, this assumption is almost certainly violated or is at least suspect. In a repeated-measures ANOVA, the subjects (or whatever the experimental units are called) are observed for each level of one or more of the other categorical variables in the model. These variables are called the repeated-measure variables. Observations from the same subject are likely to be correlated.

The approach used in repeated-measures ANOVA to correct for this lack of independence is to apply a correction to the degrees of freedom of the F test for terms in the model that involve repeated measures. This correction factor, ϵ , lies between the reciprocal of the degrees of freedom for the repeated term and 1. [Box \(1954\)](#) provided the pioneering work in this area. [Milliken and Johnson \(1984\)](#) refer to the lower bound of this correction factor as Box's conservative correction factor. [Winer, Brown, and Michels \(1991\)](#) call it simply the conservative correction factor.

[Geisser and Greenhouse \(1958\)](#) provide an estimate for the correction factor called the Greenhouse–Geisser ϵ . This value is estimated from the data. [Huynh and Feldt \(1976\)](#) show that the Greenhouse–Geisser ϵ tends to be conservatively biased. They provide a revised correction factor called the Huynh–Feldt ϵ . When the Huynh–Feldt ϵ exceeds 1, it is set to 1. Thus there is a natural ordering for these correction factors:

$$\text{Box's conservative } \epsilon \leq \text{Greenhouse–Geisser } \epsilon \leq \text{Huynh–Feldt } \epsilon \leq 1$$

A correction factor of 1 is the same as no correction.

`anova` with the `repeated()` option computes these correction factors and displays the revised test results in a table that follows the standard ANOVA table. In the resulting table, H-F stands for Huynh–Feldt, G-G stands for Greenhouse–Geisser, and Box stands for Box's conservative ϵ .

► Example 15

This example is taken from table 4.3 of [Winer, Brown, and Michels \(1991\)](#). The reaction time for five subjects each tested with four drugs was recorded in the variable `score`. Here is a table of the data (see [P] [tabdisp](#) if you are unfamiliar with `tabdisp`):

```
. use http://www.stata-press.com/data/r11/t43, clear
(T4.3 -- Winer, Brown, Michels)
. tabdisp person drug, cellvar(score)
```

person	drug			
	1	2	3	4
1	30	28	16	34
2	14	18	10	22
3	24	20	18	30
4	38	34	20	44
5	26	28	14	30

`drug` is the repeated variable in this simple repeated-measures ANOVA example. The ANOVA is specified as follows:

► Example 16

Table 7.7 of [Winer, Brown, and Michels \(1991\)](#) provides another repeated-measures ANOVA example. There are four dial shapes and two methods for calibrating dials. Subjects are nested within calibration method, and an accuracy score is obtained. The data are shown below.

```
. use http://www.stata-press.com/data/r11/t77
(T7.7 -- Winer, Brown, Michels)
. tabdisp shape subject calib, cell(score)
```

4 dial shapes	2 methods for calibrating dials and subject nested in calib					
	1			2		
	1	2	3	1	2	3
1	0	3	4	4	5	7
2	0	1	3	2	4	5
3	5	5	6	7	6	8
4	3	4	2	8	6	9

The calibration method and dial shapes are fixed factors, whereas subjects are random. The appropriate test for calibration method uses the nested subject term as the error term. Both the dial shape and the interaction between dial shape and calibration method are tested with the dial shape by subject interaction nested within calibration method. Here we drop this term from the `anova` command, and it becomes residual error. The dial shape is the repeated variable because each subject is tested with all four dial shapes. Here is the `anova` command that produces the desired results:

```
. anova score calib / subject|calib shape calib#shape, repeated(shape)
```

	Number of obs = 24		R-squared = 0.8925		
	Root MSE = 1.11181		Adj R-squared = 0.7939		
Source	Partial SS	df	MS	F	Prob > F
Model	123.125	11	11.1931818	9.06	0.0003
calib	51.0416667	1	51.0416667	11.89	0.0261
subject calib	17.1666667	4	4.29166667		
shape	47.4583333	3	15.8194444	12.80	0.0005
calib#shape	7.45833333	3	2.48611111	2.01	0.1662
Residual	14.8333333	12	1.23611111		
Total	137.958333	23	5.99818841		

```
Between-subjects error term: subject|calib
Levels: 6 (4 df)
Lowest b.s.e. variable: subject
Covariance pooled over: calib (for repeated variable)
Repeated variable: shape
```

```
Huynh-Feldt epsilon = 0.8483
Greenhouse-Geisser epsilon = 0.4751
Box's conservative epsilon = 0.3333
```

Source	df	F	Prob > F			
			Regular	H-F	G-G	Box
shape	3	12.80	0.0005	0.0011	0.0099	0.0232
calib#shape	3	2.01	0.1662	0.1791	0.2152	0.2291
Residual	12					

The repeated-measure ϵ corrections are applied to any terms that are tested in the main ANOVA table and have the repeated variable in the term. These ϵ corrections are given in a table below the main ANOVA table. Here the repeated-measures tests for `shape` and `calib#shape` are presented.

Calibration method is significant, as is dial shape. The interaction between calibration method and dial shape is not significant. The repeated-measure ϵ corrections do not change these conclusions, but they do change the significance level for the tests on `shape` and `calib#shape`. Here, though, unlike in the previous example, the Huynh–Feldt ϵ is less than 1.

Here are the predictive marginal mean scores for calibration method and dial shapes. Because the interaction was not significant, we request only the `calib` and `shape` predictive margins.

```
. margins, within(calib)
Predictive margins                                Number of obs   =           24
Expression   : Linear prediction, predict()
within       : calib
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
calib						
1	3	.3209506	9.35	0.000	2.370948	3.629052
2	5.916667	.3209506	18.43	0.000	5.287615	6.545718

```
. margins, within(shape)
Predictive margins                                Number of obs   =           24
Expression   : Linear prediction, predict()
within       : shape
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
shape						
1	3.833333	.4538926	8.45	0.000	2.94372	4.722947
2	2.5	.4538926	5.51	0.000	1.610387	3.389613
3	6.166667	.4538926	13.59	0.000	5.277053	7.05628
4	5.333333	.4538926	11.75	0.000	4.44372	6.222947



□ Technical note

The computation of the Greenhouse–Geisser and Huynh–Feldt epsilons in a repeated-measures ANOVA requires the number of levels and degrees of freedom for the between-subjects error term, as well as a value computed from a pooled covariance matrix. The observations are grouped based on all but the lowest-level variable in the between-subjects error term. The covariance over the repeated variables is computed for each resulting group, and then these covariance matrices are pooled. The dimension of the pooled covariance matrix is the number of levels of the repeated variable (or combination of levels for multiple repeated variables). In [example 16](#), there are four levels of the repeated variable (`shape`), so the resulting covariance matrix is 4×4 .

The `anova` command automatically attempts to determine the between-subjects error term and the lowest-level variable in the between-subjects error term to group the observations for computation of the pooled covariance matrix. `anova` issues an error message indicating that the `bse()` or `bseunit()` option is required when `anova` cannot determine them. You may override the default selections of `anova` by specifying the `bse()`, `bseunit()`, or `grouping()` option. The term specified in the `bse()` option must be a term in the ANOVA model.

The default selection for the between-subjects error term (the `bse()` option) is the interaction of the nonrepeated categorical variables in the ANOVA model. The first variable listed in the between-subjects error term is automatically selected as the lowest-level variable in the between-subjects error term but can be overridden with the `bseunit(varname)` option. *varname* is often a term, such as `subject` or `subsample` within `subject`, and is most often listed first in the term because of the nesting notation of ANOVA. This term makes sense in most repeated-measures ANOVA designs when the terms of the model are written in standard form. For instance, in [example 16](#), there were three categorical variables (`subject`, `calib`, and `shape`), with `shape` being the repeated variable. Here `anova` looked for a term involving only `subject` and `calib` to determine the between-subjects error term. It found `subject|calib` as the term with six levels and 4 degrees of freedom. `anova` then picked `subject` as the default for the `bseunit()` option (the lowest variable in the between-subjects error term) because it was listed first in the term.

The grouping of observations proceeds, based on the different combinations of values of the variables in the between-subjects error term, excluding the lowest level variable (as found by default or as specified with the `bseunit()` option). You may specify the `grouping()` option to change the default grouping used in computing the pooled covariance matrix.

The between-subjects error term, number of levels, degrees of freedom, lowest variable in the term, and grouping information are presented after the main ANOVA table and before the rest of the repeated-measures output.

□

► Example 17

Data with two repeated variables are given in table 7.13 of [Winer, Brown, and Michels \(1991\)](#). The accuracy scores of subjects making adjustments to three dials during three different periods are recorded. Three subjects are exposed to a certain noise background level, whereas a different set of three subjects is exposed to a different noise background level. Here is a table of accuracy scores for the `noise`, `subject`, `period`, and `dial` variables:

```
. use http://www.stata-press.com/data/r11/t713
(T7.13 -- Winer, Brown, Michels)
. tabdisp subject dial period, by(noise) cell(score) stubwidth(11)
```

noise background and subject nested in noise	10 minute time periods and dial								
	1			2			3		
	1	2	3	1	2	3	1	2	3
1									
1	45	53	60	40	52	57	28	37	46
2	35	41	50	30	37	47	25	32	41
3	60	65	75	58	54	70	40	47	50
2									
1	50	48	61	25	34	51	16	23	35
2	42	45	55	30	37	43	22	27	37
3	56	60	77	40	39	57	31	29	46

`noise`, `period`, and `dial` are fixed, whereas `subject` is random. Both `period` and `dial` are repeated variables. The ANOVA for this example is specified next.

```
. anova score noise / subject|noise period noise#period
> / period#subject|noise dial noise#dial / dial#subject|noise
> period#dial noise#period#dial, repeated(period dial)
```

	Number of obs =	54	R-squared	=	0.9872
	Root MSE	= 2.81859	Adj R-squared	=	0.9576
Source	Partial SS	df	MS	F	Prob > F
Model	9797.72222	37	264.803303	33.33	0.0000
noise	468.166667	1	468.166667	0.75	0.4348
subject noise	2491.11111	4	622.777778		
period	3722.33333	2	1861.16667	63.39	0.0000
noise#period	333	2	166.5	5.67	0.0293
period#subject noise	234.888889	8	29.3611111		
dial	2370.33333	2	1185.16667	89.82	0.0000
noise#dial	50.3333333	2	25.1666667	1.91	0.2102
dial#subject noise	105.555556	8	13.1944444		
period#dial	10.6666667	4	2.66666667	0.34	0.8499
noise#period#dial	11.3333333	4	2.83333333	0.36	0.8357
Residual	127.111111	16	7.94444444		
Total	9924.83333	53	187.261006		

Between-subjects error term: subject|noise
Levels: 6 (4 df)
Lowest b.s.e. variable: subject
Covariance pooled over: noise (for repeated variables)
Repeated variable: period

Huynh-Feldt epsilon = 1.0668
*Huynh-Feldt epsilon reset to 1.0000
Greenhouse-Geisser epsilon = 0.6476
Box's conservative epsilon = 0.5000

Source	df	F	Prob > F			
			Regular	H-F	G-G	Box
period	2	63.39	0.0000	0.0000	0.0003	0.0013
noise#period	2	5.67	0.0293	0.0293	0.0569	0.0759
period#subject noise	8					

Repeated variable: dial

Huynh-Feldt epsilon = 2.0788
*Huynh-Feldt epsilon reset to 1.0000
Greenhouse-Geisser epsilon = 0.9171
Box's conservative epsilon = 0.5000

Source	df	F	Prob > F			
			Regular	H-F	G-G	Box
dial	2	89.82	0.0000	0.0000	0.0000	0.0007
noise#dial	2	1.91	0.2102	0.2102	0.2152	0.2394
dial#subject noise	8					

Repeated variables: period#dial

Huynh-Feldt epsilon = 1.3258
 *Huynh-Feldt epsilon reset to 1.0000
 Greenhouse-Geisser epsilon = 0.5134
 Box's conservative epsilon = 0.2500

Source	df	F	Prob > F			
			Regular	H-F	G-G	Box
period#dial	4	0.34	0.8499	0.8499	0.7295	0.5934
noise#period#dial	4	0.36	0.8357	0.8357	0.7156	0.5825
Residual	16					

For each repeated variable and for each combination of interactions of repeated variables, there are different ϵ correction values. The `anova` command produces tables for each applicable combination.

The two most significant factors in this model appear to be `dial` and `period`. The noise by `period` interaction may also be significant, depending on the correction factor you use. Below is a table of predictive margins for the accuracy score for `dial`, `period`, and noise by `period`.

```
. margins, within(dial)
```

Predictive margins Number of obs = 54
 Expression : Linear prediction, predict()
 within : dial

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
dial						
1	37.38889	.6643478	56.28	0.000	36.08679	38.69099
2	42.22222	.6643478	63.55	0.000	40.92012	43.52432
3	53.22222	.6643478	80.11	0.000	51.92012	54.52432

```
. margins, within(period)
```

Predictive margins Number of obs = 54
 Expression : Linear prediction, predict()
 within : period

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
period						
1	54.33333	.6643478	81.78	0.000	53.03124	55.63543
2	44.5	.6643478	66.98	0.000	43.1979	45.8021
3	34	.6643478	51.18	0.000	32.6979	35.3021

(Continued on next page)

```
. margins, within(noise period)
Predictive margins                                Number of obs   =           54
Expression   : Linear prediction, predict()
within       : noise period
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
noise#period						
1 1	53.77778	.9395297	57.24	0.000	51.93633	55.61922
1 2	49.44444	.9395297	52.63	0.000	47.603	51.28589
1 3	38.44444	.9395297	40.92	0.000	36.603	40.28589
2 1	54.88889	.9395297	58.42	0.000	53.04744	56.73033
2 2	39.55556	.9395297	42.10	0.000	37.71411	41.397
2 3	29.55556	.9395297	31.46	0.000	27.71411	31.397

Dial shape 3 produces the highest score, and scores decrease over the periods.

Example 17 had two repeated-measurement variables. Up to four repeated-measurement variables may be specified in the anova command.

Saved results

anova saves the following in e():

Scalars

- e(N) number of observations
- e(mss) model sum of squares
- e(df_m) model degrees of freedom
- e(rss) residual sum of squares
- e(df_r) residual degrees of freedom
- e(r2) *R*-squared
- e(r2_a) adjusted *R*-squared
- e(F) *F* statistic
- e(rmse) root mean squared error
- e(ll) log likelihood
- e(ll_0) log likelihood, constant-only model
- e(ss_#) sum of squares for term #
- e(df_#) numerator degrees of freedom for term #
- e(ssdenom_#) denominator sum of squares for term # (when using nonresidual error)
- e(dfdenom_#) denominator degrees of freedom for term # (when using nonresidual error)
- e(F_#) *F* statistic for term # (if computed)
- e(N_bse) number of levels of the between-subjects error term
- e(df_bse) degrees of freedom for the between-subjects error term
- e(box#) Box's conservative epsilon for a particular combination of repeated variables
(repeated() only)
- e(gg#) Greenhouse–Geisser epsilon for a particular combination of repeated variables
(repeated() only)
- e(hf#) Huynh–Feldt epsilon for a particular combination of repeated variables
(repeated() only)
- e(version) 2
- e(rank) rank of e(V)

Macros

<code>e(cmd)</code>	<code>anova</code>
<code>e(cmdline)</code>	command as typed
<code>e(depvar)</code>	name of dependent variable
<code>e(varnames)</code>	names of the right-hand-side variables
<code>e(term_#)</code>	term #
<code>e(errorterm_#)</code>	error term for term # (when using nonresidual error)
<code>e(sstype)</code>	type of sum of squares; <code>sequential</code> or <code>partial</code>
<code>e(repvars)</code>	names of repeated variables (<code>repeated()</code> only)
<code>e(repvar#)</code>	names of repeated variables for a particular combination (<code>repeated()</code> only)
<code>e(model)</code>	<code>ols</code>
<code>e(wtype)</code>	weight type
<code>e(wexp)</code>	weight expression
<code>e(properties)</code>	<code>b V</code>
<code>e(estat_cmd)</code>	program used to implement <code>estat</code>
<code>e(predict)</code>	program used to implement <code>predict</code>
<code>e(asbalanced)</code>	factor variables <code>fvset</code> as <code>asbalanced</code>
<code>e(asobserved)</code>	factor variables <code>fvset</code> as <code>asobserved</code>

Matrices

<code>e(b)</code>	coefficient vector
<code>e(V)</code>	variance–covariance matrix of the estimators
<code>e(Srep)</code>	covariance matrix based on repeated measures (<code>repeated()</code> only)

Functions

<code>e(sample)</code>	marks estimation sample

Methods and formulas

`anova` is implemented as an ado-file.

References

- Accock, A. C. 2008. *A Gentle Introduction to Stata*. 2nd ed. College Station, TX: Stata Press.
- Afifi, A. A., and S. P. Azen. 1979. *Statistical Analysis: A Computer Oriented Approach*. 2nd ed. New York: Academic Press.
- Altman, D. G. 1991. *Practical Statistics for Medical Research*. London: Chapman & Hall/CRC.
- Anderson, R. L. 1990. Gertrude Mary Cox 1900–1978. *Biographical Memoirs, National Academy of Sciences* 59: 116–132.
- Box, G. E. P. 1954. Some theorems on quadratic forms applied in the study of analysis of variance problems, I. Effect of inequality of variance in the one-way classification. *Annals of Mathematical Statistics* 25: 290–302.
- Box, J. F. 1978. R. A. Fisher: *The Life of a Scientist*. New York: Wiley.
- Cobb, G. W. 1998. *Introduction to Design and Analysis of Experiments*. New York: Springer.
- Edwards, A. L. 1985. *Multiple Regression and the Analysis of Variance and Covariance*. 2nd ed. New York: Freeman.
- Fisher, R. A. 1925. *Statistical Methods for Research Workers*. Edinburgh: Oliver & Boyd.
- . 1935. *The Design of Experiments*. Edinburgh: Oliver & Boyd.
- . 1990. *Statistical Methods, Experimental Design, and Scientific Inference*. Oxford: Oxford University Press.

- Geisser, S., and S. W. Greenhouse. 1958. An extension of Box's results on the use of the F distribution in multivariate analysis. *Annals of Mathematical Statistics* 29: 885–891.
- Gleason, J. R. 1999. [sg103: Within subjects \(repeated measures\) ANOVA, including between subjects factors](#). *Stata Technical Bulletin* 47: 40–45. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 236–243. College Station, TX: Stata Press.
- . 2000. [sg132: Analysis of variance from summary statistics](#). *Stata Technical Bulletin* 54: 42–46. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 328–332. College Station, TX: Stata Press.
- Higgins, J. E., and G. G. Koch. 1977. Variable selection and generalized chi-square analysis of categorical data applied to a large cross-sectional occupational health survey. *International Statistical Review* 45: 51–62.
- Huynh, H. 1978. Some approximate tests for repeated measurement designs. *Psychometrika* 43: 161–175.
- Huynh, H., and L. S. Feldt. 1976. Estimation of the Box correction for degrees of freedom from sample data in randomized block and split-plot designs. *Journal of Educational Statistics* 1: 69–82.
- Kennedy Jr., W. J., and J. E. Gentle. 1980. *Statistical Computing*. New York: Dekker.
- Kuehl, R. O. 2000. *Design of Experiments: Statistical Principles of Research Design and Analysis*. 2nd ed. Belmont, CA: Duxbury.
- Marchenko, Y. 2006. [Estimating variance components in Stata](#). *Stata Journal* 6: 1–21.
- Milliken, G. A., and D. E. Johnson. 1984. *Analysis of Messy Data, Volume 1: Designed Experiments*. New York: Nostrand Reinhold.
- Rabe-Hesketh, S., and B. S. Everitt. 2007. *A Handbook of Statistical Analyses Using Stata*. 4th ed. Boca Raton, FL: Chapman & Hall/CRC.
- Scheffé, H. 1959. *The Analysis of Variance*. New York: Wiley.
- Snedecor, G. W., and W. G. Cochran. 1989. *Statistical Methods*. 8th ed. Ames, IA: Iowa State University Press.
- van Belle, G., L. D. Fisher, P. J. Heagerty, and T. S. Lumley. 2004. *Biostatistics: A Methodology for the Health Sciences*. 2nd ed. New York: Wiley.
- Winer, B. J., D. R. Brown, and K. M. Michels. 1991. *Statistical Principles in Experimental Design*. 3rd ed. New York: McGraw–Hill.

Also see

- [R] [anova postestimation](#) — Postestimation tools for anova
- [R] [loneway](#) — Large one-way ANOVA, random effects, and reliability
- [R] [oneway](#) — One-way analysis of variance
- [R] [regress](#) — Linear regression
- [MV] [manova](#) — Multivariate analysis of variance and covariance

Description

The following postestimation commands are of special interest after `anova`:

command	description
<code>dfbeta</code>	DFBETA influence statistics
<code>estat hettest</code>	tests for heteroskedasticity
<code>estat imtest</code>	information matrix test
<code>estat ovtest</code>	Ramsey regression specification-error test for omitted variables
<code>estat szroeter</code>	Szroeter’s rank test for heteroskedasticity
<code>estat vif</code>	variance inflation factors for the independent variables
<code>acprplot</code>	augmented component-plus-residual plot
<code>avplot</code>	added-variable plot
<code>avplots</code>	all added-variable plots in one image
<code>cprplot</code>	component-plus-residual plot
<code>lvr2plot</code>	leverage-versus-squared-residual plot
<code>rvfplot</code>	residual-versus-fitted plot
<code>rvpplot</code>	residual-versus-predictor plot

For information about these commands, see [\[R\] regress postestimation](#).

The following standard postestimation commands are also available:

command	description
<code>estat</code>	AIC, BIC, VCE, and estimation sample summary
<code>estimates</code>	cataloging estimation results
<code>hausman</code>	Hausman’s specification test
<code>lincom</code>	point estimates, standard errors, testing, and inference for linear combinations of coefficients
<code>linktest</code>	link test for model specification
<code>lrtest</code>	likelihood-ratio test
<code>margins</code>	marginal means, predictive margins, marginal effects, and average marginal effects
<code>nlcom</code>	point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients
<code>predict</code>	predictions, residuals, influence statistics, and other diagnostic measures
<code>predictnl</code>	point estimates, standard errors, testing, and inference for generalized predictions
<code>suest</code>	seemingly unrelated estimation
<code>test</code>	Wald tests of simple and composite linear hypotheses
<code>testnl</code>	Wald tests of nonlinear hypotheses

See the corresponding entries in the *Base Reference Manual* for details.

Special-interest postestimation commands

In addition to the common `estat` commands (see [R] [estat](#)), `estat hettest`, `estat imtest`, `estat ovtest`, `estat szroeter`, and `estat vif` are also available. `dfbeta` is also available. The syntax for `dfbeta` and these `estat` commands is the same as after `regress`; see [R] [regress postestimation](#).

In addition to the standard syntax of `test` (see [R] [test](#)), `test` after `anova` has three additionally allowed syntaxes; see below. `test` performs Wald tests of expressions involving the coefficients of the underlying regression model. Simple and composite linear hypotheses are possible.

Syntax for predict

`predict` after `anova` follows the same syntax as `predict` after `regress` and can provide predictions, residuals, standardized residuals, studentized residuals, the standard error of the residuals, the standard error of the prediction, the diagonal elements of the projection (hat) matrix, and Cook's D . See [R] [regress postestimation](#) for details.

Syntax for test after anova

In addition to the standard syntax of `test` (see [R] [test](#)), `test` after `anova` also allows the following:

`test, test(matname) [mtest[ (opt) ] matv1c(matname) ]`

syntax a

`test, showorder`

syntax b

`test [ term [ term ... ] ] [ / term [ term ... ] ] [ , symbolic ]`

syntax c

- syntax a

test expression involving the coefficients of the underlying regression model; you provide information as a matrix
- syntax b

show underlying order of design matrix, which is useful when constructing *matname* argument of the `test()` option
- syntax c

test effects and show symbolic forms

Menu

Statistics > Linear models and related > ANOVA/MANOVA > Test linear hypotheses after anova

Options for test after anova

`test(matname)` is required with syntax a of `test`. The rows of *matname* specify linear combinations of the underlying design matrix of the ANOVA that are to be jointly tested. The columns correspond to the underlying design matrix (including the constant if it has not been suppressed). The column and row names of *matname* are ignored.

A listing of the constraints imposed by the `test()` option is presented before the table containing the tests. You should examine this table to verify that you have applied the linear combinations you desired. Typing `test, showorder` allows you to examine the ordering of the columns for the design matrix from the ANOVA.

`mtest` [*opt*] specifies that tests are performed for each condition separately. *opt* specifies the method for adjusting *p*-values for multiple testing. Valid values for *opt* are

<code>bonferroni</code>	Bonferroni's method
<code>holm</code>	Holm's method
<code>sidak</code>	Šidák's method
<code>noadjust</code>	no adjustment is to be made

Specifying `mtest` with no argument is equivalent to `mtest(noadjust)`.

`matv1c(matname)`, a programmer's option, saves the variance–covariance matrix of the linear combinations involved in the suite of tests. For the test $\mathbf{Lb} = \mathbf{c}$, what is returned in *matname* is $\mathbf{LV}\mathbf{L}'$, where *V* is the estimated variance–covariance matrix of **b**.

`showorder` causes `test` to list the definition of each column in the design matrix. `showorder` is not allowed with any other option.

`symbolic` requests the symbolic form of the test rather than the test statistic. When this option is specified with no terms (`test, symbolic`), the symbolic form of the estimable functions is displayed.

Remarks

Remarks are presented under the following headings:

Testing effects

Obtaining symbolic forms

Testing coefficients

Testing effects

After fitting a model using `anova`, you can test for the significance of effects in the ANOVA table, as well as for effects that are not reported in the ANOVA table, by using the `test` command. You follow `test` by the list of effects that you wish to test. By default, `test` uses the residual mean squared error in the denominator of the *F* ratio. You can specify other error terms by using the slash notation, just as you would with `anova`.

► Example 1

Recall our byssinosis example (example 8) in [R] `anova`:

(Continued on next page)

```
. anova prob workplace smokes race workplace#smokes workplace#race
> smokes#race workplace#smokes#race [aweight=pop]
(sum of wgt is 5.4190e+03)
```

	Number of obs =	65	R-squared =	0.8300	
	Root MSE =	.025902	Adj R-squared =	0.7948	
Source	Partial SS	df	MS	F	Prob > F
Model	.173646538	11	.015786049	23.53	0.0000
workplace	.097625175	2	.048812588	72.76	0.0000
smokes	.013030812	1	.013030812	19.42	0.0001
race	.001094723	1	.001094723	1.63	0.2070
workplace#smokes	.019690342	2	.009845171	14.67	0.0000
workplace#race	.001352516	2	.000676258	1.01	0.3718
smokes#race	.001662874	1	.001662874	2.48	0.1214
workplace#smokes#race	.000950841	2	.00047542	0.71	0.4969
Residual	.035557766	53	.000670901		
Total	.209204304	64	.003268817		

We can easily obtain a test on a particular term from the ANOVA table. Here are two examples:

```
. test smokes
```

Source	Partial SS	df	MS	F	Prob > F
smokes	.013030812	1	.013030812	19.42	0.0001
Residual	.035557766	53	.000670901		

```
. test smokes#race
```

Source	Partial SS	df	MS	F	Prob > F
smokes#race	.001662874	1	.001662874	2.48	0.1214
Residual	.035557766	53	.000670901		

Both of these tests use residual error by default and agree with the ANOVA table produced earlier.

Technical note

After `anova`, you can use the `/` syntax in `test` to perform tests with a variety of non- σ^2 I error structures. However, in most unbalanced models, the mean squares are not independent and do not have equal expectations under the null hypothesis. Also, be warned that you assume responsibility for the validity of the test statistic.

Example 2

We return to the nested ANOVA example (example 11) in [R] `anova`, where five brands of machinery were compared in an assembly line. We can obtain appropriate tests for the nested terms using `test`, even if we had run the `anova` command without initially indicating the proper error terms.

```
. use http://www.stata-press.com/data/r11/machine
(machine data)
```

```
. anova output machine operator|machine
```

	Number of obs =	57	R-squared =	0.8661	
	Root MSE =	1.47089	Adj R-squared =	0.8077	
Source	Partial SS	df	MS	F	Prob > F
Model	545.822288	17	32.1071934	14.84	0.0000
machine	430.980792	4	107.745198	49.80	0.0000
operator machine	101.353804	13	7.79644648	3.60	0.0009
Residual	84.3766582	39	2.16350406		
Total	630.198947	56	11.2535526		

In this ANOVA table, `machine` is tested with residual error. With this particular nested design, the appropriate error term for testing `machine` is `operator` nested within `machine`, which is easily obtained from `test`.

```
. test machine / operator|machine
```

Source	Partial SS	df	MS	F	Prob > F
machine	430.980792	4	107.745198	13.82	0.0001
operator machine	101.353804	13	7.79644648		

This result from `test` matches what we obtained from our `anova` command.

◀

► Example 3

The other nested ANOVA example ([example 12](#)) in [\[R\] anova](#) was based on the sewage data. The ANOVA table is presented here again. As before, we will use abbreviations of variable names in typing the commands.

```
. use http://www.stata-press.com/data/r11/sewage
(Sewage treatment)
. anova particulate s / m|s / f|m|s / w|f|m|s /, dropemptycells
```

	Number of obs =	64	R-squared =	0.6338	
	Root MSE =	12.7445	Adj R-squared =	0.5194	
Source	Partial SS	df	MS	F	Prob > F
Model	13493.6094	15	899.573958	5.54	0.0000
solution	7203.76563	1	7203.76563	17.19	0.0536
manager solution	838.28125	2	419.140625		
manager solution	838.28125	2	419.140625	0.55	0.6166
facility manager solution	3064.9375	4	766.234375		
facility manager solution	3064.9375	4	766.234375	2.57	0.1193
worker facility manager solution	2386.625	8	298.328125		
worker facility manager solution	2386.625	8	298.328125	1.84	0.0931
Residual	7796.25	48	162.421875		
Total	21289.8594	63	337.934276		

In practice, it is often beneficial to pool nonsignificant nested terms to increase the power of tests on remaining terms. One rule of thumb is to allow the pooling of a term whose p -value is larger than 0.25. In this sewage example, the p -value for the test of `manager` is 0.6166. This value indicates that the manager effect is negligible and might be ignored. Currently, `solution` is tested by `manager|solution`, which has only 2 degrees of freedom. If we pool the `manager` and `facility` terms and use this pooled estimate as the error term for `solution`, we would have a term with 6 degrees of freedom.

Below are two tests: a test of `solution` with the pooled `manager` and `facility` terms and a test of this pooled term by `worker`.

```
. test s / m|s f|m|s
```

Source	Partial SS	df	MS	F	Prob > F
solution	7203.76563	1	7203.76563	11.07	0.0159
manager solution					
facility manager solution	3903.21875	6	650.536458		

```
. test m|s f|m|s / w|f|m|s
```

Source	Partial SS	df	MS	F	Prob > F
manager solution					
facility manager solution	3903.21875	6	650.536458	2.18	0.1520
worker facility manager solution	2386.625	8	298.328125		

In the first test, we included two terms after the forward slash (`m|s` and `f|m|s`). `test` after `anova` allows multiple terms both before and after the slash. The terms before the slash are combined and are then tested by the combined terms that follow the slash (or residual error if no slash is present).

The p -value for `solution` using the pooled term is 0.0159. Originally, it was 0.0536. The increase in the power of the test is due to the increase in degrees of freedom for the pooled error term.

We can get identical results if we drop `manager` from the `anova` model. (This dataset has unique numbers for each facility so that there is no confusion of facilities when `manager` is dropped.)

```
. anova particulate s / f|s / w|f|s /, dropemptycells
```

Number of obs = 64

Root MSE = 12.7445

R-squared = 0.6338

Adj R-squared = 0.5194

Source	Partial SS	df	MS	F	Prob > F
Model	13493.6094	15	899.573958	5.54	0.0000
solution	7203.76562	1	7203.76562	11.07	0.0159
facility solution	3903.21875	6	650.536458		
facility solution	3903.21875	6	650.536458	2.18	0.1520
worker facility solution	2386.625	8	298.328125		
worker facility solution	2386.625	8	298.328125	1.84	0.0931
Residual	7796.25	48	162.421875		
Total	21289.8594	63	337.934276		

This output agrees with our earlier `test` results.

In the following example, two terms from the anova are jointly tested (pooled).

► Example 4

In [example 10](#) of [\[R\] anova](#), we fit the model `anova drate region c.mage region#c.mage`. Now we `test` for the overall significance of `region`.

```
. test region region#c.mage
```

Source	Partial SS	df	MS	F	Prob > F
region region#mage	1781.35344	6	296.89224	5.65	0.0002
Residual	2206.7263	42	52.5411023		

The overall F statistic associated with the `region` and `region#c.mage` terms is 5.65, and it is significant at the 0.02% level.

Typing `test` followed by one term in our model should produce output that exactly matches that provided by the `anova` command. In the ANOVA output, the `region` term, by itself, had a sum of squares of 1166.15, which, based on 3 degrees of freedom, yielded an F statistic of 7.40 and a significance level of 0.0004.

```
. test region
```

Source	Partial SS	df	MS	F	Prob > F
region	1166.14735	3	388.715783	7.40	0.0004
Residual	2206.7263	42	52.5411023		

`test` yields the same result.



Obtaining symbolic forms

`test` can also produce the symbolic form of the estimable functions and symbolic forms for particular tests.

► Example 5

After fitting an ANOVA model, we type `test`, `symbolic` to obtain the symbolic form of the estimable functions. For instance, returning to our blood pressure data introduced in [example 4](#) of [\[R\] anova](#), let's begin by reestimating `systolic` on `drug`, `disease`, and `drug#disease`:

(Continued on next page)

```
. use http://www.stata-press.com/data/r11/systolic, clear
(Systolic Blood Pressure Data)
. anova systolic drug##disease
```

	Number of obs =	58	R-squared =	0.4560	
	Root MSE =	10.5096	Adj R-squared =	0.3259	
Source	Partial SS	df	MS	F	Prob > F
Model	4259.33851	11	387.212591	3.51	0.0013
drug	2997.47186	3	999.157287	9.05	0.0001
disease	415.873046	2	207.936523	1.88	0.1637
drug#disease	707.266259	6	117.87771	1.07	0.3958
Residual	5080.81667	46	110.452536		
Total	9340.15517	57	163.862371		

To obtain the symbolic form of the estimable functions, type

```
. test, symbolic
drug
1  -(r2+r3+r4-r0)
2  r2
3  r3
4  r4
disease
1  -(r6+r7-r0)
2  r6
3  r7
drug#disease
1 1  -(r2+r3+r4+r6+r7-r12-r13-r15-r16-r18-r19-r0)
1 2  r6 - (r12+r15+r18)
1 3  r7 - (r13+r16+r19)
2 1  r2 - (r12+r13)
2 2  r12
2 3  r13
3 1  r3 - (r15+r16)
3 2  r15
3 3  r16
4 1  r4 - (r18+r19)
4 2  r18
4 3  r19
_cons      r0
```

► Example 6

To obtain the symbolic form for a particular test, we type `test term [term ...]`, `symbolic`. For instance, the symbolic form for the test of the main effect of `drug` is



```

. test drug, symbolic
drug
      1  -(r2+r3+r4)
      2   r2
      3   r3
      4   r4
disease
      1   0
      2   0
      3   0
drug#disease
      1  1  -1/3 (r2+r3+r4)
      1  2  -1/3 (r2+r3+r4)
      1  3  -1/3 (r2+r3+r4)
      2  1   1/3 r2
      2  2   1/3 r2
      2  3   1/3 r2
      3  1   1/3 r3
      3  2   1/3 r3
      3  3   1/3 r3
      4  1   1/3 r4
      4  2   1/3 r4
      4  3   1/3 r4
_cons      0

```

If we omit the symbolic option, we instead see the result of the test:

```

. test drug

```

Source	Partial SS	df	MS	F	Prob > F
drug	2997.47186	3	999.157287	9.05	0.0001
Residual	5080.81667	46	110.452536		

◀

Testing coefficients

The `test` command allows you to perform tests directly on the coefficients of the underlying regression model. For instance, the coefficient on the third drug and the second disease is referred to as `3.drug#2.disease`. This could also be written as `i3.drug#i2.disease`, or `_b[3.drug#2.disease]`, or even `_coef[i3.drug#i2.disease]`; see [U] 13.5 Accessing coefficients and standard errors.

► Example 7

Let's begin by testing whether the coefficient on the third drug is equal to the coefficient on the fourth in our blood pressure data. We have already fit the model `anova systolic drug##disease` (equivalent to `anova systolic drug disease drug#disease`), and you can see the results of that estimation in example 5. Even though we have performed many tasks since we fit the model, Stata still remembers, and we can perform tests at any time.

```

. test 3.drug = 4.drug
( 1) 3.drug - 4.drug = 0
      F( 1, 46) = 0.13
      Prob > F = 0.7234

```

We find that the two coefficients are not significantly different, at least at any significance level smaller than 73%. Let's now add the constraint that the coefficient on the third drug interacted with the third disease is equal to the coefficient on the fourth drug, again interacted with the third disease. We do that by typing the new constraint and adding the `accumulate` option:

```
. test 3.drug#3.disease = 4.drug#3.disease, accumulate
( 1) 3.drug - 4.drug = 0
( 2) 3.drug#3.disease - 4.drug#3.disease = 0
      F( 2, 46) = 0.39
      Prob > F = 0.6791
```

Let's continue. Our goal is to determine whether the third drug is significantly different from the fourth drug. So far, our test includes the equality of the two drug coefficients, along with the equality of the two drug coefficients when interacted with the third disease. We must add two more equations, one for each of the remaining two diseases.

```
. test 3.drug#2.disease = 4.drug#2.disease, accumulate
( 1) 3.drug - 4.drug = 0
( 2) 3.drug#3.disease - 4.drug#3.disease = 0
( 3) 3.drug#2.disease - 4.drug#2.disease = 0
      F( 3, 46) = 0.85
      Prob > F = 0.4761

. test 3.drug#1.disease = 4.drug#1.disease, accumulate
( 1) 3.drug - 4.drug = 0
( 2) 3.drug#3.disease - 4.drug#3.disease = 0
( 3) 3.drug#2.disease - 4.drug#2.disease = 0
( 4) 3o.drug#1b.disease - 4o.drug#1b.disease = 0
      Constraint 4 dropped
      F( 3, 46) = 0.85
      Prob > F = 0.4761
```

The overall F statistic is 0.85, which is hardly significant. We cannot reject the hypothesis that the third drug has the same effect as the fourth drug.

You may notice that we also got the message “Constraint 4 dropped”. For the technically inclined, this constraint was unnecessary, given the normalization of the model. You need not worry about such problems because Stata handles them automatically.

◀

The `test()` option of `test` provides a convenient alternative for testing coefficients. Instead of spelling out each coefficient involved in the test, a matrix representing the test provides the needed information. `test`, `showorder` shows the order of the terms in the ANOVA corresponding to the order of the columns for the matrix argument of `test()`.

► Example 8

We repeat the last test of [example 7](#) above with the `test()` option. First, we view the definition and order of the columns underlying the ANOVA performed on the systolic data.

```
. test, showorder

Order of columns in the design matrix
1: (drug==1)
2: (drug==2)
3: (drug==3)
4: (drug==4)
5: (disease==1)
6: (disease==2)
7: (disease==3)
8: (drug==1)*(disease==1)
9: (drug==1)*(disease==2)
10: (drug==1)*(disease==3)
11: (drug==2)*(disease==1)
12: (drug==2)*(disease==2)
13: (drug==2)*(disease==3)
14: (drug==3)*(disease==1)
15: (drug==3)*(disease==2)
16: (drug==3)*(disease==3)
17: (drug==4)*(disease==1)
18: (drug==4)*(disease==2)
19: (drug==4)*(disease==3)
20: _cons
```

Columns 1–4 correspond to the four levels of `drug`. Columns 5–7 correspond to the three levels of `disease`. Columns 8–19 correspond to the interaction of `drug` and `disease`. The last column corresponds to `_cons`, the constant in the model.

We construct the matrix `dr1vs2` with the same four constraints as the last test shown in [example 7](#) and then use the `test(dr1vs2)` option to perform the test.

```
. mat dr1vs2 = (0,0,1,-1, 0,0,0, 0,0,0,0,0,0,0,0, 0, 0, 0, 0 \
> 0,0,0, 0, 0,0,0, 0,0,0,0,0,0,0,0,1, 0, 0,-1, 0 \
> 0,0,0, 0, 0,0,0, 0,0,0,0,0,0,1,0, 0,-1, 0, 0 \
> 0,0,0, 0, 0,0,0, 0,0,0,0,0,0,1,0,0,-1, 0, 0, 0)

. test, test(dr1vs2)

( 1) 3.drug - 4.drug = 0
( 2) 3.drug#3.disease - 4.drug#3.disease = 0
( 3) 3.drug#2.disease - 4.drug#2.disease = 0
( 4) 3o.drug#1b.disease - 4o.drug#1b.disease = 0
    Constraint 4 dropped

F( 3, 46) = 0.85
Prob > F = 0.4761
```

Here the effort involved with spelling out the coefficients is similar to that of constructing a matrix and using it in the `test()` option. When the test involving coefficients is more complicated, many people prefer using the `test()` option.

◀

□ Technical note

You can use `test` to perform other, more complicated tests. In such cases, you will probably want to review the symbolic forms of particular tests, and you will certainly want to review the symbolic form of the estimable functions. We explained how to do that above.

Let's check that Stata gives the right answers by laboriously typing the gory details of the test for the main effect of `drug`. Stata already told us the symbolic form in the previous subsection. The obsessed among you have no doubt already worked through the algebra and established that Stata was

correct. Here we spell out each coefficient instead of using the `test()` option. Using the `test()` option would have saved us some typing. Our chances of typing all the constraints correctly, however, are so small that we typed them into a do-file:

```
. do mainef
. test 2.drug + (2.drug#1.disease + 2.drug#2.disease +
>          2.drug#3.disease - 1.drug#1.disease -
>          1.drug#2.disease - 1.drug#3.disease)/3 = 0, notest
(output omitted)
. test 3.drug + (3.drug#1.disease + 3.drug#2.disease +
>          3.drug#3.disease - 1.drug#1.disease -
>          1.drug#2.disease - 1.drug#3.disease)/3 = 0, accumulate notest
(output omitted)
. test 4.drug + (4.drug#1.disease + 4.drug#2.disease +
>          4.drug#3.disease - 1.drug#1.disease -
>          1.drug#2.disease - 1.drug#3.disease)/3 = 0, accumulate
( 1) 2.drug - .3333333*1b.drug#1b.disease - .3333333*1b.drug#2o.disease -
.3333333*1b.drug#3o.disease + .3333333*2o.drug#1b.disease +
.3333333*2.drug#2.disease + .3333333*2.drug#3.disease = 0
( 2) 3.drug - .3333333*1b.drug#1b.disease - .3333333*1b.drug#2o.disease -
.3333333*1b.drug#3o.disease + .3333333*3o.drug#1b.disease +
.3333333*3.drug#2.disease + .3333333*3.drug#3.disease = 0
( 3) 4.drug - .3333333*1b.drug#1b.disease - .3333333*1b.drug#2o.disease -
.3333333*1b.drug#3o.disease + .3333333*4o.drug#1b.disease +
.3333333*4.drug#2.disease + .3333333*4.drug#3.disease = 0
      F( 3, 46) =      9.05
      Prob > F =      0.0001
end of do-file
```

We have our result. The F statistic has 3 degrees of freedom and is 9.05. This is the same result we obtained when we typed `test drug`. However, typing `test drug` was much easier. □

Tests involving the underlying coefficients of an ANOVA model are often done to obtain various single-degree-of-freedom tests comparing different levels of a categorical variable. The `mttest` option presents each single-degree-of-freedom test in addition to the combined test.

► Example 9

[Rencher and Schaalje \(2008\)](#) illustrate single-degree-of-freedom contrasts for an ANOVA comparing the net weight of cans filled by five machines (labeled A–E). The data were originally obtained from [Ostle and Mensing \(1975\)](#). [Rencher and Schaalje](#) use a cell-means ANOVA model approach for this problem. We could do the same by using the `noconstant` option of `anova`; see [\[R\] anova](#). Instead, we obtain the same results by using the standard overparameterized ANOVA approach (i.e., we keep the constant in the model).

```
. use http://www.stata-press.com/data/r11/canfill
(Can Fill Data)
. list, sepby(machine)
```

	machine	weight
1.	A	11.95
2.	A	12.00
3.	A	12.25
4.	A	12.10
5.	B	12.18
6.	B	12.11
7.	C	12.16
8.	C	12.15
9.	C	12.08
10.	D	12.25
11.	D	12.30
12.	D	12.10
13.	E	12.10
14.	E	12.04
15.	E	12.02
16.	E	12.02

```
. anova weight machine
```

Number of obs = 16					
Root MSE = .087758					
R-squared = 0.4123					
Adj R-squared = 0.1986					
Source	Partial SS	df	MS	F	Prob > F
Model	.059426993	4	.014856748	1.93	0.1757
machine	.059426993	4	.014856748	1.93	0.1757
Residual	.084716701	11	.007701518		
Total	.144143694	15	.00960958		

```
. test, showorder
```

```
Order of columns in the design matrix
```

- ```
1: (machine==1)
2: (machine==2)
3: (machine==3)
4: (machine==4)
5: (machine==5)
6: _cons
```

The four single-degree-of-freedom tests of interest between the five machines are A and D versus B, C, and E; B and E versus C; A versus D; and B versus E. We place the orthogonal (unweighted) contrasts for these tests into the matrix C1 and use that with `test` to obtain the results.

*(Continued on next page)*

```
. mat C1 = (3,-2,-2,3,-2,0 \ 0,1,-2,0,1,0 \ 1,0,0,-1,0,0 \ 0,1,0,0,-1,0)
. test, test(C1) mtest
(1) 3*1b.machine - 2*2.machine - 2*3.machine + 3*4.machine - 2*5.machine = 0
(2) 2.machine - 2*3.machine + 5.machine = 0
(3) 1b.machine - 4.machine = 0
(4) 2.machine - 5.machine = 0
```

|     | F(df,11) | df | p        |
|-----|----------|----|----------|
| (1) | 0.75     | 1  | 0.4055 # |
| (2) | 0.31     | 1  | 0.5916 # |
| (3) | 4.47     | 1  | 0.0582 # |
| (4) | 1.73     | 1  | 0.2150 # |
| all | 1.93     | 4  | 0.1757   |

# unadjusted p-values

The `mtest` option causes `test` to produce the single-degree-of-freedom tests for each row of the matrix `C1` provided in the `test()` option.

The significance values above are not adjusted for multiple comparisons. `mtest` takes an optional argument for specifying multiple comparison adjustments. We could have produced the Bonferroni adjusted significance values by using the `mtest(bonferroni)` option.

```
. test, test(C1) mtest(bonferroni)
(1) 3*1b.machine - 2*2.machine - 2*3.machine + 3*4.machine - 2*5.machine = 0
(2) 2.machine - 2*3.machine + 5.machine = 0
(3) 1b.machine - 4.machine = 0
(4) 2.machine - 5.machine = 0
```

|     | F(df,11) | df | p        |
|-----|----------|----|----------|
| (1) | 0.75     | 1  | 1.0000 # |
| (2) | 0.31     | 1  | 1.0000 # |
| (3) | 4.47     | 1  | 0.2329 # |
| (4) | 1.73     | 1  | 0.8601 # |
| all | 1.93     | 4  | 0.1757   |

# Bonferroni adjusted p-values

➤ Example 10

Here there are two factors, A and B, each with three levels. The levels are evenly spaced so that linear and quadratic contrasts are of interest.

```
. use http://www.stata-press.com/data/r11/twowaytrend
```



```
. anova Y A B A#B
```

|          | Number of obs = | 36     | R-squared =     | 0.9304 |          |
|----------|-----------------|--------|-----------------|--------|----------|
|          | Root MSE =      | 2.6736 | Adj R-squared = | 0.9097 |          |
| Source   | Partial SS      | df     | MS              | F      | Prob > F |
| Model    | 2578.55556      | 8      | 322.319444      | 45.09  | 0.0000   |
| A        | 2026.72222      | 2      | 1013.36111      | 141.77 | 0.0000   |
| B        | 383.72222       | 2      | 191.86111       | 26.84  | 0.0000   |
| A#B      | 168.11111       | 4      | 42.0277778      | 5.88   | 0.0015   |
| Residual | 193             | 27     | 7.14814815      |        |          |
| Total    | 2771.55556      | 35     | 79.1873016      |        |          |

```
. test, showorder
```

```
Order of columns in the design matrix
```

- 1: (A==1)
- 2: (A==2)
- 3: (A==3)
- 4: (B==1)
- 5: (B==2)
- 6: (B==3)
- 7: (A==1)\*(B==1)
- 8: (A==1)\*(B==2)
- 9: (A==1)\*(B==3)
- 10: (A==2)\*(B==1)
- 11: (A==2)\*(B==2)
- 12: (A==2)\*(B==3)
- 13: (A==3)\*(B==1)
- 14: (A==3)\*(B==2)
- 15: (A==3)\*(B==3)
- 16: \_cons

We create the matrices `Alin`, `Aquad`, `Blin`, and `Bquad` corresponding to the linear and quadratic single-degree-of-freedom contrasts for terms `A` and `B`. These are combined into matrices `A` and `B`. Then the four interaction contrasts are formed from the elementwise multiplication of appropriate pairs of the single-degree-of-freedom contrasts. We use `foreach` and the `vecdiag()` matrix function to automate the process. For this particular problem, the interaction contrasts are combined into two matrices: `AxBlin` corresponding to the 2-degree-of-freedom test for factor `A` versus linear `B` and `AxBquad` corresponding to the 2-degree-of-freedom test for factor `A` versus quadratic `B`.

```
. mat Alin = -3, 0, 3, 0, 0, 0, -1, -1, -1, 0, 0, 0, 1, 1, 1, 0
. mat Aquad = 3, -6, 3, 0, 0, 0, 1, 1, 1, -2, -2, -2, 1, 1, 1, 0
. mat Blin = 0, 0, 0, -3, 0, 3, -1, 0, 1, -1, 0, 1, -1, 0, 1, 0
. mat Bquad = 0, 0, 0, 3, -6, 3, 1, -2, 1, 1, -2, 1, 1, -2, 1, 0
. mat A = Alin \ Aquad
. mat B = Blin \ Bquad
. foreach i in lin quad {
2. foreach j in lin quad {
3. mat A'i'B'j' = vecdiag(A'i'*B'j')
4. }
5. }
. mat AxBlin = AlinBlin \ AquadBlin
. mat AxBquad = AlinBquad \ AquadBquad
```

We use the `test()` and `mtest` options of `test` to obtain the single-degree-of-freedom tests for linear and quadratic `A` and `B`.

```
. mat list A, nonames
A[2,16]
 -3 0 3 0 0 0 -1 -1 -1 0 0 0 1 1 1 0
 3 -6 3 0 0 0 1 1 1 -2 -2 -2 1 1 1 0

. test, test(A) mtest
(1) - 3*1b.A + 3*3.A - 1b.A#1b.B - 1b.A#2o.B - 1b.A#3o.B + 3o.A#1b.B +
 3.A#2.B + 3.A#3.B = 0
(2) 3*1b.A - 6*2.A + 3*3.A + 1b.A#1b.B + 1b.A#2o.B + 1b.A#3o.B - 2*2o.A#1b.B
 - 2*2.A#2.B - 2*2.A#3.B + 3o.A#1b.B + 3.A#2.B + 3.A#3.B = 0
```

|     | F(df,27) | df | p        |
|-----|----------|----|----------|
| (1) | 212.65   | 1  | 0.0000 # |
| (2) | 70.88    | 1  | 0.0000 # |
| all | 141.77   | 2  | 0.0000   |

# unadjusted p-values

```
. mat list B, nonames
B[2,16]
 0 0 0 -3 0 3 -1 0 1 -1 0 1 -1 0 1 0
 0 0 0 3 -6 3 1 -2 1 1 -2 1 1 -2 1 0

. test, test(B) mtest
(1) - 3*1b.B + 3*3.B - 1b.A#1b.B + 1b.A#3o.B - 2o.A#1b.B + 2.A#3.B -
 3o.A#1b.B + 3.A#3.B = 0
(2) 3*1b.B - 6*2.B + 3*3.B + 1b.A#1b.B - 2*1b.A#2o.B + 1b.A#3o.B + 2o.A#1b.B
 - 2*2.A#2.B + 2.A#3.B + 3o.A#1b.B - 2*3.A#2.B + 3.A#3.B = 0
```

|     | F(df,27) | df | p        |
|-----|----------|----|----------|
| (1) | 26.17    | 1  | 0.0000 # |
| (2) | 27.51    | 1  | 0.0000 # |
| all | 26.84    | 2  | 0.0000   |

# unadjusted p-values

All the above tests appear to be significant. In addition to presenting the single-degree-of-freedom tests, the combined tests for A and B are produced and agree with the original ANOVA results.

Now we explore the interaction between A and B.

```
. mat list AxBlin, nonames
AxBlin[2,16]
 0 0 0 0 0 0 1 0 -1 0 0 0 -1 0 1 0
 0 0 0 0 0 0 -1 0 1 2 0 -2 -1 0 1 0

. test, test(AxBlin) mtest
(1) 1b.A#1b.B - 1b.A#3o.B - 3o.A#1b.B + 3.A#3.B = 0
(2) - 1b.A#1b.B + 1b.A#3o.B + 2*2o.A#1b.B - 2*2.A#3.B - 3o.A#1b.B + 3.A#3.B
 = 0
```

|     | F(df,27) | df | p        |
|-----|----------|----|----------|
| (1) | 17.71    | 1  | 0.0003 # |
| (2) | 0.07     | 1  | 0.7893 # |
| all | 8.89     | 2  | 0.0011   |

# unadjusted p-values

The 2-degree-of-freedom test of the interaction of A with the linear components of B is significant at the 0.0011 level. But, when we examine the two single-degree-of-freedom tests that compose this result, the significance is due to the linear A by linear B contrast (significance level of 0.0003). A significance value of 0.7893 for the quadratic A by linear B indicates that this factor is not significant for these data.

```
. mat list AxBquad, nonames
AxBquad[2,16]
 0 0 0 0 0 0 -1 2 -1 0 0 0 1 -2 1 0
 0 0 0 0 0 0 1 -2 1 -2 4 -2 1 -2 1 0
. test, test(AxBquad) mtest
(1) - 1b.A#1b.B + 2*1b.A#2o.B - 1b.A#3o.B + 3o.A#1b.B - 2*3.A#2.B + 3.A#3.B
 = 0
(2) 1b.A#1b.B - 2*1b.A#2o.B + 1b.A#3o.B - 2*2o.A#1b.B + 4*2.A#2.B -
 2*2.A#3.B + 3o.A#1b.B - 2*3.A#2.B + 3.A#3.B = 0
```

|     | F(df,27) | df | p        |
|-----|----------|----|----------|
| (1) | 2.80     | 1  | 0.1058 # |
| (2) | 2.94     | 1  | 0.0979 # |
| all | 2.87     | 2  | 0.0741   |

# unadjusted p-values

The test of A with the quadratic components of B does not fall below the 0.05 significance level.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## References

Ostle, B., and R. W. Mensing. 1975. *Statistics in Research*. 3rd ed. Ames, IA: Iowa State University Press.

Rencher, A. C., and G. B. Schaalje. 2008. *Linear Models in Statistics*. 2nd ed. New York: Wiley.

## Also see

- [R] [anova](#) — Analysis of variance and covariance
- [R] [regress postestimation](#) — Postestimation tools for regress
- [U] [20 Estimation and postestimation commands](#)

# Title

areg — Linear regression with a large dummy-variable set

## Syntax

areg *depvar* [*indepvars*] [*if*] [*in*] [*weight*] , absorb(*varname*) [*options*]

| <i>options</i>                     | description                                                                                                                |
|------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| Model                              |                                                                                                                            |
| * <u>absorb</u> ( <i>varname</i> ) | categorical variable to be absorbed                                                                                        |
| SE/Robust                          |                                                                                                                            |
| <u>vce</u> ( <i>vcetype</i> )      | <i>vcetype</i> may be <u>ols</u> , <u>robust</u> , <u>cluster</u> <i>clustvar</i> , <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting                          |                                                                                                                            |
| <u>level</u> (#)                   | set confidence level; default is <u>level</u> (95)                                                                         |
| <i>display_options</i>             | control spacing and display of omitted variables and base and empty cells                                                  |
| <sup>†</sup> <u>coeflegend</u>     | display coefficients' legend instead of coefficient table                                                                  |

\*absorb(*varname*) is required.

<sup>†</sup>coeflegend does not appear in the dialog box.

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

bootstrap, by, jackknife, mi estimate, rolling, and statsby are allowed; see [U] 11.1.10 Prefix commands.

vce(bootstrap) and vce(jackknife) are not allowed with the mi estimate prefix.

Weights are not allowed with the bootstrap prefix.

aweights are not allowed with the jackknife prefix.

aweights, fweights, and pweights are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

## Menu

Statistics > Linear models and related > Other > Linear regression absorbing one cat. variable

## Description

**areg** fits a linear regression absorbing one categorical factor. **areg** is designed for datasets with many groups, but not a number of groups that increases with the sample size. See the **xtreg**, **fe** command in [XT] **xtreg** for an estimator that handles the case in which the number of groups increases with the sample size.

## Options

### Model

`absorb(varname)` specifies the categorical variable, which is to be included in the regression as if it were specified by dummy variables. `absorb()` is required.

### SE/Robust

`vce(vctype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

`vce(ols)`, the default, uses the standard variance estimator for ordinary least-squares regression.

Exercise caution when using the `vce(cluster clustvar)` option with `areg`. The effective number of degrees of freedom for the robust variance estimator is  $n_g - 1$ , where  $n_g$  is the number of clusters. Thus the number of levels of the `absorb()` variable should not exceed the number of clusters.

### Reporting

`level(#)`; see [R] [estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

The following option is available with `areg` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

Suppose that you have a regression model that includes among the explanatory variables a large number,  $k$ , of mutually exclusive and exhaustive dummies:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{d}_1\gamma_1 + \mathbf{d}_2\gamma_2 + \cdots + \mathbf{d}_k\gamma_k + \boldsymbol{\epsilon}$$

For instance, the dummy variables,  $\mathbf{d}_i$ , might indicate countries in the world or states of the United States. One solution would be to fit the model with `regress`, but this solution is possible only if  $k$  is small enough so that the total number of variables (the number of columns of  $\mathbf{X}$  plus the number of  $\mathbf{d}_i$ 's plus one for  $\mathbf{y}$ ) is sufficiently small—meaning less than `matsize` (see [R] [matsize](#)). For problems with more variables than the largest possible value of `matsize` (100 for Small Stata, 800 for Stata/IC, and 11,000 for Stata/SE and Stata/MP), `regress` will not work. `areg` provides a way of obtaining estimates of  $\boldsymbol{\beta}$ —but not the  $\gamma_i$ 's—in these cases. The effects of the dummy variables are said to be absorbed.

### ► Example 1

So that we can compare the results produced by `areg` with Stata's other regression commands, we will fit a model in which  $k$  is small. `areg`'s real use, however, is when  $k$  is large.

In our automobile data, we have a variable called `rep78` that is coded 1, 2, 3, 4, and 5, where 1 means poor and 5 means excellent. Let's assume that we wish to fit a regression of `mpg` on `weight`, `gear_ratio`, and `rep78` (parameterized as a set of dummies).

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. regress mpg weight gear_ratio b5.rep78
```

|          |            |    |            |  |                 |        |
|----------|------------|----|------------|--|-----------------|--------|
| Source   | SS         | df | MS         |  | Number of obs = | 69     |
| Model    | 1575.97621 | 6  | 262.662702 |  | F( 6, 62) =     | 21.31  |
| Residual | 764.226686 | 62 | 12.3262369 |  | Prob > F =      | 0.0000 |
|          |            |    |            |  | R-squared =     | 0.6734 |
|          |            |    |            |  | Adj R-squared = | 0.6418 |
| Total    | 2340.2029  | 68 | 34.4147485 |  | Root MSE =      | 3.5109 |

|            |           |           |       |       |                      |          |
|------------|-----------|-----------|-------|-------|----------------------|----------|
| mpg        | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
| weight     | -.0051031 | .0009206  | -5.54 | 0.000 | -.0069433            | -.003263 |
| gear_ratio | .901478   | 1.565552  | 0.58  | 0.567 | -2.228015            | 4.030971 |
| rep78      |           |           |       |       |                      |          |
| 1          | -2.036937 | 2.740728  | -0.74 | 0.460 | -7.515574            | 3.4417   |
| 2          | -2.419822 | 1.764338  | -1.37 | 0.175 | -5.946682            | 1.107039 |
| 3          | -2.557432 | 1.370912  | -1.87 | 0.067 | -5.297846            | .1829814 |
| 4          | -2.788389 | 1.395259  | -2.00 | 0.050 | -5.577473            | .0006939 |
| _cons      | 36.23782  | 7.01057   | 5.17  | 0.000 | 22.22389             | 50.25175 |

To fit the `areg` equivalent, we type

```
. areg mpg weight gear_ratio, absorb(rep78)
Linear regression, absorbing indicators
```

|  |  |  |  |  |                 |        |
|--|--|--|--|--|-----------------|--------|
|  |  |  |  |  | Number of obs = | 69     |
|  |  |  |  |  | F( 2, 62) =     | 41.64  |
|  |  |  |  |  | Prob > F =      | 0.0000 |
|  |  |  |  |  | R-squared =     | 0.6734 |
|  |  |  |  |  | Adj R-squared = | 0.6418 |
|  |  |  |  |  | Root MSE =      | 3.5109 |

|            |            |           |       |       |                      |          |
|------------|------------|-----------|-------|-------|----------------------|----------|
| mpg        | Coef.      | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
| weight     | -.0051031  | .0009206  | -5.54 | 0.000 | -.0069433            | -.003263 |
| gear_ratio | .901478    | 1.565552  | 0.58  | 0.567 | -2.228015            | 4.030971 |
| _cons      | 34.05889   | 7.056383  | 4.83  | 0.000 | 19.95338             | 48.1644  |
| rep78      | F(4, 62) = |           | 1.117 | 0.356 | (5 categories)       |          |

Both `regress` and `areg` display the same  $R^2$  values, root mean squared error, and—for `weight` and `gear_ratio`—the same parameter estimates, standard errors,  $t$  statistics, significance levels, and confidence intervals. `areg`, however, does not report the coefficients for `rep78`, and, in fact, they are not even calculated. This computational trick makes the problem manageable when  $k$  is large. `areg` reports a test that the coefficients associated with `rep78` are jointly zero. Here this test has a significance level of 35.6%. This  $F$  test for `rep78` is the same that we would obtain after `regress` if we were to specify `test 1.rep78 2.rep78 3.rep78 4.rep78`; see [\[R\] test](#).

The model  $F$  tests reported by `regress` and `areg` also differ. The `regress` command reports a test that all coefficients except that of the constant are equal to zero; thus, the dummies are included in this test. The `areg` output shows a test that all coefficients excluding the dummies and the constant are equal to zero. This is the same test that can be obtained after `regress` by typing `test weight gear_ratio`.

## □ Technical note

`areg` is designed for datasets with many groups, but not a number that grows with the sample size. Consider two different samples from the U.S. population. In the first sample, we have 10,000 individuals and we want to include an indicator for each of the 50 states, whereas in the second sample we have 3 observations on each of 10,000 individuals and we want to include an indicator for each individual. `areg` was designed for datasets similar to the first sample in which we have a fixed number of groups, the 50 states. In the second sample, the number of groups, which is the number of individuals, grows as we include more individuals in the sample. For an estimator designed to handle the case in which the number of groups grows with the sample size, see the `xtreg`, `fe` command in [XT] [xtreg](#).

Although the point estimates produced by `areg` and `xtreg`, `fe` are the same, the estimated VCEs differ when `cluster()` is specified because the commands make different assumptions about whether the number of groups increases with the sample size.

□

## □ Technical note

The intercept reported by `areg` deserves some explanation because, given  $k$  mutually exclusive and exhaustive dummies, it is arbitrary. `areg` identifies the model by choosing the intercept that makes the prediction calculated at the means of the independent variables equal to the mean of the dependent variable:  $\bar{y} = \bar{x}\hat{\beta}$ .

```
. predict yhat
(option xb assumed; fitted values)
. summarize mpg yhat if rep78 != .
```

| Variable | Obs | Mean     | Std. Dev. | Min      | Max      |
|----------|-----|----------|-----------|----------|----------|
| mpg      | 69  | 21.28986 | 5.866408  | 12       | 41       |
| yhat     | 69  | 21.28986 | 4.383224  | 11.58643 | 28.07367 |

We had to include `if rep78 < .` in our `summarize` command because we have missing values in our data. `areg` automatically dropped those missing values (as it should) in forming the estimates, but `predict` with the `xb` option will make predictions for cases with missing `rep78` because it does not know that `rep78` is really part of our model.

These predicted values do not include the absorbed effects (i.e., the  $d_i\gamma_i$ ). For predicted values that include these effects, use the `xbd` option of `predict` (see [R] [areg postestimation](#)) or see [XT] [xtreg](#).

□

## ▷ Example 2

`areg, vce(robust)` is a Huberized version of `areg`; see [P] [\\_robust](#). Just as `areg` is equivalent to using `regress` with dummies, `areg, vce(robust)` is equivalent to using `regress, vce(robust)` with dummies. You can use `areg, vce(robust)` when you expect heteroskedastic or nonnormal errors. `areg, vce(robust)`, like ordinary regression, assumes that the observations are independent, unless the `vce(cluster clustvar)` option is specified. If the `vce(cluster clustvar)` option is specified, this independence assumption is relaxed and only the clusters identified by equal values of `clustvar` are assumed to be independent.

Assume that we were to collect data by randomly sampling 10,000 doctors (from 100 hospitals) and then sampling 10 patients of each doctor, yielding a total dataset of 100,000 patients in a cluster sample. If in some regression we wished to include effects of the hospitals to which the doctors belonged, we would want to include a dummy variable for each hospital, adding 100 variables to our model. `areg` could fit this model by

```
. areg depvar patient_vars, absorb(hospital) vce(cluster doctor)
```

◀

## Saved results

`areg` saves the following in `e()`:

### Scalars

|                          |                                                                                         |
|--------------------------|-----------------------------------------------------------------------------------------|
| <code>e(N)</code>        | number of observations                                                                  |
| <code>e(tss)</code>      | total sum of squares                                                                    |
| <code>e(df_m)</code>     | model degrees of freedom                                                                |
| <code>e(rss)</code>      | residual sum of squares                                                                 |
| <code>e(df_r)</code>     | residual degrees of freedom                                                             |
| <code>e(r2)</code>       | <i>R</i> -squared                                                                       |
| <code>e(r2_a)</code>     | adjusted <i>R</i> -squared                                                              |
| <code>e(df_a)</code>     | degrees of freedom for absorbed effect                                                  |
| <code>e(rmse)</code>     | root mean squared error                                                                 |
| <code>e(ll)</code>       | log likelihood                                                                          |
| <code>e(ll_0)</code>     | log likelihood, constant-only model                                                     |
| <code>e(N_clust)</code>  | number of clusters                                                                      |
| <code>e(F)</code>        | <i>F</i> statistic                                                                      |
| <code>e(F_absorb)</code> | <i>F</i> statistic for absorbed effect (when <code>vce(robust)</code> is not specified) |
| <code>e(rank)</code>     | rank of <code>e(V)</code>                                                               |

### Macros

|                                   |                                                                |
|-----------------------------------|----------------------------------------------------------------|
| <code>e(cmd)</code>               | <code>areg</code>                                              |
| <code>e(cmdline)</code>           | command as typed                                               |
| <code>e(depvar)</code>            | name of dependent variable                                     |
| <code>e(absvar)</code>            | name of <code>absorb</code> variable                           |
| <code>e(wtype)</code>             | weight type                                                    |
| <code>e(wexp)</code>              | weight expression                                              |
| <code>e(title)</code>             | title in estimation output                                     |
| <code>e(clustvar)</code>          | name of cluster variable                                       |
| <code>e(vce)</code>               | <i>vcetype</i> specified in <code>vce()</code>                 |
| <code>e(vcetype)</code>           | title used to label Std. Err.                                  |
| <code>e(datasignature)</code>     | the checksum                                                   |
| <code>e(datasignaturevars)</code> | variables used in calculation of checksum                      |
| <code>e(properties)</code>        | <code>b V</code>                                               |
| <code>e(predict)</code>           | program used to implement <code>predict</code>                 |
| <code>e(marginsnotok)</code>      | predictions disallowed by <code>margins</code>                 |
| <code>e(asbalanced)</code>        | factor variables <code>fvset</code> as <code>asbalanced</code> |
| <code>e(asobserved)</code>        | factor variables <code>fvset</code> as <code>asobserved</code> |



## Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`areg` is implemented as an ado-file.

`areg` begins by recalculating *depvar* and *indepvars* to have mean 0 within the groups specified by `absorb()`. The overall mean of each variable is then added back in. The adjusted *depvar* is then regressed on the adjusted *indepvars* with `regress`, yielding the coefficient estimates. The degrees of freedom of the variance–covariance matrix of the coefficients is then adjusted to account for the absorbed variables—this calculation yields the same results (up to numerical roundoff error) as if the matrix had been calculated directly by the formulas given in [R] [regress](#).

`areg` with `vce(robust)` or `vce(cluster clustvar)` works similarly, calling `_robust` after `regress` to produce the Huber/White/sandwich estimator of the variance or its clustered version. See [P] [\\_robust](#), in particular, in [Introduction](#) and [Methods and formulas](#). The model *F* test uses the robust variance estimates. There is, however, no simple computational means of obtaining a robust test of the absorbed dummies; thus this test is not displayed when the `vce(robust)` or `vce(cluster clustvar)` option is specified.

The number of groups specified in `absorb()` are included in the degrees of freedom used in the finite-sample adjustment of the cluster–robust VCE estimator. This statement is only valid if the number of groups is small relative to the sample size. (Technically, the number of groups must remain fixed as the sample size grows.) For an estimator that allows the number of groups to grow with the sample size, see the `xtreg`, `fe` command in [XT] [xtreg](#).

## Reference

Blackwell III, J. L. 2005. [Estimation and testing of fixed-effect panel-data systems](#). *Stata Journal* 5: 202–207.

## Also see

[R] [areg postestimation](#) — Postestimation tools for `areg`

[R] [regress](#) — Linear regression

[XT] [xtreg](#) — Fixed-, between-, and random-effects, and population-averaged linear models

[U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `areg`:

| command                | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>     | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estimates</code> | cataloging estimation results                                                                       |
| <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>  | link test for model specification                                                                   |
| <code>lrtest</code>    | likelihood-ratio test                                                                               |
| <code>margins</code>   | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>   | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code> | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic]
```

where  $y_j = \mathbf{x}_j\mathbf{b} + d_{\text{absorbvar}} + e_j$  and *statistic* is

| statistic                | description                                                 |
|--------------------------|-------------------------------------------------------------|
| Main                     |                                                             |
| <code>xb</code>          | $\mathbf{x}_j\mathbf{b}$ , fitted values; the default       |
| <code>stdp</code>        | standard error of the prediction                            |
| <code>dresiduals</code>  | $d_{\text{absorbvar}} + e_j = y_j - \mathbf{x}_j\mathbf{b}$ |
| * <code>xbd</code>       | $\mathbf{x}_j\mathbf{b} + d_{\text{absorbvar}}$             |
| * <code>d</code>         | $d_{\text{absorbvar}}$                                      |
| * <code>residuals</code> | residual                                                    |
| * <code>score</code>     | score; equivalent to <code>residuals</code>                 |

Unstarred statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample. Starred statistics are calculated only for the estimation sample, even when `if e(sample)` is not specified.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

**xb**, the default, calculates the prediction of  $\mathbf{x}_j\mathbf{b}$ , the fitted values, by using the average effect of the absorbed variable. Also see **xbd** below.

**stdp** calculates the standard error of  $\mathbf{x}_j\mathbf{b}$ .

**dresiduals** calculates  $y_j - \mathbf{x}_j\mathbf{b}$ , which are the residuals plus the effect of the absorbed variable.

**xbd** calculates  $\mathbf{x}_j\mathbf{b} + d_{\text{absorbvar}}$ , which are the fitted values including the individual effects of the absorbed variable.

**d** calculates  $d_{\text{absorbvar}}$ , the individual coefficients for the absorbed variable.

**residuals** calculates the residuals, that is,  $y_j - (\mathbf{x}_j\mathbf{b} + d_{\text{absorbvar}})$ .

**score** is a synonym for **residuals**.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [areg](#) — Linear regression with a large dummy-variable set

[U] [20 Estimation and postestimation commands](#)

Syntax

```
asclogit depvar [indepvars] [if] [in] [weight] , case(varname)
 alternatives(varname) [options]
```

| <i>options</i>                                        | description                                                                                                                |
|-------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| Model                                                 |                                                                                                                            |
| * <u>case</u> ( <i>varname</i> )                      | use <i>varname</i> to identify cases                                                                                       |
| * <u>alternatives</u> ( <i>varname</i> )              | use <i>varname</i> to identify the alternatives available for each case                                                    |
| <u>casevars</u> ( <i>varlist</i> )                    | case-specific variables                                                                                                    |
| <u>basealternative</u> (#   <i>lbl</i>   <i>str</i> ) | alternative used as base category                                                                                          |
| <u>noconstant</u>                                     | suppress alternative-specific constant terms                                                                               |
| <u>altwise</u>                                        | use alternative-wise deletion instead of casewise deletion                                                                 |
| <u>offset</u> ( <i>varname</i> )                      | include <i>varname</i> in model with coefficient constrained to 1                                                          |
| <u>constraints</u> ( <i>constraints</i> )             | apply specified linear constraints                                                                                         |
| <u>collinear</u>                                      | keep collinear variables                                                                                                   |
| SE/Robust                                             |                                                                                                                            |
| <u>vce</u> ( <i>vcetype</i> )                         | <i>vcetype</i> may be <u>oim</u> , <u>robust</u> , <u>cluster</u> <i>clustvar</i> , <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting                                             |                                                                                                                            |
| <u>level</u> (#)                                      | set confidence level; default is <u>level</u> (95)                                                                         |
| or                                                    | report odds ratios                                                                                                         |
| <u>noheader</u>                                       | do not display the header on the coefficient table                                                                         |
| <u>nocnsreport</u>                                    | do not display constraints                                                                                                 |
| Maximization                                          |                                                                                                                            |
| <u>maximize_options</u>                               | control the maximization process; seldom used                                                                              |
| † <u>coeflegend</u>                                   | display coefficients’ legend instead of coefficient table                                                                  |

\* case(*varname*) and alternatives(*varname*) are required.  
† coeflegend does not appear in the dialog box.  
bootstrap, by, jackknife, statsby, and xi are allowed; see [U] 11.1.10 Prefix commands.  
Weights are not allowed with the bootstrap prefix.  
fweights, iweights, and pweights are allowed (see [U] 11.1.6 weight), but they are interpreted to apply to cases as a whole, not to individual observations. See Use of weights in [R] clogit.  
See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Categorical outcomes > Alternative-specific conditional logit

## Description

`asclogit` fits McFadden's choice model, which is a specific case of the more general conditional logistic regression model (McFadden 1974). `asclogit` requires multiple observations for each case (individual or decision), where each observation represents an alternative that may be chosen. The cases are identified by the variable specified in the `case()` option, whereas the alternatives are identified by the variable specified in the `alternatives()` option. The outcome or chosen alternative is identified by a value of 1 in `depvar`, whereas zeros indicate the alternatives that were not chosen. There can be multiple alternatives chosen for each case.

`asclogit` allows two types of independent variables: alternative-specific variables and case-specific variables. Alternative-specific variables vary across both cases and alternatives and are specified in `indepvars`. Case-specific variables vary only across cases and are specified in the `casevars()` option.

See [R] `clogit` for a more general application of conditional logistic regression. For example, `clogit` would be used when you have grouped data where each observation in a group may be a different individual, but all individuals in a group have a common characteristic. You may use `clogit` to obtain the same estimates as `asclogit` by specifying the `case()` variable as the `group()` variable in `clogit` and generating variables that interact the `casevars()` in `asclogit` with each alternative (in the form of an indicator variable), excluding the interaction variable associated with the base alternative. `asclogit` takes care of this data-management burden for you. Also, for `clogit`, each record (row in your data) is an observation, whereas in `asclogit` each case, consisting of several records (the alternatives) in your data, is an observation. This last point is important because `asclogit` will drop observations, by default, in a casewise fashion. That is, if there is at least one missing value in any of the variables for each record of a case, the entire case is dropped from estimation. To use alternative-wise deletion, specify the `altwise` option and only the records with missing values will be dropped from estimation.

## Options

### Model

`case(varname)` specifies the numeric variable that identifies each case. `case()` is required and must be integer valued.

`alternatives(varname)` specifies the variable that identifies the alternatives for each case. The number of alternatives can vary with each case; the maximum number of alternatives cannot exceed the limits of `tabulate oneway`; see [R] `tabulate oneway`. `alternatives()` is required and may be a numeric or a string variable.

`casevars(varlist)` specifies the case-specific numeric variables. These are variables that are constant for each case. If there are a maximum of  $J$  alternatives, there will be  $J - 1$  sets of coefficients associated with the `casevars()`.

`basealternative(#|lbl|str)` specifies the alternative that normalizes the latent-variable location (the level of utility). The base alternative may be specified as a number, label, or string depending on the storage type of the variable indicating alternatives. The default is the alternative with the highest frequency.

If `vce(bootstrap)` or `vce(jackknife)` is specified, you must specify the base alternative. This is to ensure that the same model is fit with each call to `asclogit`.

`noconstant` suppresses the  $J - 1$  alternative-specific constant terms.

`altwise` specifies that alternative-wise deletion be used when marking out observations due to missing values in your variables. The default is to use casewise deletion; that is, the entire group

of observations making up a case is deleted if any missing values are encountered. This option does not apply to observations that are marked out by the `if` or `in` qualifier or the `by` prefix.

`offset(varname)`, `constraints(numlist | matname)`, `collinear`; see [R] [estimation options](#).

#### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

#### Reporting

`level(#)`; see [R] [estimation options](#).

`or` reports the estimated coefficients transformed to odds ratios, i.e.,  $e^b$  rather than  $b$ . Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated. `or` may be specified at estimation or when replaying previously estimated results.

`noheader` prevents the coefficient table header from being displayed.

`nocnsreport`; see [R] [estimation options](#).

#### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [R] [maximize](#). These options are seldom used.

`technique(bhhh)` is not allowed.

The initial estimates must be specified as `from(matname [ , copy ])`, where *matname* is the matrix containing the initial estimates and the `copy` option specifies that only the position of each element in *matname* is relevant. If `copy` is not specified, the column stripe of *matname* identifies the estimates.

The following option is available with `asclogit` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

`asclogit` fits McFadden's choice model (McFadden [1974]; for a brief introduction, see Greene [2008, sec. 23.11] or Cameron and Trivedi [2009, sec. 15.5]). In this model, we have a set of unordered alternatives indexed by  $1, 2, \dots, J$ . Let  $y_{ij}$ ,  $j = 1, \dots, J$ , be an indicator variable for the alternative actually chosen by the  $i$ th individual (case). That is,  $y_{ij} = 1$  if individual  $i$  chose alternative  $j$  and  $y_{ij} = 0$  otherwise. The independent variables come in two forms: alternative specific and case specific. Alternative-specific variables vary among the alternatives (as well as cases), and case-specific variables vary only among cases. Assume that we have  $p$  alternative-specific variables so that for case  $i$  we have a  $J \times p$  matrix,  $\mathbf{X}_i$ . Further, assume that we have  $q$  case-specific variables so that we have a  $1 \times q$  vector  $\mathbf{z}_i$  for case  $i$ . Our random-utility model can then be expressed as

$$\mathbf{u}_i = \mathbf{X}_i\beta + (\mathbf{z}_i\mathbf{A})' + \epsilon_i$$

Here  $\beta$  is a  $p \times 1$  vector of alternative-specific regression coefficients and  $\mathbf{A} = (\alpha_1, \dots, \alpha_J)$  is a  $q \times J$  matrix of case-specific regression coefficients. The elements of the  $J \times 1$  vector  $\epsilon_i$  are independent Type I (Gumbel-type) extreme-value random variables with mean  $\gamma$  (the Euler–Mascheroni constant, approximately 0.577) and variance  $\pi^2/6$ . We must fix one of the  $\alpha_j$  to the constant vector to normalize the location. We set  $\alpha_k = 0$ , where  $k$  is specified by the `basealternative()` option. The vector  $\mathbf{u}_i$  quantifies the utility that the individual gains from the  $J$  alternatives. The alternative chosen by individual  $i$  is the one that maximizes utility.

Daniel Little McFadden was born in 1937 in North Carolina. He studied physics, psychology, and economics at the University of Minnesota and has taught economics at Pittsburgh, Berkeley, and MIT. His contributions to logit models were triggered by a student's project on freeway routing decisions, and his work consistently links economic theory and applied problems. In 2000, he shared the Nobel Prize in Economics with James J. Heckman.

## ► Example 1

We have data on 295 consumers and their choice of automobile. Each consumer chose among an American, Japanese, or European car; the variable `car` indicates the nationality of the car for each alternative. We want to explore the relationship between the choice of `car` to the consumer's sex (variable `sex`) and income (variable `income` in thousands of dollars). We also have information on the number of dealerships of each nationality in the consumer's city in the variable `dealer` that we want to include as a regressor. We assume that consumers' preferences are influenced by the number of dealerships in an area but that the number of dealerships is not influenced by consumer preferences (which we admit is a rather strong assumption). The variable `dealer` is an alternative-specific variable ( $\mathbf{X}_i$  is a  $3 \times 1$  vector in our previous notation), and `sex` and `income` are case-specific variables ( $\mathbf{z}_i$  is a  $1 \times 2$  vector). Each consumer's chosen car is indicated by the variable `choice`.

Let's list some of the data.

```
. use http://www.stata-press.com/data/r11/choice
. list id car choice dealer sex income in 1/12, sepby(id)
```

|     | id | car      | choice | dealer | sex    | income |
|-----|----|----------|--------|--------|--------|--------|
| 1.  | 1  | American | 0      | 18     | male   | 46.7   |
| 2.  | 1  | Japan    | 0      | 8      | male   | 46.7   |
| 3.  | 1  | Europe   | 1      | 5      | male   | 46.7   |
| 4.  | 2  | American | 1      | 17     | male   | 26.1   |
| 5.  | 2  | Japan    | 0      | 6      | male   | 26.1   |
| 6.  | 2  | Europe   | 0      | 2      | male   | 26.1   |
| 7.  | 3  | American | 1      | 12     | male   | 32.7   |
| 8.  | 3  | Japan    | 0      | 6      | male   | 32.7   |
| 9.  | 3  | Europe   | 0      | 2      | male   | 32.7   |
| 10. | 4  | American | 0      | 18     | female | 49.2   |
| 11. | 4  | Japan    | 1      | 7      | female | 49.2   |
| 12. | 4  | Europe   | 0      | 4      | female | 49.2   |

We see, for example, that the first consumer, a male earning \$46,700 per year, chose to purchase a European car even though there are more American and Japanese car dealers in his area. The fourth consumer, a female earning \$49,200 per year, purchased a Japanese car.

We now fit our model.

```
. asclogit choice dealer, casevars(sex income) case(id) alternatives(car)

Iteration 0: log likelihood = -273.55685
Iteration 1: log likelihood = -252.75109
Iteration 2: log likelihood = -250.78555
Iteration 3: log likelihood = -250.7794
Iteration 4: log likelihood = -250.7794

Alternative-specific conditional logit
Case variable: id
Alternative variable: car

Number of obs = 885
Number of cases = 295
Alts per case: min = 3
 avg = 3.0
 max = 3

Wald chi2(5) = 15.86
Prob > chi2 = 0.0072

Log likelihood = -250.7794
```

| choice   | Coef.              | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|--------------------|-----------|-------|-------|----------------------|-----------|
| car      |                    |           |       |       |                      |           |
| dealer   | .0680938           | .0344465  | 1.98  | 0.048 | .00058               | .1356076  |
| American | (base alternative) |           |       |       |                      |           |
| Japan    |                    |           |       |       |                      |           |
| sex      | -.5346039          | .3141564  | -1.70 | 0.089 | -1.150339            | .0811314  |
| income   | .0325318           | .012824   | 2.54  | 0.011 | .0073973             | .0576663  |
| _cons    | -1.352189          | .6911829  | -1.96 | 0.050 | -2.706882            | .0025049  |
| Europe   |                    |           |       |       |                      |           |
| sex      | .5704109           | .4540247  | 1.26  | 0.209 | -.3194612            | 1.460283  |
| income   | .032042            | .0138676  | 2.31  | 0.021 | .004862              | .0592219  |
| _cons    | -2.355249          | .8526681  | -2.76 | 0.006 | -4.026448            | -.6840501 |

Displaying the results as odds ratios makes interpretation easier.

```
. asclogit, or noheader
```

| choice   | Odds Ratio         | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|----------|--------------------|-----------|-------|-------|----------------------|----------|
| car      |                    |           |       |       |                      |          |
| dealer   | 1.070466           | .0368737  | 1.98  | 0.048 | 1.00058              | 1.145232 |
| American | (base alternative) |           |       |       |                      |          |
| Japan    |                    |           |       |       |                      |          |
| sex      | .5859013           | .1840647  | -1.70 | 0.089 | .3165294             | 1.084513 |
| income   | 1.033067           | .013248   | 2.54  | 0.011 | 1.007425             | 1.059361 |
| Europe   |                    |           |       |       |                      |          |
| sex      | 1.768994           | .8031669  | 1.26  | 0.209 | .7265404             | 4.307178 |
| income   | 1.032561           | .0143191  | 2.31  | 0.021 | 1.004874             | 1.061011 |

These results indicate that men (*sex* = 1) are less likely to pick a Japanese car over an American car than women (odds ratio 0.59) but that men are more likely to choose a European car over an American car (odds ratio 1.77). Raising a person's income increases the likelihood that he or she purchases a Japanese or European car; interestingly, the effect of higher income is about the same for these two types of cars.



□ Technical note

McFadden's choice model is related to multinomial logistic regression (see [R] [mlogit](#)). If all the independent variables are case specific, then the two models are identical. We verify this supposition by running the previous example without the alternative-specific variable, `dealer`.

```
. asclogit choice, casevars(sex income) case(id) alternatives(car) nolog
Alternative-specific conditional logit Number of obs = 885
Case variable: id Number of cases = 295
Alternative variable: car Alts per case: min = 3
 avg = 3.0
 max = 3
 Wald chi2(4) = 12.53
Log likelihood = -252.72012 Prob > chi2 = 0.0138
```

| choice   | Coef.              | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|--------------------|-----------|-------|-------|----------------------|-----------|
| American | (base alternative) |           |       |       |                      |           |
| Japan    |                    |           |       |       |                      |           |
| sex      | -.4694799          | .3114939  | -1.51 | 0.132 | -1.079997            | .141037   |
| income   | .0276854           | .0123666  | 2.24  | 0.025 | .0034472             | .0519236  |
| _cons    | -1.962652          | .6216804  | -3.16 | 0.002 | -3.181123            | -.7441807 |
| Europe   |                    |           |       |       |                      |           |
| sex      | .5388441           | .4525279  | 1.19  | 0.234 | -.3480942            | 1.425782  |
| income   | .0273669           | .013787   | 1.98  | 0.047 | .000345              | .0543889  |
| _cons    | -3.180029          | .7546837  | -4.21 | 0.000 | -4.659182            | -1.700876 |

To run `mlogit`, we must rearrange the dataset. `mlogit` requires a dependent variable that indicates the choice—1, 2, or 3—for each individual. We will use `car` as our dependent variable for those observations that represent the choice actually chosen.

(Continued on next page)

```
. keep if choice == 1
(590 observations deleted)

. mlogit car sex income

Iteration 0: log likelihood = -259.1712
Iteration 1: log likelihood = -252.81165
Iteration 2: log likelihood = -252.72014
Iteration 3: log likelihood = -252.72012

Multinomial logistic regression Number of obs = 295
 LR chi2(4) = 12.90
 Prob > chi2 = 0.0118
Log likelihood = -252.72012 Pseudo R2 = 0.0249
```

| car      | Coef.          | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|----------------|-----------|-------|-------|----------------------|-----------|
| American | (base outcome) |           |       |       |                      |           |
| Japan    |                |           |       |       |                      |           |
| sex      | -.4694798      | .3114939  | -1.51 | 0.132 | -1.079997            | .1410371  |
| income   | .0276854       | .0123666  | 2.24  | 0.025 | .0034472             | .0519236  |
| _cons    | -1.962651      | .6216803  | -3.16 | 0.002 | -3.181122            | -.7441801 |
| Europe   |                |           |       |       |                      |           |
| sex      | .5388443       | .4525278  | 1.19  | 0.234 | -.348094             | 1.425783  |
| income   | .027367        | .013787   | 1.98  | 0.047 | .000345              | .0543889  |
| _cons    | -3.18003       | .7546837  | -4.21 | 0.000 | -4.659182            | -1.700877 |

The results are the same except for the model statistic: `asclogit` uses a Wald test and `mlogit` uses a likelihood-ratio test. If you prefer the likelihood-ratio test, you can fit the constant-only model for `asclogit` followed by the full model and use [R] [lrtest](#). The following example will carry this out.

```
. use http://www.stata-press.com/data/r11/choice, clear
. asclogit choice, case(id) alternatives(car)
. estimates store null
. asclogit choice, casevars(sex income) case(id) alternatives(car)
. lrtest null .
```

Technical note

We force you to explicitly identify the case-specific variables in the `casevars()` option to ensure that the program behaves as you expect. For example, an `if` or `in` qualifier may drop observations in such a way that (what was expected to be) an alternative-specific variable turns into a case-specific variable. Here you would probably want `asclogit` to terminate instead of interacting the variable with the alternative indicators. This situation could also occur if `asclogit` drops cases, or observations if you use the `altwise` option, because of missing values.

## Saved results

asclogit saves the following in `e()`:

### Scalars

|                            |                                                |
|----------------------------|------------------------------------------------|
| <code>e(N)</code>          | number of observations                         |
| <code>e(N_case)</code>     | number of cases                                |
| <code>e(k)</code>          | number of parameters                           |
| <code>e(k_alt)</code>      | number of alternatives                         |
| <code>e(k_indvars)</code>  | number of alternative-specific variables       |
| <code>e(k_casevars)</code> | number of case-specific variables              |
| <code>e(k_eq)</code>       | number of equations in <code>e(b)</code>       |
| <code>e(k_eq_model)</code> | number of equations in model Wald test         |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints |
| <code>e(df_m)</code>       | model degrees of freedom                       |
| <code>e(ll)</code>         | log likelihood                                 |
| <code>e(N_clust)</code>    | number of clusters                             |
| <code>e(const)</code>      | constant indicator                             |
| <code>e(i_base)</code>     | base alternative index                         |
| <code>e(chi2)</code>       | $\chi^2$                                       |
| <code>e(F)</code>          | $F$ statistic                                  |
| <code>e(p)</code>          | significance                                   |
| <code>e(alt_min)</code>    | minimum number of alternatives                 |
| <code>e(alt_avg)</code>    | average number of alternatives                 |
| <code>e(alt_max)</code>    | maximum number of alternatives                 |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                      |
| <code>e(ic)</code>         | number of iterations                           |
| <code>e(rc)</code>         | return code                                    |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                    |

*(Continued on next page)*

**Macros**

|                                   |                                                                          |
|-----------------------------------|--------------------------------------------------------------------------|
| <code>e(cmd)</code>               | <code>asclogit</code>                                                    |
| <code>e(cmdline)</code>           | command as typed                                                         |
| <code>e(depvar)</code>            | name of dependent variable                                               |
| <code>e(indvars)</code>           | alternative-specific independent variable                                |
| <code>e(casevars)</code>          | case-specific variables                                                  |
| <code>e(case)</code>              | variable defining cases                                                  |
| <code>e(altvar)</code>            | variable defining alternatives                                           |
| <code>e(alteqs)</code>            | alternative equation names                                               |
| <code>e(alt#)</code>              | alternative labels                                                       |
| <code>e(wtype)</code>             | weight type                                                              |
| <code>e(wexp)</code>              | weight expression                                                        |
| <code>e(title)</code>             | title in estimation output                                               |
| <code>e(clustvar)</code>          | name of cluster variable                                                 |
| <code>e(offset)</code>            | offset                                                                   |
| <code>e(chi2type)</code>          | Wald, type of model $\chi^2$ test                                        |
| <code>e(vce)</code>               | <i>vcetype</i> specified in <code>vce()</code>                           |
| <code>e(vcetype)</code>           | title used to label Std. Err.                                            |
| <code>e(opt)</code>               | type of optimization                                                     |
| <code>e(which)</code>             | max or min; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>         | type of ml method                                                        |
| <code>e(user)</code>              | name of likelihood-evaluator program                                     |
| <code>e(technique)</code>         | maximization technique                                                   |
| <code>e(singularHmethod)</code>   | m-marquardt or hybrid; method used when Hessian is singular              |
| <code>e(crittype)</code>          | optimization criterion                                                   |
| <code>e(datasignature)</code>     | the checksum                                                             |
| <code>e(datasignaturevars)</code> | variables used in calculation of checksum                                |
| <code>e(properties)</code>        | b V                                                                      |
| <code>e(estat_cmd)</code>         | program used to implement <code>estat</code>                             |
| <code>e(predict)</code>           | program used to implement <code>predict</code>                           |
| <code>e(marginsnotok)</code>      | predictions disallowed by margins                                        |

**Matrices**

|                              |                                                                                            |
|------------------------------|--------------------------------------------------------------------------------------------|
| <code>e(b)</code>            | coefficient vector                                                                         |
| <code>e(stats)</code>        | alternative statistics                                                                     |
| <code>e(altvals)</code>      | alternative values                                                                         |
| <code>e(altfreq)</code>      | alternative frequencies                                                                    |
| <code>e(alt_casevars)</code> | indicators for estimated case-specific coefficients— <code>e(k_alt) × e(k_casevars)</code> |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)                                                        |
| <code>e(gradient)</code>     | gradient vector                                                                            |
| <code>e(V)</code>            | variance–covariance matrix of the estimators                                               |
| <code>e(V_modelbased)</code> | model-based variance                                                                       |

**Functions**

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`asclogit` is implemented as an ado-file.

In this model, we have a set of unordered alternatives indexed by  $1, 2, \dots, J$ . Let  $y_{ij}$ ,  $j = 1, \dots, J$ , be an indicator variable for the alternative actually chosen by the  $i$ th individual (case). That is,  $y_{ij} = 1$  if individual  $i$  chose alternative  $j$  and  $y_{ij} = 0$  otherwise. The independent variables come in two forms: alternative specific and case specific. Alternative-specific variables vary among the alternatives (as well as cases), and case-specific variables vary only among cases. Assume that we have  $p$  alternative-specific variables so that for case  $i$  we have a  $J \times p$  matrix,  $\mathbf{X}_i$ . Further, assume that we have  $q$  case-specific variables so that we have a  $1 \times q$  vector  $\mathbf{z}_i$  for case  $i$ . The deterministic component of the random-utility model can then be expressed as

$$\begin{aligned}\eta_i &= \mathbf{X}_i \boldsymbol{\beta} + (\mathbf{z}_i \mathbf{A})' \\ &= \mathbf{X}_i \boldsymbol{\beta} + (\mathbf{z}_i \otimes \mathbf{I}_J) \text{vec}(\mathbf{A}') \\ &= (\mathbf{X}_i, \mathbf{z}_i \otimes \mathbf{I}_J) \begin{pmatrix} \boldsymbol{\beta} \\ \text{vec}(\mathbf{A}') \end{pmatrix} \\ &= \mathbf{X}_i^* \boldsymbol{\beta}^*\end{aligned}$$

As before,  $\boldsymbol{\beta}$  is a  $p \times 1$  vector of alternative-specific regression coefficients, and  $\mathbf{A} = (\alpha_1, \dots, \alpha_J)$  is a  $q \times J$  matrix of case-specific regression coefficients; remember that we must fix one of the  $\alpha_j$  to the constant vector to normalize the location. Here  $\mathbf{I}_J$  is the  $J \times J$  identity matrix,  $\text{vec}()$  is the vector function that creates a vector from a matrix by placing each column of the matrix on top of the other (see [M-5] `vec()`), and  $\otimes$  is the Kronecker product (see [M-2] `op_kronecker`).

We have rewritten the linear equation so that it is in a form that can be used by `clogit`, namely,  $\mathbf{X}_i^* \boldsymbol{\beta}^*$ , where

$$\begin{aligned}\mathbf{X}_i^* &= (\mathbf{X}_i, \mathbf{z}_i \otimes \mathbf{I}_J) \\ \boldsymbol{\beta}^* &= \begin{pmatrix} \boldsymbol{\beta} \\ \text{vec}(\mathbf{A}') \end{pmatrix}\end{aligned}$$

With this in mind, see [Methods and formulas](#) in [R] `clogit` for the computational details of the conditional logit model.

This command supports the clustered version of the Huber/White/sandwich estimator of the variance using `vce(robust)` and `vce(cluster clustvar)`. See [P] `_robust`, in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#). Specifying `vce(robust)` is equivalent to specifying `vce(cluster casevar)`, where `casevar` is the variable that identifies the cases.

## References

- Cameron, A. C., and P. K. Trivedi. 2009. *Microeconometrics Using Stata*. College Station, TX: Stata Press.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice-Hall.
- McFadden, D. L. 1974. Conditional logit analysis of qualitative choice behavior. In *Frontiers in Econometrics*, ed. P. Zarembka, 105–142. New York: Academic Press.

## Also see

- [R] **asclogit postestimation** — Postestimation tools for asclogit
- [R] **asmprobit** — Alternative-specific multinomial probit regression
- [R] **asroprobit** — Alternative-specific rank-ordered probit regression
- [R] **clogit** — Conditional (fixed-effects) logistic regression
- [R] **logistic** — Logistic regression, reporting odds ratios
- [R] **logit** — Logistic regression, reporting coefficients
- [R] **nlogit** — Nested logit regression
- [R] **ologit** — Ordered logistic regression
- [U] **20 Estimation and postestimation commands**

Description

The following postestimation commands are of special interest after `asclogit`:

| commands                        | description                    |
|---------------------------------|--------------------------------|
| <code>estat alternatives</code> | alternative summary statistics |
| <code>estat mfx</code>          | marginal effects               |

For information about these commands, see below.

The following standard postestimation commands are also available:

| commands               | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>     | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estimates</code> | cataloging estimation results                                                                       |
| <code>hausman</code>   | Hausman’s specification test                                                                        |
| <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code>    | likelihood-ratio test                                                                               |
| <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>   | predicted probabilities, estimated linear predictor and its standard error                          |
| <code>predictnl</code> | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details.

Special-interest postestimation commands

- `estat alternatives` displays summary statistics about the alternatives in the estimation sample.
- `estat mfx` computes probability marginal effects.

## Syntax for predict

```
predict [type] newvar [if] [in] [, statistic options]

predict [type] { stub* | newvarlist } [if] [in], scores
```

| statistic | description |
|-----------|-------------|
|-----------|-------------|

|                   |                                                          |
|-------------------|----------------------------------------------------------|
| Main              |                                                          |
| <code>pr</code>   | probability that each alternative is chosen; the default |
| <code>xb</code>   | linear prediction                                        |
| <code>stdp</code> | standard error of the linear prediction                  |

| options | description |
|---------|-------------|
|---------|-------------|

|                           |                                                                                            |
|---------------------------|--------------------------------------------------------------------------------------------|
| Main                      |                                                                                            |
| <code>*k(condspec)</code> | condition on <i>condspec</i> alternatives chosen by each case when computing probabilities |
| <code>altwise</code>      | use alternative-wise deletion instead of casewise deletion when computing probabilities    |
| <code>nooffset</code>     | ignore the <code>offset()</code> variable specified in <code>asclogit</code>               |

`*k(condspec)` may be used only with `pr`.

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

- `pr` computes the probability of choosing each alternative conditioned on each case choosing `k(condspec)` alternatives. This is the default statistic with default `k(1)`; one alternative per case is chosen.
- `xb` computes the linear prediction.
- `stdp` computes the standard error of the linear prediction.
- `k(#|observed)` conditions the probability on choosing `#` alternatives per case, or use `k(observed)` to condition on the observed number of alternatives chosen. The default is `k(1)`. This option may be used only with the `pr` option.
- `altwise` specifies that alternative-wise deletion be used when marking out observations due to missing values in your variables. The default is to use casewise deletion. The `xb` and `stdp` options always use alternative-wise deletion.
- `nooffset` is relevant only if you specified `offset(varname)` for `asclogit`. It modifies the calculations made by `predict` so that they ignore the offset variable; the linear prediction is treated as  $x\beta$  rather than as  $x\beta + \text{offset}$ .
- `scores` calculates the scores for each coefficient in `e(b)`. This option requires a new variable list of length equal to the number of columns in `e(b)`. Otherwise, use the `stub*` option to have `predict` generate enumerated variables with prefix `stub`.



## Syntax for estat alternatives

```
estat alternatives
```

## Menu

Statistics > Postestimation > Reports and statistics

## Syntax for estat mfx

```
estat mfx [if] [in] [, options]
```

| <i>options</i>                                                | description                                                               |
|---------------------------------------------------------------|---------------------------------------------------------------------------|
| Main                                                          |                                                                           |
| <u>varlist</u> ( <i>varlist</i> )                             | display marginal effects for <i>varlist</i>                               |
| <u>at</u> (mean [ <i>atlist</i> ]   median [ <i>atlist</i> ]) | calculate marginal effects at these values                                |
| <u>k</u> (#)                                                  | condition on the number of alternatives chosen to be #                    |
| Options                                                       |                                                                           |
| <u>level</u> (#)                                              | set confidence interval level; default is level(95)                       |
| <u>nodiscrete</u>                                             | treat indicator variables as continuous                                   |
| <u>noesample</u>                                              | do not restrict calculation of means and medians to the estimation sample |
| <u>nowght</u>                                                 | ignore weights when calculating means and medians                         |

## Menu

Statistics > Postestimation > Reports and statistics

## Options for estat mfx

### Main

**varlist**(*varlist*) specifies the variables for which to display marginal effects. The default is all variables.

**at**(mean [*atlist*] | median [*atlist*]) specifies the values at which the marginal effects are to be calculated. *atlist* is

```
[[alternative:variable = #] [variable = #] [alternative:offset = #] [...]
```

The default is to calculate the marginal effects at the means of the independent variables by using the estimation sample, **at**(mean). If **offset**() is used during estimation, the means of the offsets (by alternative) are computed by default.

After specifying the summary statistic, you can specify a series of specific values for variables. You can specify values for alternative-specific variables by **alternative**, or you can specify one value for all alternatives. You can specify only one value for case-specific variables. You specify values for the **offset**() variable (if present) the same way as for alternative-specific variables. For example, in the **choice** dataset (car choice), **income** is a case-specific variable, whereas **dealer** is an alternative-specific variable. The following would be a legal syntax for **estat mfx**:

```
. estat mfx, at(mean American:dealer=18 income=40)
```

When `nodiscrete` is not specified, `at(mean [atlist])` or `at(median [atlist])` has no effect on computing marginal effects for indicator variables, which are calculated as the discrete change in the simulated probability as the indicator variable changes from 0 to 1.

The mean and median computations respect any `if` or `in` qualifiers, so you can restrict the data over which the statistic is computed. You can even restrict the values to a specific case, e.g.,

```
. estat mfx if case==21
```

`k(#)` computes the probabilities conditioned on # alternatives chosen. The default is one alternative chosen.

#### Options

`level(#)` sets the confidence level; default is `level(95)`.

`nodiscrete` specifies that indicator variables be treated as continuous variables. An indicator variable is one that takes on the value 0 or 1 in the estimation sample. By default, the discrete change in the simulated probability is computed as the indicator variable changes from 0 to 1.

`noesample` specifies that the whole dataset be considered instead of only those marked in the `e(sample)` defined by the `asclogit` command.

`nowght` specifies that weights be ignored when calculating the medians.

## Remarks

Remarks are presented under the following headings:

*Predicted probabilities*

*Obtaining estimation statistics*

## Predicted probabilities

After fitting a McFadden's choice model with alternative-specific conditional logistic regression, you can use `predict` to obtain the estimated probability of alternative choices given case profiles.

### ► Example 1

In [example 1](#) of [\[R\] asclogit](#), we fit a model of consumer choice of automobile. The alternatives are nationality of the automobile manufacturer: American, Japanese, or European. There is one alternative-specific variable in the model, `dealer`, which contains the number of dealerships of each nationality in the consumer's city. The case-specific variables are `sex`, the consumer's sex, and `income`, the consumer's income in thousands of dollars.

```
. use http://www.stata-press.com/data/r11/choice
. asclogit choice dealer, casevars(sex income) case(id) alternatives(car)
 (output omitted)
. predict p
(option pr assumed; Pr(car))
. predict p2, k(2)
(option pr assumed; Pr(car))
. format p p2 %6.4f
```

```
. list car choice dealer sex income p p2 in 1/9, sepby(id)
```

|    | car      | choice | dealer | sex  | income | p      | p2     |
|----|----------|--------|--------|------|--------|--------|--------|
| 1. | American | 0      | 18     | male | 46.7   | 0.6025 | 0.8589 |
| 2. | Japan    | 0      | 8      | male | 46.7   | 0.2112 | 0.5974 |
| 3. | Europe   | 1      | 5      | male | 46.7   | 0.1863 | 0.5437 |
| 4. | American | 1      | 17     | male | 26.1   | 0.7651 | 0.9293 |
| 5. | Japan    | 0      | 6      | male | 26.1   | 0.1282 | 0.5778 |
| 6. | Europe   | 0      | 2      | male | 26.1   | 0.1067 | 0.4929 |
| 7. | American | 1      | 12     | male | 32.7   | 0.6519 | 0.8831 |
| 8. | Japan    | 0      | 6      | male | 32.7   | 0.1902 | 0.5995 |
| 9. | Europe   | 0      | 2      | male | 32.7   | 0.1579 | 0.5174 |

◀

## Obtaining estimation statistics

Here we will demonstrate the specialized `estat` subcommands after `asclogit`. Use `estat alternatives` to obtain a table of alternative statistics. The table will contain the alternative values, labels (if any), the number of cases in which each alternative is present, the frequency that the alternative is selected, and the percent selected.

Use `estat mfx` to obtain marginal effects after `asclogit`.

### ► Example 2

We will continue with the automobile choice example, where we first list the alternative statistics and then compute the marginal effects at the mean income in our sample, assuming that there are five automobile dealers for each nationality. We will evaluate the probabilities for females because `sex` is coded 0 for females, and we will be obtaining the discrete change from 0 to 1.

```
. estat alternatives
```

Alternatives summary for car

| index | Alternative value | label    | Cases present | Frequency selected | Percent selected |
|-------|-------------------|----------|---------------|--------------------|------------------|
| 1     | 1                 | American | 295           | 192                | 65.08            |
| 2     | 2                 | Japan    | 295           | 64                 | 21.69            |
| 3     | 3                 | Europe   | 295           | 39                 | 13.22            |

```
. estat mfx, at(dealer=0 sex=0) varlist(sex income)
```

```
Pr(choice = American|1 selected) = .41964329
```

| variable | dp/dx    | Std. Err. | z     | P> z  | [        | 95% C.I. | ] | X      |
|----------|----------|-----------|-------|-------|----------|----------|---|--------|
| casevars |          |           |       |       |          |          |   |        |
| sex*     | .026238  | .068311   | 0.38  | 0.701 | -.107649 | .160124  |   | 0      |
| income   | -.007891 | .002674   | -2.95 | 0.003 | -.013132 | -.00265  |   | 42.097 |

(\*) dp/dx is for discrete change of indicator variable from 0 to 1

Pr(choice = Japan|1 selected) = .42696187

| variable | dp/dx    | Std. Err. | z     | P> z  | [        | 95% C.I. | ] | X      |
|----------|----------|-----------|-------|-------|----------|----------|---|--------|
| casevars |          |           |       |       |          |          |   |        |
| sex*     | -.161164 | .079238   | -2.03 | 0.042 | -.316468 | -.005859 |   | 0      |
| income   | .005861  | .002997   | 1.96  | 0.051 | -.000014 | .011735  |   | 42.097 |

(\*) dp/dx is for discrete change of indicator variable from 0 to 1

Pr(choice = Europe|1 selected) = .15339484

| variable | dp/dx   | Std. Err. | z    | P> z  | [        | 95% C.I. | ] | X      |
|----------|---------|-----------|------|-------|----------|----------|---|--------|
| casevars |         |           |      |       |          |          |   |        |
| sex*     | .134926 | .076556   | 1.76 | 0.078 | -.015122 | .284973  |   | 0      |
| income   | .00203  | .001785   | 1.14 | 0.255 | -.001469 | .00553   |   | 42.097 |

(\*) dp/dx is for discrete change of indicator variable from 0 to 1

The marginal effect of `income` indicates that there is a lower chance for a consumer to buy American automobiles with an increase in income. There is an indication that men have a higher preference for European automobiles than women but a lower preference for Japanese automobiles. We did not include the marginal effects for `dealer` because we view these as nuisance parameters, so we adjusted the probabilities by fixing `dealer` to a constant, 0.



## Saved results

`estat mfx` saves the following in `r()`:

### Scalars

`r(pr_`*alt*`)` scalars containing the computed probability of each alternative evaluated at the value that is labeled `X` in the table output. Here *alt* are the labels in the macro `e(alteqs)`.

### Matrices

`r(`*alt*`)` matrices containing the computed marginal effects and associated statistics. There is one matrix for each alternative, where *alt* are the labels in the macro `e(alteqs)`. Column 1 of each matrix contains the marginal effects; column 2, their standard errors; column 3, their *z* statistics; and columns 4 and 5, the confidence intervals. Column 6 contains the values of the independent variables used to compute the probabilities `r(pr_`*alt*`)`.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

The deterministic component of the random-utility model can be expressed as

$$\begin{aligned}
 \eta &= \mathbf{X}\beta + (\mathbf{z}\mathbf{A})' \\
 &= \mathbf{X}\beta + (\mathbf{z} \otimes \mathbf{I}_J) \text{vec}(\mathbf{A}') \\
 &= (\mathbf{X}, \mathbf{z} \otimes \mathbf{I}_J) \begin{pmatrix} \beta \\ \text{vec}(\mathbf{A}') \end{pmatrix} \\
 &= \mathbf{X}^* \beta^*
 \end{aligned}$$

where  $\mathbf{X}$  is the  $J \times p$  matrix containing the alternative-specific covariates,  $\mathbf{z}$  is a  $1 \times q$  vector of case-specific variables,  $\beta$  is a  $p \times 1$  vector of alternative-specific regression coefficients, and  $\mathbf{A} = (\alpha_1, \dots, \alpha_J)$  is a  $q \times J$  matrix of case-specific regression coefficients (with one of the  $\alpha_j$  fixed to the constant). Here  $\mathbf{I}_J$  is the  $J \times J$  identity matrix,  $\text{vec}()$  is the vector function that creates a vector from a matrix by placing each column of the matrix on top of the other (see [M-5] [vec\(\)](#)), and  $\otimes$  is the Kronecker product (see [M-2] [op\\_kronecker](#)).

We have rewritten the linear equation so that it is a form that we all recognize, namely,  $\eta = \mathbf{X}^* \beta^*$ , where

$$\begin{aligned}
 \mathbf{X}^* &= (\mathbf{X}, \mathbf{z} \otimes \mathbf{I}_J) \\
 \beta^* &= \begin{pmatrix} \beta \\ \text{vec}(\mathbf{A}') \end{pmatrix}
 \end{aligned}$$

To compute the marginal effects, we use the derivative of the log likelihood  $\partial \ell(\mathbf{y}|\eta)/\partial \eta$ , where  $\ell(\mathbf{y}|\eta) = \log \Pr(\mathbf{y}|\eta)$  is the log of the probability of the choice indicator vector  $\mathbf{y}$  given the linear predictor vector  $\eta$ . Namely,

$$\begin{aligned}
 \frac{\partial \Pr(\mathbf{y}|\eta)}{\partial \text{vec}(\mathbf{X}^*)'} &= \Pr(\mathbf{y}|\eta) \frac{\partial \ell(\mathbf{y}|\eta)}{\partial \eta'} \frac{\partial \eta}{\partial \text{vec}(\mathbf{X}^*)'} \\
 &= \Pr(\mathbf{y}|\eta) \frac{\partial \ell(\mathbf{y}|\eta)}{\partial \eta'} (\beta^{*'} \otimes \mathbf{I}_J)
 \end{aligned}$$

The standard errors of the marginal effects are computed using the delta method.

## Also see

[R] [asclogit](#) — Alternative-specific conditional logit (McFadden’s choice) model

[U] [20 Estimation and postestimation commands](#)

## Syntax

```
asmprobit depvar [indepvars] [if] [in] [weight] , case(varname)
 alternatives(varname) [options]
```

| <i>options</i>                                         | description                                                                                                                                                                 |
|--------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                                  |                                                                                                                                                                             |
| * <b>case</b> ( <i>varname</i> )                       | use <i>varname</i> to identify cases                                                                                                                                        |
| * <b>alternatives</b> ( <i>varname</i> )               | use <i>varname</i> to identify the alternatives available for each case                                                                                                     |
| <b>casevars</b> ( <i>varlist</i> )                     | case-specific variables                                                                                                                                                     |
| <b>constraints</b> ( <i>constraints</i> )              | apply specified linear constraints                                                                                                                                          |
| <b>collinear</b>                                       | keep collinear variables                                                                                                                                                    |
| Model 2                                                |                                                                                                                                                                             |
| <b>correlation</b> ( <i>correlation</i> )              | correlation structure of the latent-variable errors                                                                                                                         |
| <b>stddev</b> ( <i>stddev</i> )                        | variance structure of the latent-variable errors                                                                                                                            |
| <b>structural</b>                                      | use the structural covariance parameterization; default is the differenced covariance parameterization                                                                      |
| <b>factor</b> (#)                                      | use the factor covariance structure with dimension #                                                                                                                        |
| <b>noconstant</b>                                      | suppress the alternative-specific constant terms                                                                                                                            |
| <b>basealternative</b> (#   <i>lbl</i>   <i>str</i> )  | alternative used for normalizing location                                                                                                                                   |
| <b>scalealternative</b> (#   <i>lbl</i>   <i>str</i> ) | alternative used for normalizing scale                                                                                                                                      |
| <b>altwise</b>                                         | use alternative-wise deletion instead of casewise deletion                                                                                                                  |
| SE/Robust                                              |                                                                                                                                                                             |
| <b>vce</b> ( <i>vcetype</i> )                          | <i>vcetype</i> may be <code>oim</code> , <code>robust</code> , <code>cluster</code> <i>clustvar</i> , <code>opg</code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                                              |                                                                                                                                                                             |
| <b>level</b> (#)                                       | set confidence level; default is <code>level(95)</code>                                                                                                                     |
| <b>notransform</b>                                     | do not transform variance–covariance estimates to the standard deviation and correlation metric                                                                             |
| <b>nocnsreport</b>                                     | do not display constraints                                                                                                                                                  |

## Integration

|                                              |                                                                |
|----------------------------------------------|----------------------------------------------------------------|
| <u>intmethod</u> ( <i>seqtype</i> )          | type of quasi- or pseudouniform point set                      |
| <u>intpoints</u> (#)                         | number of points in each sequence                              |
| <u>intburn</u> (#)                           | starting index in the Hammersley or Halton sequence            |
| <u>intseed</u> ( <i>code</i>   #)            | pseudouniform random-number seed                               |
| <u>antithetics</u>                           | use antithetic draws                                           |
| <u>nopivot</u>                               | do not use integration interval pivoting                       |
| <u>initbhhh</u> (#)                          | use the BHHH optimization algorithm for the first # iterations |
| <u>favor</u> ( <u>speed</u>   <u>space</u> ) | favor speed or space when generating integration points        |

## Maximization

|                         |                                  |
|-------------------------|----------------------------------|
| <u>maximize_options</u> | control the maximization process |
|-------------------------|----------------------------------|

|                                |                                                           |
|--------------------------------|-----------------------------------------------------------|
| <sup>†</sup> <u>coeflegend</u> | display coefficients' legend instead of coefficient table |
|--------------------------------|-----------------------------------------------------------|

| <i>correlation</i>            | description                                                                                                                         |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <u>unstructured</u>           | one correlation parameter for each pair of alternatives; correlations with the <code>basealternative()</code> are zero; the default |
| <u>exchangeable</u>           | one correlation parameter common to all pairs of alternatives; correlations with the <code>basealternative()</code> are zero        |
| <u>independent</u>            | constrain all correlation parameters to zero                                                                                        |
| <u>pattern</u> <i>matname</i> | user-specified matrix identifying the correlation pattern                                                                           |
| <u>fixed</u> <i>matname</i>   | user-specified matrix identifying the fixed and free correlation parameters                                                         |

| <i>stddev</i>                 | description                                                                                                                                             |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>heteroskedastic</u>        | estimate standard deviation for each alternative; standard deviations for <code>basealternative()</code> and <code>scalealternative()</code> set to one |
| <u>homoskedastic</u>          | all standard deviations are one                                                                                                                         |
| <u>pattern</u> <i>matname</i> | user-specified matrix identifying the standard deviation pattern                                                                                        |
| <u>fixed</u> <i>matname</i>   | user-specified matrix identifying the fixed and free standard deviation parameters                                                                      |

| <i>seqtype</i>    | description                    |
|-------------------|--------------------------------|
| <u>hammersley</u> | Hammersley point set           |
| <u>halton</u>     | Halton point set               |
| <u>random</u>     | uniform pseudorandom point set |

\*`case(varname)` and `alternatives(varname)` are required.

<sup>†</sup>`coeflegend` does not appear in the dialog box.

`bootstrap`, `by`, `jackknife`, `statsby`, and `xi` are allowed; see [U] 11.1.10 Prefix commands.

Weights are not allowed with the `bootstrap` prefix.

`fweights`, `iweights`, and `pweights` are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

## Menu

Statistics > Categorical outcomes > Alternative-specific multinomial probit

## Description

`asmprobit` fits multinomial probit (MNP) models by using maximum simulated likelihood (MSL) implemented by the Geweke–Hajivassiliou–Keane (GHK) algorithm. By estimating the variance–covariance parameters of the latent-variable errors, the model allows you to relax the independence of irrelevant alternatives (IIA) property that is characteristic of the multinomial logistic model.

`asmprobit` requires multiple observations for each case (decision), where each observation represents an alternative that may be chosen. The cases are identified by the variable specified in the `case()` option, whereas the alternatives are identified by the variable specified in the `alternative()` option. The outcome (chosen alternative) is identified by a value of 1 in `depvar`, with 0 indicating the alternatives that were not chosen; only one alternative may be chosen for each case.

`asmprobit` allows two types of independent variables: alternative-specific variables and case-specific variables. Alternative-specific variables vary across both cases and alternatives and are specified in `indepvars`. Case-specific variables vary only across cases and are specified in the `casevars()` option.

## Options

### Model

`case(varname)` specifies the variable that identifies each case. This variable identifies the individuals or entities making a choice. `case()` is required.

`alternatives(varname)` specifies the variable that identifies the alternatives available for each case. The number of alternatives can vary with each case; the maximum number of alternatives is 20. `alternatives()` is required.

`casevars(varlist)` specifies the case-specific variables that are constant for each `case()`. If there are a maximum of  $J$  alternatives, there will be  $J - 1$  sets of coefficients associated with `casevars()`.

`constraints(constraints)`, `collinear`; see [\[R\] estimation options](#).

### Model 2

`correlation(correlation)` specifies the correlation structure of the latent-variable errors.

`correlation(unstructured)` is the most general and has  $J(J - 3)/2 + 1$  unique correlation parameters. This is the default unless `stdev()` or `structural` are specified.

`correlation(exchangeable)` provides for one correlation coefficient common to all latent variables, except the latent variable associated with the `basealternative()` option.

`correlation(independent)` assumes that all correlations are zero.

`correlation(pattern matname)` and `correlation(fixed matname)` give you more flexibility in defining the correlation structure. See [Variance structures](#) later in this entry for more information.

`stddev(stddev)` specifies the variance structure of the latent-variable errors.

`stddev(heteroskedastic)` is the most general and has  $J - 2$  estimable parameters. The standard deviations of the latent-variable errors for the alternatives specified in `basealternative()` and `scalealternative()` are fixed to one.



`stddev(homoskedastic)` constrains all the standard deviations to equal one.

`stddev(pattern matname)` and `stddev(fixed matname)` give you added flexibility in defining the standard deviation parameters. See [Variance structures](#) later in this entry for more information.

`structural` requests the  $J \times J$  structural covariance parameterization instead of the default  $J-1 \times J-1$  differenced covariance parameterization (the covariance of the latent errors differenced with that of the base alternative). The differenced covariance parameterization will achieve the same MSL regardless of the choice of `basealternative()` and `scalealternative()`. On the other hand, the structural covariance parameterization imposes more normalizations that may bound the model away from the maximum likelihood estimates and thus prevent convergence with some datasets or choices of `basealternative()` and `scalealternative()`.

`factor(#)` requests that the factor covariance structure of dimension `#` be used. The `factor()` option can be used with the `structural` option but cannot be used with `stddev()` or `correlation()`. A  $\# \times J$  (or  $\# \times J-1$ ) matrix,  $\mathbf{C}$ , is used to factor the covariance matrix as  $\mathbf{I} + \mathbf{C}'\mathbf{C}$ , where  $\mathbf{I}$  is the identity matrix of dimension  $J$  (or  $J-1$ ). The column dimension of  $\mathbf{C}$  depends on whether the covariance is structural or differenced. The row dimension of  $\mathbf{C}$ , `#`, must be less than or equal to  $\text{floor}((J(J-1)/2-1)/(J-2))$ , because there are only  $J(J-1)/2-1$  identifiable variance-covariance parameters. This covariance parameterization may be useful for reducing the number of covariance parameters that need to be estimated.

If the covariance is structural, the column of  $\mathbf{C}$  corresponding to the base alternative contains zeros. The column corresponding to the scale alternative has a one in the first row and zeros elsewhere. If the covariance is differenced, the column corresponding to the scale alternative (differenced with the base) has a one in the first row and zeros elsewhere.

`noconstant` suppresses the  $J-1$  alternative-specific constant terms.

`basealternative(#|lbl|str)` specifies the alternative used to normalize the latent-variable location (also referred to as the level of utility). The base alternative may be specified as a number, label, or string. The standard deviation for the latent-variable error associated with the base alternative is fixed to one, and its correlations with all other latent-variable errors are set to zero. The default is the first alternative when sorted. If a `fixed` or `pattern` matrix is given in the `stddev()` and `correlation()` options, the `basealternative()` will be implied by the fixed standard deviations and correlations in the matrix specifications. `basealternative()` cannot be equal to `scalealternative()`.

`scalealternative(#|lbl|str)` specifies the alternative used to normalize the latent-variable scale (also referred to as the scale of utility). The scale alternative may be specified as a number, label, or string. The default is to use the second alternative when sorted. If a `fixed` or `pattern` matrix is given in the `stddev()` option, the `scalealternative()` will be implied by the fixed standard deviations in the matrix specification. `scalealternative()` cannot be equal to `basealternative()`.

If a `fixed` or `pattern` matrix is given for the `stddev()` option, the base alternative and scale alternative are implied by the standard deviations and correlations in the matrix specifications, and they need not be specified in the `basealternative()` and `scalealternative()` options.

`altwise` specifies that alternative-wise deletion be used when marking out observations due to missing values in your variables. The default is to use casewise deletion; that is, the entire group of observations making up a case is deleted if any missing values are encountered. This option does not apply to observations that are marked out by the `if` or `in` qualifier or the `by` prefix.

## SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

If specifying `vce(bootstrap)` or `vce(jackknife)`, you must also specify `basealternative()` and `scalealternative()`.

## Reporting

`level(#)`; see [R] [estimation options](#).

`notransform` prevents retransforming the Cholesky-factored variance–covariance estimates to the correlation and standard deviation metric.

This option has no effect if `structural` is not specified because the default differenced variance–covariance estimates have no interesting interpretation as correlations and standard deviations. `notransform` also has no effect if the `correlation()` and `stddev()` options are specified with anything other than their default values. Here it is generally not possible to factor the variance–covariance matrix, so optimization is already performed using the standard deviation and correlation representations.

`nocnsreport`; see [R] [estimation options](#).

## Integration

`intmethod(hammersley | halton | random)` specifies the method of generating the point sets used in the quasi–Monte Carlo integration of the multivariate normal density. `intmethod(hammersley)`, the default, uses the Hammersley sequence; `intmethod(halton)` uses the Halton sequence; and `intmethod(random)` uses a sequence of uniform random numbers.

`intpoints(#)` specifies the number of points to use in the quasi–Monte Carlo integration. If this option is not specified, the number of points is  $50 \times J$  if `intmethod(hammersley)` or `intmethod(halton)` is used and  $100 \times J$  if `intmethod(random)` is used. Larger values of `intpoints()` provide better approximations of the log likelihood, but at the cost of added computation time.

`intburn(#)` specifies where in the Hammersley or Halton sequence to start, which helps reduce the correlation between the sequences of each dimension. The default is 0. This option may not be specified with `intmethod(random)`.

`intseed(code | #)` specifies the seed to use for generating the uniform pseudorandom sequence. This option may be specified only with `intmethod(random)`. `code` refers to a string that records the state of the random-number generator `runiform()`; see [R] [set seed](#). An integer value `#` may be used also. The default is to use the current seed value from Stata’s uniform random-number generator, which can be obtained from `c(seed)`.

`antithetics` specifies that antithetic draws be used. The antithetic draw for the  $J - 1$  vector uniform-random variables,  $\mathbf{x}$ , is  $1 - \mathbf{x}$ .

`nopivot` turns off integration interval pivoting. By default, `asmprobit` will pivot the wider intervals of integration to the interior of the multivariate integration. This improves the accuracy of the quadrature estimate. However, discontinuities may result in the computation of numerical second-order derivatives using finite differencing (for the Newton–Raphson optimize technique, `tech(nr)`) when few simulation points are used, resulting in a nonpositive-definite Hessian. `asmprobit` uses the Broyden–Fletcher–Goldfarb–Shanno optimization algorithm, by default, which does not require computing the Hessian numerically using finite differencing.

`initbhhh(#)` specifies that the Berndt–Hall–Hall–Hausman (BHHH) algorithm be used for the initial # optimization steps. This option is the only way to use the BHHH algorithm along with other optimization techniques. The algorithm switching feature of `ml`'s `technique()` option cannot include `bhhh`.

`favor(speed|space)` instructs `asmprobit` to favor either speed or space when generating the integration points. `favor(speed)` is the default. When favoring speed, the integration points are generated once and stored in memory, thus increasing the speed of evaluating the likelihood. This speed increase can be seen when there are many cases or when the user specifies a large number of integration points, `intpoints(#)`. When favoring space, the integration points are generated repeatedly with each likelihood evaluation.

For unbalanced data, where the number of alternatives varies with each case, the estimates computed using `intmethod(random)` will vary slightly between `favor(speed)` and `favor(space)`. This is because the uniform sequences will not be identical, even when initiating the sequences using the same uniform seed, `intseed(code|#)`. For `favor(speed)`, `n` case blocks of `intpoints(#)`  $\times$   $J - 2$  uniform points are generated, where  $J$  is the maximum number of alternatives. For `favor(space)`, the column dimension of the matrices of points varies with the number of alternatives that each case has.

#### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [R] [maximize](#).

The following options may be particularly useful in obtaining convergence with `asmprobit`: `difficult`, `technique(algorithm_spec)`, `nrtolerance(#)`, `nonrtolerance`, and `from(init_specs)`.

If `technique()` contains more than one algorithm specification, `bhhh` cannot be one of them. To use the BHHH algorithm with another algorithm, use the `initbhhh()` option and specify the other algorithm in `technique()`.

Setting the optimization type to `technique(bhhh)` resets the default `vcetype` to `vce(opg)`.

The following option is available with `asmprobit` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

Remarks are presented under the following headings:

[Introduction](#)

[Variance structures](#)

## Introduction

The MNP model is used with discrete dependent variables that take on more than two outcomes that do not have a natural ordering. The stochastic error terms are assumed to have a multivariate normal distribution that is heteroskedastic and correlated. Say that you have a set of  $J$  unordered alternatives that are modeled by a regression of both case-specific and alternative-specific covariates. A “case” refers to the information on one decision maker. Underlying the model is the set of  $J$  latent variables (utilities),

$$\eta_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_i\boldsymbol{\alpha}_j + \xi_{ij} \quad (1)$$

where  $i$  denotes cases and  $j$  denotes alternatives.  $\mathbf{x}_{ij}$  is a  $1 \times p$  vector of alternative-specific variables,  $\boldsymbol{\beta}$  is a  $p \times 1$  vector of parameters,  $\mathbf{z}_i$  is a  $1 \times q$  vector of case-specific variables,  $\boldsymbol{\alpha}_j$  is a  $q \times 1$  vector of parameters for the  $j$ th alternative, and  $\boldsymbol{\xi}_i = (\xi_{i1}, \dots, \xi_{iJ})$  is distributed multivariate normal with mean zero and covariance matrix  $\boldsymbol{\Omega}$ . The decision maker selects the alternative whose latent variable is highest.

Because the MNP model allows for a general covariance structure in  $\boldsymbol{\xi}_{ij}$ , it does not impose the IIA property inherent in multinomial logistic and conditional logistic models. That is, the MNP model permits the odds of choosing one alternative over another to depend on the remaining alternatives. For example, consider the choice of travel mode between two cities: air, train, bus, or car, as a function of the travel mode cost, travel time (alternative-specific variables), and an individual's income (a case-specific variable). The odds of choosing air travel over a bus may not be independent of the train alternative because both bus and train travel are public ground transportation. That is, the probability of choosing air travel is  $\Pr(\eta_{\text{air}} > \eta_{\text{bus}}, \eta_{\text{air}} > \eta_{\text{train}}, \eta_{\text{air}} > \eta_{\text{car}})$ , and the two events  $\eta_{\text{air}} > \eta_{\text{bus}}$  and  $\eta_{\text{air}} > \eta_{\text{train}}$  may be correlated.

An alternative to MNP that will allow a nested correlation structure in  $\boldsymbol{\xi}_{ij}$  is the nested logit model (see [R] `nlogit`).

The added flexibility of the MNP model does impose a significant computation burden because of the need to evaluate probabilities from the multivariate normal distribution. These probabilities are evaluated using simulation techniques because a closed-form solution does not exist. See [Methods and formulas](#) for more information.

Not all the  $J$  sets of regression coefficients  $\boldsymbol{\alpha}_j$  are identifiable, nor are all  $J(J+1)/2$  elements of the variance–covariance matrix  $\boldsymbol{\Omega}$ . As described by [Train \(2003\)](#), the model requires normalization because both the location (level) and scale of the latent variable are irrelevant. Increasing the latent variables by a constant does not change which  $\eta_{ij}$  is the maximum for decision maker  $i$ , nor does multiplying them by a constant. To normalize location, we choose an alternative, indexed by  $k$ , say, and take the difference between the latent variable  $k$  and the  $J-1$  others,

$$\begin{aligned} v_{ijk} &= \eta_{ij} - \eta_{ik} \\ &= (\mathbf{x}_{ij} - \mathbf{x}_{ik})\boldsymbol{\beta} + \mathbf{z}_i(\boldsymbol{\alpha}_j - \boldsymbol{\alpha}_k) + \xi_{ij} - \xi_{ik} \\ &= \boldsymbol{\delta}_{ij'}\boldsymbol{\beta} + \mathbf{z}_i\boldsymbol{\gamma}_{j'} + \epsilon_{ij'} \\ &= \lambda_{ij'} + \epsilon_{ij'} \end{aligned} \tag{2}$$

where  $j' = j$  if  $j < k$  and  $j' = j - 1$  if  $j > k$ , so that  $j' = 1, \dots, J-1$ . One can now work with the  $(J-1) \times (J-1)$  covariance matrix  $\boldsymbol{\Sigma}_{(k)}$  for  $\boldsymbol{\epsilon}'_i = (\epsilon_{i1}, \dots, \epsilon_{i,J-1})$ . The  $k$ th alternative here is the `basealternative()` in `asmprobit`. From (2), the probability that decision maker  $i$  chooses alternative  $k$ , for example, is

$$\begin{aligned} \Pr(i \text{ chooses } k) &= \Pr(v_{i1k} \leq 0, \dots, v_{i,J-1,k} \leq 0) \\ &= \Pr(\epsilon_{i1} \leq -\lambda_{i1}, \dots, \epsilon_{i,J-1} \leq -\lambda_{i,J-1}) \end{aligned}$$

To normalize for scale, one of the diagonal elements of  $\boldsymbol{\Sigma}_{(k)}$  must be fixed to a constant. In `asmprobit`, this is the error variance for the alternative specified by `scalealternative()`. Thus there are a total of, at most,  $J(J-1)/2 - 1$  identifiable variance–covariance parameters. See [Variance structures](#) below for more on this issue.

In fact, the model is slightly more general in that not all cases need to have faced all  $J$  alternatives. The model allows for situations in which some cases chose among all possible alternatives, whereas other cases were given a choice among a subset of them, and perhaps other cases were given a choice among a different subset. The number of observations for each case is equal to the number of alternatives faced.

The MNP model is often motivated using a random-utility consumer-choice framework. Equation (1) represents the utility that consumer  $i$  receives from good  $j$ . The consumer purchases the good for which the utility is highest. Because utility is ordinal, all that matters is the ranking of the utilities from the alternatives. Thus one must normalize for location and scale.

## Example 1

Application of MNP models is common in the analysis of transportation data. [Greene \(2008, sec. 23.11.7\)](#) uses travel-mode choice data between Sydney and Melbourne to demonstrate estimating parameters of various discrete-choice models. The data contain information on 210 individuals' choices of travel mode. The four alternatives are air, train, bus, and car, with indices 1, 2, 3, and 4, respectively. One alternative-specific variable is `travelcost`, a measure of generalized cost of travel that is equal to the sum of in-vehicle cost and a wagelike measure times the amount of time spent traveling. A second alternative-specific variable is the terminal time, `termtime`, which is zero for car transportation. Household income, `income`, is a case-specific variable.

```
. use http://www.stata-press.com/data/r11/travel
. list id mode choice travelcost termtime income in 1/12, sepby(id)
```

|     | id | mode  | choice | travel-t | termtime | income |
|-----|----|-------|--------|----------|----------|--------|
| 1.  | 1  | air   | 0      | 70       | 69       | 35     |
| 2.  | 1  | train | 0      | 71       | 34       | 35     |
| 3.  | 1  | bus   | 0      | 70       | 35       | 35     |
| 4.  | 1  | car   | 1      | 30       | 0        | 35     |
| 5.  | 2  | air   | 0      | 68       | 64       | 30     |
| 6.  | 2  | train | 0      | 84       | 44       | 30     |
| 7.  | 2  | bus   | 0      | 85       | 53       | 30     |
| 8.  | 2  | car   | 1      | 50       | 0        | 30     |
| 9.  | 3  | air   | 0      | 129      | 69       | 40     |
| 10. | 3  | train | 0      | 195      | 34       | 40     |
| 11. | 3  | bus   | 0      | 149      | 35       | 40     |
| 12. | 3  | car   | 1      | 101      | 0        | 40     |

The model of travel choice is

$$\eta_{ij} = \beta_1 \text{travelcost}_{ij} + \beta_2 \text{termtime}_{ij} + \alpha_{1j} \text{income}_i + \alpha_{0j} + \xi_{ij}$$

The alternatives can be grouped as air and ground travel. With this in mind, we set the `air` alternative to be the `basealternative()` and choose `train` as the scaling alternative. Because these are the first and second alternatives in the `mode` variable, they are also the defaults.

(Continued on next page)

```
. asmprobit choice travelcost termtime, casevars(income) case(id)
> alternatives(mode)

(output omitted)

Alternative-specific multinomial probit
Case variable: id
Alternative variable: mode

Number of obs = 840
Number of cases = 210
Alts per case: min = 4
 avg = 4.0
 max = 4

Integration sequence: Hammersley
Integration points: 200
Log simulated-likelihood = -190.09419

Wald chi2(5) = 32.06
Prob > chi2 = 0.0000
```

| choice     | Coef.              | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|------------|--------------------|-----------|-------|-------|----------------------|-----------|
| mode       |                    |           |       |       |                      |           |
| travelcost | -.0097707          | .0027835  | -3.51 | 0.000 | -.0152261            | -.0043152 |
| termtime   | -.0377034          | .0094046  | -4.01 | 0.000 | -.0561361            | -.0192708 |
| air        | (base alternative) |           |       |       |                      |           |
| train      |                    |           |       |       |                      |           |
| income     | -.0291886          | .0089232  | -3.27 | 0.001 | -.0466778            | -.0116995 |
| _cons      | .5615485           | .394619   | 1.42  | 0.155 | -.2118906            | 1.334988  |
| bus        |                    |           |       |       |                      |           |
| income     | -.0127473          | .0079269  | -1.61 | 0.108 | -.0282839            | .0027892  |
| _cons      | -.0572738          | .4791635  | -0.12 | 0.905 | -.9964169            | .8818693  |
| car        |                    |           |       |       |                      |           |
| income     | -.0049067          | .0077481  | -0.63 | 0.527 | -.0200927            | .0102792  |
| _cons      | -1.833159          | .81842    | -2.24 | 0.025 | -3.437233            | -.2290856 |
| /ln12_2    | -.5499745          | .3903368  | -1.41 | 0.159 | -1.315021            | .2150717  |
| /ln13_3    | -.6008993          | .3354232  | -1.79 | 0.073 | -1.258317            | .056518   |
| /12_1      | 1.131589           | .2125186  | 5.32  | 0.000 | .7150604             | 1.548118  |
| /13_1      | .9720683           | .2352248  | 4.13  | 0.000 | .5110362             | 1.4331    |
| /13_2      | .5196988           | .2860692  | 1.82  | 0.069 | -.0409865            | 1.080384  |

```
(mode=air is the alternative normalizing location)
(mode=train is the alternative normalizing scale)

. estimates store full
```

By default, the differenced covariance parameterization is used, so the covariance matrix for this model is  $3 \times 3$ . There are two free variances to estimate and three correlations. To help ensure that the covariance matrix remains positive definite, `asmprobit` uses the square root transformation, where it optimizes on the Cholesky-factored variance–covariance. To ensure that the diagonal elements of the Cholesky estimates remain positive, we use the log transformation. The estimates labeled `/ln12_2` and `/ln13_3` in the coefficient table are the log-transformed diagonal elements of the Cholesky matrix. The estimates labeled `/12_1`, `/13_1`, and `/13_2` are the off-diagonal entries for elements (2, 1), (3, 1), and (3, 2) of the Cholesky matrix.

Although the transformed parameters of the differenced covariance parameterization are difficult to interpret, you can view them untransformed by using the `estat` command. Typing

```
. estat correlation
```

|       | train  | bus    | car    |
|-------|--------|--------|--------|
| train | 1.0000 |        |        |
| bus   | 0.8909 | 1.0000 |        |
| car   | 0.7896 | 0.8952 | 1.0000 |

Note: correlations are for alternatives differenced with air

gives the correlations, and typing

```
. estat covariance
```

|       | train    | bus      | car      |
|-------|----------|----------|----------|
| train | 2        |          |          |
| bus   | 1.600309 | 1.613382 |          |
| car   | 1.374712 | 1.39983  | 1.515656 |

Note: covariances are for alternatives differenced with air

gives the (co)variances.

We can reduce the number of covariance parameters in the model by using the factor model by [Cameron and Trivedi \(2005\)](#). For large models with many alternatives, the parameter reduction can be dramatic, but for our example we will use `factor(1)`, a one-dimension factor model, to reduce by 3 the number of parameters associated with the covariance matrix.

*(Continued on next page)*

```
. asmprobit choice travelcost termtime, casevars(income) case(id)
> alternatives(mode) factor(1)

(output omitted)

Alternative-specific multinomial probit Number of obs = 840
Case variable: id Number of cases = 210
Alternative variable: mode Alts per case: min = 4
 avg = 4.0
 max = 4

Integration sequence: Hammersley
Integration points: 200
Log simulated-likelihood = -196.85094 Wald chi2(5) = 107.85
 Prob > chi2 = 0.0000
```

| choice     | Coef.              | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|------------|--------------------|-----------|-------|-------|----------------------|-----------|
| mode       |                    |           |       |       |                      |           |
| travelcost | -.0093696          | .0036329  | -2.58 | 0.010 | -.01649              | -.0022492 |
| termtime   | -.0593173          | .0064585  | -9.18 | 0.000 | -.0719757            | -.0466589 |
| air        | (base alternative) |           |       |       |                      |           |
| train      |                    |           |       |       |                      |           |
| income     | -.0373511          | .0098219  | -3.80 | 0.000 | -.0566018            | -.0181004 |
| _cons      | .1092322           | .3949529  | 0.28  | 0.782 | -.6648613            | .8833257  |
| bus        |                    |           |       |       |                      |           |
| income     | -.0158793          | .0112239  | -1.41 | 0.157 | -.0378777            | .0061191  |
| _cons      | -1.082181          | .4678732  | -2.31 | 0.021 | -1.999196            | -.1651666 |
| car        |                    |           |       |       |                      |           |
| income     | .0042677           | .0092601  | 0.46  | 0.645 | -.0138817            | .0224171  |
| _cons      | -3.765445          | .5540636  | -6.80 | 0.000 | -4.851389            | -2.6795   |
| /c1_2      | 1.182805           | .3060299  | 3.86  | 0.000 | .5829972             | 1.782612  |
| /c1_3      | 1.227705           | .3401237  | 3.61  | 0.000 | .5610747             | 1.894335  |

(mode=air is the alternative normalizing location)  
(mode=train is the alternative normalizing scale)

The estimates labeled /c1\_2 and /c1\_3 in the coefficient table are the factor loadings. These factor loadings produce the following differenced covariance estimates:

```
. estat covariance
```

|       | train    | bus      | car      |
|-------|----------|----------|----------|
| train | 2        |          |          |
| bus   | 1.182805 | 2.399027 |          |
| car   | 1.227705 | 1.452135 | 2.507259 |

Note: covariances are for alternatives differenced with air



## Variance structures

The matrix  $\Omega$  has  $J(J+1)/2$  distinct elements because it is symmetric. Selecting a base alternative, normalizing its error variance to one, and constraining the correlations between its error and the other errors reduces the number of estimable parameters by  $J$ . Moreover, selecting a scale alternative and normalizing its error variance to one reduces the number by one, as well. Hence, there are at most  $m = J(J-1)/2 - 1$  estimable parameters in  $\Omega$ .



In practice, estimating all  $m$  parameters can be difficult, so one must often place more restrictions on the parameters. The `asmprobit` command provides the `correlation()` option to specify restrictions on the  $J(J-3)/2 + 1$  correlation parameters not already restricted as a result of choosing the base alternatives, and it provides `stddev()` to specify restrictions on the  $J-2$  standard deviations not already restricted as a result of choosing the base and scale alternatives.

When the `structural` option is used, `asmprobit` fits the model by assuming that all  $m$  parameters can be estimated, which is equivalent to specifying `correlation(unstructured)` and `stddev(heteroskedastic)`. The unstructured correlation structure means that all  $J(J-3)/2 + 1$  of the remaining correlation parameters will be estimated, and the heteroskedastic specification means that all  $J-2$  standard deviations will be estimated. With these default settings, the log likelihood is maximized with respect to the Cholesky decomposition of  $\Omega$ , and then the parameters are transformed to the standard deviation and correlation form.

The `correlation(exchangeable)` option forces the  $J(J-3)/2 + 1$  correlation parameters to be equal, and `correlation(independent)` forces all the correlations to be zero. Using the `stddev(homoskedastic)` option forces all  $J$  standard deviations to be one. These options may help in obtaining convergence for a model if the default options do not produce satisfactory results. In fact, when fitting a complex model, it may be advantageous to first fit a simple one and then proceed with removing the restrictions one at a time.

Advanced users may wish to specify alternative variance structures of their own choosing, and the next few paragraphs explain how to do so.

`correlation(pattern matname)` allows you to give the name of a  $J \times J$  matrix that identifies a correlation structure. Sequential positive integers starting at 1 are used to identify each correlation parameter: if there are three correlation parameters, they are identified by 1, 2, and 3. The integers can be repeated to indicate that correlations with the same number should be constrained to be equal. A zero or a missing value (.) indicates that the correlation is to be set to zero. `asmprobit` considers only the elements of the matrix *below* the main diagonal.

Suppose that you have a model with four alternatives, numbered 1–4, and alternative 1 is the base. The unstructured and exchangeable correlation structures identified in the  $4 \times 4$  lower triangular matrices are

|   | unstructured | exchangeable  |
|---|--------------|---------------|
|   | 1 2 3 4      | 1 2 3 4       |
| 1 | ( . )        | 1 ( . )       |
| 2 | ( 0 . )      | 2 ( 0 . )     |
| 3 | ( 0 1 . )    | 3 ( 0 1 . )   |
| 4 | ( 0 2 3 . )  | 4 ( 0 1 1 . ) |

`asmprobit` labels these correlation structures unstructured and exchangeable, even though the correlations corresponding to the base alternative are set to zero. More formally: these terms are appropriate when considering the  $(J-1) \times (J-1)$  submatrix  $\Sigma_{(k)}$  defined in the [Introduction](#) above.

You can also use the `correlation(fixed matname)` option to specify a matrix that specifies fixed and free parameters. Here the free parameters (those that are to be estimated) are identified by a missing value, and nonmissing values represent correlations that are to be taken as given. Below is a correlation structure that would set the correlations of alternative 1 to be 0.5:

|   | 1             | 2 | 3 | 4 |
|---|---------------|---|---|---|
| 1 | ( . )         |   |   |   |
| 2 | ( 0.5 . )     |   |   |   |
| 3 | ( 0.5 . . )   |   |   |   |
| 4 | ( 0.5 . . . ) |   |   |   |

The order of the elements of the `pattern` or `fixed` matrices must be the same as the numeric order of the alternative levels.

To specify the structure of the standard deviations—the diagonal elements of  $\Omega$ —you can use the `stddev(pattern matname)` option, where *matname* is a  $1 \times J$  matrix. Sequential positive integers starting at 1 are used to identify each standard deviation parameter. The integers can be repeated to indicate that standard deviations with the same number are to be constrained to be equal. A missing value indicates that the corresponding standard deviation is to be set to one. In the four-alternative example mentioned above, suppose that you wish to set the first and second standard deviations to one and that you wish to constrain the third and fourth standard deviations to be equal; the following `pattern` matrix will do that:

$$\begin{array}{cccc} & 1 & 2 & 3 & 4 \\ 1 & (\cdot & \cdot & 1 & 1) \end{array}$$

Using the `stddev(fixed matname)` option allows you to identify the fixed and free standard deviations. Fixed standard deviations are entered as positive real numbers, and free parameters are identified with missing values. For example, to constrain the first and second standard deviations to equal one and to allow the third and fourth to be estimated, you would use this `fixed` matrix:

$$\begin{array}{cccc} & 1 & 2 & 3 & 4 \\ 1 & (1 & 1 & \cdot & \cdot) \end{array}$$

When supplying either the `pattern` or the `fixed` matrices, you must ensure that the model is properly scaled. At least two standard deviations must be constant for the model to be scaled. A warning is issued if `asmprobbit` detects that the model is not scaled.

The order of the elements of the `pattern` or `fixed` matrices must be the same as the numeric order of the alternative levels.

## ► Example 2

In [example 1](#), we used the differenced covariance parameterization, the default. We now use the `structural` option to view the  $J - 2$  standard deviation estimates and the  $(J - 1)(J - 2)/2$  correlation estimates. Here we will fix the standard deviations for the `air` and `train` alternatives to 1 and the correlations between `air` and the rest of the alternatives to 0.

```
. asmprobit choice travelcost termtime, casevars(income) case(id)
> alternatives(mode) structural
(output omitted)
Alternative-specific multinomial probit Number of obs = 840
Case variable: id Number of cases = 210
Alternative variable: mode Alts per case: min = 4
 avg = 4.0
 max = 4
Integration sequence: Hammersley
Integration points: 200
Log simulated-likelihood = -190.09418 Wald chi2(5) = 32.05
 Prob > chi2 = 0.0000
```

| choice     | Coef.                 | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|------------|-----------------------|-----------|-------|-------|----------------------|-----------|
| mode       |                       |           |       |       |                      |           |
| travelcost | -.0097703             | .0027834  | -3.51 | 0.000 | -.0152257            | -.0043149 |
| termtime   | -.0377103             | .0094092  | -4.01 | 0.000 | -.056152             | -.0192687 |
| air        | (base alternative)    |           |       |       |                      |           |
| train      |                       |           |       |       |                      |           |
| income     | -.0291975             | .0089246  | -3.27 | 0.001 | -.0466895            | -.0117055 |
| _cons      | .5616448              | .3946529  | 1.42  | 0.155 | -.2118607            | 1.33515   |
| bus        |                       |           |       |       |                      |           |
| income     | -.01275               | .0079266  | -1.61 | 0.108 | -.0282858            | .0027858  |
| _cons      | -.0571664             | .4791996  | -0.12 | 0.905 | -.9963803            | .8820476  |
| car        |                       |           |       |       |                      |           |
| income     | -.0049085             | .0077486  | -0.63 | 0.526 | -.0200955            | .0102785  |
| _cons      | -1.833444             | .8186343  | -2.24 | 0.025 | -3.437938            | -.22895   |
| /lnsigma3  | -.2447428             | .4953363  | -0.49 | 0.621 | -1.215584            | .7260985  |
| /lnsigma4  | -.3309429             | .6494493  | -0.51 | 0.610 | -1.60384             | .9419543  |
| /atanhr3_2 | 1.01193               | .3890994  | 2.60  | 0.009 | .249309              | 1.774551  |
| /atanhr4_2 | .5786576              | .3940461  | 1.47  | 0.142 | -.1936586            | 1.350974  |
| /atanhr4_3 | .8885204              | .5600561  | 1.59  | 0.113 | -.2091693            | 1.98621   |
| sigma1     | 1 (base alternative)  |           |       |       |                      |           |
| sigma2     | 1 (scale alternative) |           |       |       |                      |           |
| sigma3     | .7829059              | .3878017  |       |       | .2965368             | 2.067     |
| sigma4     | .7182462              | .4664645  |       |       | .2011227             | 2.564989  |
| rho3_2     | .766559               | .1604596  |       |       | .244269              | .9441061  |
| rho4_2     | .5216891              | .2868027  |       |       | -.1912734            | .874283   |
| rho4_3     | .7106622              | .277205   |       |       | -.2061713            | .9630403  |

(mode=air is the alternative normalizing location)

(mode=train is the alternative normalizing scale)

When comparing this output to that of [example 1](#), we see that we have achieved the same log likelihood. That is, the structural parameterization using `air` as the base alternative and `train` as the scale alternative applied no restrictions on the model. This will not always be the case. We leave it up to you to try different base and scale alternatives, and you will see that not all the different combinations will achieve the same log likelihood. This is not true for the differenced covariance parameterization: it will always achieve the same log likelihood (and the maximum possible likelihood) regardless of the base and scale alternatives. This is why it is the default parameterization.

For an exercise, we can compute the differenced covariance displayed in [example 1](#) by using the following ado-code.

```
. estat covariance
```

|       | air | train    | bus      | car      |
|-------|-----|----------|----------|----------|
| air   | 1   |          |          |          |
| train | 0   | 1        |          |          |
| bus   | 0   | .6001436 | .6129416 |          |
| car   | 0   | .3747012 | .399619  | .5158776 |

```
. return list
matrices:
 r(cov) : 4 x 4

. matrix cov = r(cov)
. matrix M = (1,-1,0,0 \ 1,0,-1,0 \ 1,0,0,-1)
. matrix cov1 = M*cov*M'
. matrix list cov1
symmetric cov1[3,3]
 r1 r2 r3
r1 2
r2 1.6001436 1.6129416
r3 1.3747012 1.399619 1.5158776
```

The slight difference in the regression coefficients between the [example 1](#) and [example 2](#) coefficient tables reflects the accuracy of the [\[M-5\] ghk\(\)](#) algorithm using 200 points from the Hammersley sequence.

We now fit the model using the exchangeable correlation matrix and compare the models with a likelihood-ratio test.

```
. asmprobit choice travelcost termtime, casevars(income) case(id)
> alternatives(mode) correlation(exchangeable)
(output omitted)
Alternative-specific multinomial probit Number of obs = 840
Case variable: id Number of cases = 210
Alternative variable: mode Alts per case: min = 4
 avg = 4.0
 max = 4
Integration sequence: Hammersley
Integration points: 200
Log simulated-likelihood = -190.4679 Wald chi2(5) = 53.60
 Prob > chi2 = 0.0000
```

| choice     | Coef.                 | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|------------|-----------------------|-----------|-------|-------|----------------------|-----------|
| mode       |                       |           |       |       |                      |           |
| travelcost | -.0084636             | .0020452  | -4.14 | 0.000 | -.012472             | -.0044551 |
| termtime   | -.0345394             | .0072812  | -4.74 | 0.000 | -.0488103            | -.0202684 |
| air        | (base alternative)    |           |       |       |                      |           |
| train      |                       |           |       |       |                      |           |
| income     | -.0290357             | .0083226  | -3.49 | 0.000 | -.0453477            | -.0127237 |
| _cons      | .5517445              | .3719913  | 1.48  | 0.138 | -.177345             | 1.280834  |
| bus        |                       |           |       |       |                      |           |
| income     | -.0132562             | .0074133  | -1.79 | 0.074 | -.0277859            | .0012735  |
| _cons      | -.0052517             | .4337932  | -0.01 | 0.990 | -.8554708            | .8449673  |
| car        |                       |           |       |       |                      |           |
| income     | -.0060878             | .006638   | -0.92 | 0.359 | -.0190981            | .0069224  |
| _cons      | -1.565918             | .6633007  | -2.36 | 0.018 | -2.865964            | -.265873  |
| /lnsigmaP1 | -.3557589             | .1972809  | -1.80 | 0.071 | -.7424222            | .0309045  |
| /lnsigmaP2 | -1.308596             | .8872957  | -1.47 | 0.140 | -3.047663            | .4304719  |
| /atanhrP1  | 1.116589              | .3765488  | 2.97  | 0.003 | .3785667             | 1.854611  |
| sigma1     | 1 (base alternative)  |           |       |       |                      |           |
| sigma2     | 1 (scale alternative) |           |       |       |                      |           |
| sigma3     | .7006416              | .1382232  |       |       | .4759596             | 1.031387  |
| sigma4     | .2701992              | .2397466  |       |       | .0474697             | 1.537983  |
| rho3_2     | .8063791              | .131699   |       |       | .3614621             | .9521783  |
| rho4_2     | .8063791              | .131699   |       |       | .3614621             | .9521783  |
| rho4_3     | .8063791              | .131699   |       |       | .3614621             | .9521783  |

(mode=air is the alternative normalizing location)

(mode=train is the alternative normalizing scale)

```
. lrtest full .
```

```
Likelihood-ratio test LR chi2(2) = 0.75
(Assumption: . nested in full) Prob > chi2 = 0.6882
```

The likelihood-ratio test suggests that a common correlation is a plausible hypothesis, but this could be an artifact of the small sample size. The labeling of the standard deviation and correlation estimates has changed from `/lnsigma` and `/atanhr`, in the previous example, to `/lnsigmaP` and `/atanhrP`. The “P” identifies the parameter’s index in the pattern matrices used by `asmprobit`. The pattern matrices are saved in `e(stdpattern)` and `e(corpattern)`.



| choice     | Coef.              | Std. Err.           | z     | P> z  | [95% Conf. Interval] |           |
|------------|--------------------|---------------------|-------|-------|----------------------|-----------|
| mode       |                    |                     |       |       |                      |           |
| travelcost | -.0100335          | .0026203            | -3.83 | 0.000 | -.0151692            | -.0048979 |
| termtime   | -.0385731          | .008608             | -4.48 | 0.000 | -.0554445            | -.0217018 |
| air        | (base alternative) |                     |       |       |                      |           |
| train      |                    |                     |       |       |                      |           |
| income     | -.029271           | .0089739            | -3.26 | 0.001 | -.0468595            | -.0116824 |
| _cons      | .56528             | .4008037            | 1.41  | 0.158 | -.2202809            | 1.350841  |
| bus        |                    |                     |       |       |                      |           |
| income     | -.0124658          | .0080043            | -1.56 | 0.119 | -.0281539            | .0032223  |
| _cons      | -.0741685          | .4763422            | -0.16 | 0.876 | -1.007782            | .859445   |
| car        |                    |                     |       |       |                      |           |
| income     | -.0046905          | .0079934            | -0.59 | 0.557 | -.0203573            | .0109763  |
| _cons      | -1.897931          | .7912106            | -2.40 | 0.016 | -3.448675            | -.3471867 |
| /lnsigmaP1 | -.197697           | .2751269            | -0.72 | 0.472 | -.7369359            | .3415418  |
| /atanhrP1  | .9704403           | .3286981            | 2.95  | 0.003 | .3262038             | 1.614677  |
| /atanhrP2  | .5830923           | .3690419            | 1.58  | 0.114 | -.1402165            | 1.306401  |
| sigma1     | 1                  | (base alternative)  |       |       |                      |           |
| sigma2     | 1                  | (scale alternative) |       |       |                      |           |
| sigma3     | .8206185           | .2257742            |       |       | .4785781             | 1.407115  |
| sigma4     | .8206185           | .2257742            |       |       | .4785781             | 1.407115  |
| rho3_2     | .7488977           | .1443485            |       |       | .3151056             | .9238482  |
| rho4_2     | .5249094           | .2673598            |       |       | -.1393048            | .863362   |
| rho4_3     | .7488977           | .1443485            |       |       | .3151056             | .9238482  |

(mode=air is the alternative normalizing location)

(mode=train is the alternative normalizing scale)

In the call to `asmprobit`, we did not need to specify the `basealternative()` and `scalealternative()` options because they are implied by the specifications of the pattern matrices.

◀

## □ Technical note

If you experience convergence problems, try specifying `nopivot`, increasing `intpoints()`, specifying `antithetics`, specifying `technique(nr)` with `difficult`, or specifying a switching algorithm in the `technique()` option. As a last resort, you can use the `nrtolerance()` and `showtolerance` options. Changing the base and scale alternative in the model specification can also affect convergence if the `structural` option is used.

Because simulation methods are used to obtain multivariate normal probabilities, the estimates obtained have a limited degree of precision. Moreover, the solutions are particularly sensitive to the starting values used. Experimenting with different starting values may help in obtaining convergence, and doing so is a good way to verify previous results.

If you wish to use the BHHH algorithm along with another maximization algorithm, you must specify the `initbhhh(#)` option, where `#` is the number of BHHH iterations to use before switching to the algorithm specified in `technique()`. The BHHH algorithm uses an outer-product-of-gradients

approximation for the Hessian, and `asmprobit` must perform the gradient calculations differently than for the other algorithms.



## □ Technical note

If there are no alternative-specific variables in your model, the variance–covariance matrix parameters are not identifiable. For such a model to converge, you would therefore need to use `correlation(independent)` and `stddev(homoskedastic)`. A better alternative is to use `mprobit`, which is geared specifically toward models with only case-specific variables. See [R] [mprobit](#).



## Saved results

`asmprobit` saves the following in `e()`:

### Scalars

|                                |                                                |
|--------------------------------|------------------------------------------------|
| <code>e(N)</code>              | number of observations                         |
| <code>e(N_case)</code>         | number of cases                                |
| <code>e(k)</code>              | number of parameters                           |
| <code>e(k_alt)</code>          | number of alternatives                         |
| <code>e(k_indvars)</code>      | number of alternative-specific variables       |
| <code>e(k_casevars)</code>     | number of case-specific variables              |
| <code>e(k_sigma)</code>        | number of variance estimates                   |
| <code>e(k_rho)</code>          | number of correlation estimates                |
| <code>e(k_eq)</code>           | number of equations in <code>e(b)</code>       |
| <code>e(k_eq_model)</code>     | number of equations in model Wald test         |
| <code>e(k_autoCns)</code>      | number of base, empty, and omitted constraints |
| <code>e(df_m)</code>           | model degrees of freedom                       |
| <code>e(ll)</code>             | log simulated-likelihood                       |
| <code>e(N_clust)</code>        | number of clusters                             |
| <code>e(const)</code>          | constant indicator                             |
| <code>e(i_base)</code>         | base alternative index                         |
| <code>e(i_scale)</code>        | scale alternative index                        |
| <code>e(mc_points)</code>      | number of Monte Carlo replications             |
| <code>e(mc_burn)</code>        | starting sequence index                        |
| <code>e(mc_antithetics)</code> | antithetics indicator                          |
| <code>e(chi2)</code>           | $\chi^2$                                       |
| <code>e(p)</code>              | significance                                   |
| <code>e(fullcov)</code>        | unstructured covariance indicator              |
| <code>e(structcov)</code>      | 1 if structured covariance; 0 otherwise        |
| <code>e(cholesky)</code>       | Cholesky-factored covariance indicator         |
| <code>e(alt_min)</code>        | minimum number of alternatives                 |
| <code>e(alt_avg)</code>        | average number of alternatives                 |
| <code>e(alt_max)</code>        | maximum number of alternatives                 |
| <code>e(rank)</code>           | rank of <code>e(V)</code>                      |
| <code>e(ic)</code>             | number of iterations                           |
| <code>e(rc)</code>             | return code                                    |
| <code>e(converged)</code>      | 1 if converged, 0 otherwise                    |



## Macros

|                      |                                                                          |
|----------------------|--------------------------------------------------------------------------|
| e(cmd)               | asmpbit                                                                  |
| e(cmdline)           | command as typed                                                         |
| e(depvar)            | name of dependent variable                                               |
| e(indvars)           | alternative-specific independent variable                                |
| e(casevars)          | case-specific variables                                                  |
| e(case)              | variable defining cases                                                  |
| e(altvar)            | variable defining alternatives                                           |
| e(alteqs)            | alternative equation names                                               |
| e(alt#)              | alternative labels                                                       |
| e(wtype)             | weight type                                                              |
| e(wexp)              | weight expression                                                        |
| e(title)             | title in estimation output                                               |
| e(clustvar)          | name of cluster variable                                                 |
| e(correlation)       | correlation structure                                                    |
| e(stddev)            | variance structure                                                       |
| e(cov_class)         | class of the covariance structure                                        |
| e(chi2type)          | Wald, type of model $\chi^2$ test                                        |
| e(vce)               | <i>vcetype</i> specified in <i>vce()</i>                                 |
| e(vcetype)           | title used to label Std. Err.                                            |
| e(opt)               | type of optimization                                                     |
| e(which)             | max or min; whether optimizer is to perform maximization or minimization |
| e(ml_method)         | type of ml method                                                        |
| e(mc_method)         | technique used to generate sequences                                     |
| e(mc_seed)           | random-number generator seed                                             |
| e(user)              | name of likelihood-evaluator program                                     |
| e(technique)         | maximization technique                                                   |
| e(singularHmethod)   | m-marquardt or hybrid; method used when Hessian is singular              |
| e(crittype)          | optimization criterion                                                   |
| e(datasignature)     | the checksum                                                             |
| e(datasignaturevars) | variables used in calculation of checksum                                |
| e(properties)        | b V                                                                      |
| e(estat_cmd)         | program used to implement estat                                          |
| e(mfx_dlg)           | program used to implement estat mfx dialog                               |
| e(predict)           | program used to implement predict                                        |
| e(marginsnotok)      | predictions disallowed by margins                                        |

(Continued on next page)

## Matrices

|                              |                                                                                            |
|------------------------------|--------------------------------------------------------------------------------------------|
| <code>e(b)</code>            | coefficient vector                                                                         |
| <code>e(Cns)</code>          | constraints matrix                                                                         |
| <code>e(stats)</code>        | alternative statistics                                                                     |
| <code>e(stdpattern)</code>   | variance pattern                                                                           |
| <code>e(stdfixed)</code>     | fixed and free standard deviations                                                         |
| <code>e(altvals)</code>      | alternative values                                                                         |
| <code>e(altfreq)</code>      | alternative frequencies                                                                    |
| <code>e(alt_casevars)</code> | indicators for estimated case-specific coefficients— <code>e(k_alt) × e(k_casevars)</code> |
| <code>e(corpattern)</code>   | correlation structure                                                                      |
| <code>e(corfixed)</code>     | fixed and free correlations                                                                |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)                                                        |
| <code>e(gradient)</code>     | gradient vector                                                                            |
| <code>e(V)</code>            | variance–covariance matrix of the estimators                                               |
| <code>e(V_modelbased)</code> | model-based variance                                                                       |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`asmpbrot` is implemented as an ado-file.

The simulated maximum likelihood estimates for the MNP are obtained using `ml`; see [R] [ml](#). The likelihood evaluator implements the GHK algorithm to approximate the multivariate distribution function (Geweke 1989; Hajivassiliou and McFadden 1998; Keane and Wolpin 1994). The technique is also described in detail by Genz (1992), but Genz describes a more general algorithm where both lower and upper bounds of integration are finite. We briefly describe the GHK simulator and refer you to Bolduc (1999) for the score computations.

As discussed earlier, the latent variables for a  $J$ -alternative model are  $\eta_{ij} = \mathbf{x}_{ij}\beta + \mathbf{z}_i\alpha_j + \xi_{ij}$ , for  $j = 1, \dots, J$ ,  $i = 1, \dots, n$ , and  $\xi'_i = (\xi_{i,1}, \dots, \xi_{i,J}) \sim \text{MVN}(\mathbf{0}, \mathbf{\Omega})$ . The experimenter observes alternative  $k$  for the  $i$ th observation if  $k = \arg \max(\eta_{ij}, j = 1, \dots, J)$ . Let

$$\begin{aligned}
 v_{ij'} &= \eta_{ij} - \eta_{ik} \\
 &= (\mathbf{x}_{ij} - \mathbf{x}_{ik})\beta + \mathbf{z}_i(\alpha_j - \alpha_k) + \xi_{ij} - \xi_{ik} \\
 &= \delta_{ij'}\beta + \mathbf{z}_i\gamma_{j'} + \epsilon_{ij'}
 \end{aligned}$$

where  $j' = j$  if  $j < k$  and  $j' = j - 1$  if  $j > k$ , so that  $j' = 1, \dots, J - 1$ . Further,  $\epsilon_i = (\epsilon_{i1}, \dots, \epsilon_{i,J-1}) \sim \text{MVN}(\mathbf{0}, \mathbf{\Sigma}_{(k)})$ .  $\mathbf{\Sigma}$  is indexed by  $k$  because it depends on the choice made. We denote the deterministic part of the model as  $\lambda_{ij'} = \delta_{ij'}\beta + \mathbf{z}_i\gamma_{j'}$ , and the probability of this event is

$$\begin{aligned}
 \Pr(y_i = k) &= \Pr(v_{i1} \leq 0, \dots, v_{i,J-1} \leq 0) \\
 &= \Pr(\epsilon_{i1} \leq -\lambda_{i1}, \dots, \epsilon_{i,J-1} \leq -\lambda_{i,J-1}) \\
 &= (2\pi)^{-(J-1)/2} |\mathbf{\Sigma}_{(k)}|^{-1/2} \int_{-\infty}^{-\lambda_{i1}} \dots \int_{-\infty}^{-\lambda_{i,J-1}} \exp\left(-\frac{1}{2}\mathbf{z}'\mathbf{\Sigma}_{(k)}^{-1}\mathbf{z}\right) d\mathbf{z}
 \end{aligned} \tag{3}$$

## Simulated likelihood

For clarity in the discussion that follows, we drop the index denoting case so that for an arbitrary observation  $v' = (v_1, \dots, v_{J-1})$ ,  $\lambda' = (\lambda_1, \dots, \lambda_{J-1})$ , and  $\epsilon' = (\epsilon_1, \dots, \epsilon_{J-1})$ .

The Cholesky-factored variance–covariance,  $\Sigma = \mathbf{L}\mathbf{L}'$ , is lower triangular,

$$\mathbf{L} = \begin{pmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & & \vdots \\ l_{J-1,1} & l_{J-1,2} & \dots & l_{J-1,J-1} \end{pmatrix}$$

and the correlated latent-variable errors can be expressed as linear functions of uncorrelated normal variates,  $\epsilon = \mathbf{L}\zeta$ , where  $\zeta' = (\zeta_1, \dots, \zeta_{J-1})$  and  $\zeta_j \sim \text{iid } N(0, 1)$ . We now have  $v = \lambda + \mathbf{L}\zeta$ , and by defining

$$z_j = \begin{cases} -\frac{\lambda_1}{l_{11}} & \text{for } j = 1 \\ -\frac{\lambda_j + \sum_{i=1}^{j-1} l_{ji}\zeta_i}{l_{jj}} & \text{for } j = 2, \dots, J-1 \end{cases} \quad (4)$$

we can express the probability statement (3) as the product of conditional probabilities

$$\begin{aligned} \Pr(y_i = k) &= \Pr(\zeta_1 \leq z_1) \Pr(\zeta_2 \leq z_2 \mid \zeta_1 \leq z_1) \dots \\ &\quad \Pr(\zeta_{J-1} \leq z_{J-1} \mid \zeta_1 \leq z_1, \dots, \zeta_{J-2} \leq z_{J-2}) \end{aligned}$$

because

$$\begin{aligned} \Pr(v_1 \leq 0) &= \Pr(\lambda_1 + l_{11}\zeta_1 \leq 0) \\ &= \Pr\left(\zeta_1 \leq -\frac{\lambda_1}{l_{11}}\right) \\ \Pr(v_2 \leq 0) &= \Pr(\lambda_2 + l_{21}\zeta_1 + l_{22}\zeta_2 \leq 0) \\ &= \Pr\left(\zeta_2 \leq -\frac{\lambda_2 + l_{21}\zeta_1}{l_{22}} \mid \zeta_1 \leq -\frac{\lambda_1}{l_{11}}\right) \\ &\dots \end{aligned}$$

The Monte Carlo algorithm then must make draws from the truncated standard normal distribution. It does so by generating  $J-1$  uniform variates,  $\delta_j, j = 1, \dots, J-1$ , and computing

$$\tilde{\zeta}_j = \begin{cases} \Phi^{-1}\left\{\delta_1 \Phi\left(-\frac{\lambda_1}{l_{11}}\right)\right\} & \text{for } j = 1 \\ \Phi^{-1}\left\{\delta_j \Phi\left(\frac{-\lambda_j - \sum_{i=1}^{j-1} l_{ji}\tilde{\zeta}_i}{l_{jj}}\right)\right\} & \text{for } j = 2, \dots, J-1 \end{cases}$$

Define  $\tilde{z}_j$  by replacing  $\tilde{\zeta}_i$  for  $\zeta_i$  in (4) so that the simulated probability for the  $l$ th draw is

$$p_l = \prod_{j=1}^{J-1} \Phi(\tilde{z}_j)$$

To increase accuracy, the bounds of integration,  $\lambda_j$ , are ordered so that the largest integration intervals are on the inside. The rows and columns of the variance–covariance matrix are pivoted accordingly (Genz 1992).

For a more detailed description of the GHK algorithm in Stata, see Gates (2006).

Repeated draws are made, say,  $N$ , and the simulated likelihood for the  $i$ th case, denoted  $\hat{L}_i$ , is computed as

$$\hat{L}_i = \frac{1}{N} \sum_{l=1}^N p_l$$

The overall simulated log likelihood is  $\sum_i \log \hat{L}_i$ .

If the true likelihood is  $L_i$ , the error bound on the approximation can be expressed as

$$|\hat{L}_i - L_i| \leq V(L_i) D_N\{(\delta_i)\}$$

where  $V(L_i)$  is the total variation of  $L_i$  and  $D_N$  is the discrepancy, or nonuniformity, of the set of abscissas. For the uniform pseudorandom sequence,  $\delta_i$ , the discrepancy is of order  $O\{(\log \log N/N)^{1/2}\}$ . The order of discrepancy can be improved by using quasirandom sequences.

Quasi–Monte Carlo integration is carried out by `asmprobbit` by replacing the uniform deviates with either the Halton or the Hammersley sequences. These sequences spread the points more evenly than the uniform random sequence and have a smaller order of discrepancy,  $O[\{(\log N)^{J-1}\}/N]$  and  $O[\{(\log N)^{J-2}\}/N]$ , respectively. The Halton sequence of dimension  $J-1$  is generated from the first  $J-1$  primes,  $p_k$ , so that on draw  $l$  we have  $\mathbf{h}_l = \{r_{p_1}(l), r_{p_2}(l), \dots, r_{p_{J-1}}(l)\}$ , where

$$r_{p_k}(l) = \sum_{j=0}^q b_{jk}(l) p_k^{-j-1} \in (0, 1)$$

is the radical inverse function of  $l$  with base  $p_k$  so that  $\sum_{j=0}^q b_{jk}(l) p_k^j = l$ , where  $p_k^q \leq l < p_k^{q+1}$  (Fang and Wang 1994).

This function is demonstrated with base  $p_3 = 5$  and  $l = 33$ , which generates  $r_5(33)$ . Here  $q = 2$ ,  $b_{0,3}(33) = 3$ ,  $b_{1,5}(33) = 1$ , and  $b_{2,5}(33) = 1$ , so that  $r_5(33) = 3/5 + 1/25 + 1/625$ .

The Hammersley sequence uses an evenly spaced set of points with the first  $J-2$  components of the Halton sequence

$$\mathbf{h}_l = \left\{ \frac{2l-1}{2N}, r_{p_1}(l), r_{p_2}(l), \dots, r_{p_{J-2}}(l) \right\}$$

for  $l = 1, \dots, N$ .

For a more detailed description of the Halton and Hammersley sequences, see Drukker and Gates (2006).

Computations for the derivatives of the simulated likelihood are taken from Bolduc (1999). Bolduc gives the analytical first-order derivatives for the log of the simulated likelihood with respect to the regression coefficients and the parameters of the Cholesky-factored variance–covariance matrix. `asmprobbit` uses these analytical first-order derivatives and numerical second-order derivatives.

This command supports the clustered version of the Huber/White/sandwich estimator of the variance using `vce(robust)` and `vce(cluster clustvar)`. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#). Specifying `vce(robust)` is equivalent to specifying `vce(cluster casevar)`, where `casevar` is the variable that identifies the cases.

# References

- Bolduc, D. 1999. A practical technique to estimate multinomial probit models in transportation. *Transportation Research Part B* 33: 63–79.
- Bunch, D. S. 1991. Estimability of the multinomial probit model. *Transportation Research Part B* 25: 1–12.
- Cameron, A. C., and P. K. Trivedi. 2005. *Microeconometrics: Methods and Applications*. New York: Cambridge University Press.
- Cappellari, L., and S. P. Jenkins. 2003. Multivariate probit regression using simulated maximum likelihood. *Stata Journal* 3: 278–294.
- Drukker, D. M., and R. Gates. 2006. Generating Halton sequences using Mata. *Stata Journal* 6: 214–228.
- Fang, K.-T., and Y. Wang. 1994. *Number-theoretic Methods in Statistics*. London: Chapman & Hall.
- Gates, R. 2006. A Mata Geweke–Hajivassiliou–Keane multivariate normal simulator. *Stata Journal* 6: 190–213.
- Genz, A. 1992. Numerical computation of multivariate normal probabilities. *Journal of Computational and Graphical Statistics* 1: 141–149.
- Geweke, J. 1989. Bayesian inference in econometric models using Monte Carlo integration. *Econometrica* 57: 1317–1339.
- Geweke, J., and M. P. Keane. 2001. Computationally intensive methods for integration in econometrics. In Vol. 5 of *Handbook of Econometrics*, ed. J. Heckman and E. Leamer, 3463–3568. Amsterdam: North-Holland.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice-Hall.
- Haan, P., and A. Uhlenborff. 2006. Estimation of multinomial logit models with unobserved heterogeneity using maximum simulated likelihood. *Stata Journal* 6: 229–245.
- Hajivassiliou, V. A., and D. L. McFadden. 1998. The method of simulated scores for the estimation of LDV models. *Econometrica* 66: 863–896.
- Hole, A. R. 2007. Fitting mixed logit models by using maximum simulated likelihood. *Stata Journal* 7: 388–401.
- Keane, M. P., and K. I. Wolpin. 1994. The solution and estimation of discrete choice dynamic programming models by simulation and interpolation: Monte Carlo evidence. *Review of Economics and Statistics* 76: 648–672.
- Train, K. E. 2003. *Discrete Choice Methods with Simulation*. Cambridge: Cambridge University Press.

# Also see

- [R] [asmprobit postestimation](#) — Postestimation tools for asmprobit
- [R] [asclogit](#) — Alternative-specific conditional logit (McFadden’s choice) model
- [R] [asroprobit](#) — Alternative-specific rank-ordered probit regression
- [R] [mprobit](#) — Multinomial probit regression
- [R] [mlogit](#) — Multinomial (polytomous) logistic regression
- [U] [20 Estimation and postestimation commands](#)

## Description

The following postestimation commands are of special interest after `asmpbfit`:

| command                         | description                                    |
|---------------------------------|------------------------------------------------|
| <code>estat alternatives</code> | alternative summary statistics                 |
| <code>estat covariance</code>   | variance–covariance matrix of the alternatives |
| <code>estat correlation</code>  | correlation matrix of the alternatives         |
| <code>estat facweights</code>   | covariance factor weights matrix               |
| <code>estat mfx</code>          | marginal effects                               |

For information about these commands, see below.

The following standard postestimation commands are also available:

| command                | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>     | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estimates</code> | cataloging estimation results                                                                       |
| <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code>    | likelihood-ratio test                                                                               |
| <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>   | predicted probabilities, estimated linear predictor and its standard error                          |
| <code>predictnl</code> | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details.

## Special-interest postestimation commands

`estat alternatives` displays summary statistics about the alternatives in the estimation sample and provides a mapping between the index numbers that label the covariance parameters of the model and their associated values and labels for the alternative variable.

`estat covariance` computes the estimated variance–covariance matrix for the alternatives. The estimates are displayed, and the variance–covariance matrix is stored in `r(cov)`.

`estat correlation` computes the estimated correlation matrix for the alternatives. The estimates are displayed, and the correlation matrix is stored in `r(cor)`.

`estat facweights` displays the covariance factor weights matrix and stores it in `r(C)`.

`estat mfx` computes the simulated probability marginal effects.

## Syntax for predict

```
predict [type] newvar [if] [in] [, statistic altwise]
```

```
predict [type] { stub* | newvarlist } [if] [in], scores
```

| statistic | description |
|-----------|-------------|
|-----------|-------------|

---

Main

|             |                                                |
|-------------|------------------------------------------------|
| <u>pr</u>   | probability alternative is chosen; the default |
| <u>xb</u>   | linear prediction                              |
| <u>stdp</u> | standard error of the linear prediction        |

---

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

---

Main

pr, the default, calculates the probability that alternative  $j$  is chosen in case  $i$ .

xb calculates the linear prediction  $\mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_i\boldsymbol{\alpha}_j$  for alternative  $j$  and case  $i$ .

stdp calculates the standard error of the linear predictor.

altwise specifies that alternative-wise deletion be used when marking out observations due to missing values in your variables. The default is to use casewise deletion. The xb and stdp options always use alternative-wise deletion.

scores calculates the scores for each coefficient in `e(b)`. This option requires a new variable list of length equal to the number of columns in `e(b)`. Otherwise, use the `stub*` option to have `predict` generate enumerated variables with prefix `stub`.

## Syntax for estat alternatives

```
estat alternatives
```

## Menu

Statistics > Postestimation > Reports and statistics

## Syntax for estat covariance

```
estat covariance [, format(string) border(string) left(#)]
```

## Menu

Statistics > Postestimation > Reports and statistics

## Options for estat covariance

`format(string)` sets the matrix display format. The default is `format(%9.0g)`.

`border(string)` sets the matrix display border style. The default is `border(all)`. See [\[P\] matlist](#).

`left(#)` sets the matrix display left indent. The default is `left(2)`. See [\[P\] matlist](#).

## Syntax for estat correlation

```
estat correlation [, format(string) border(string) left(#)]
```

## Menu

Statistics > Postestimation > Reports and statistics

## Options for estat correlation

`format(string)` sets the matrix display format. The default is `format(%9.4f)`.

`border(string)` sets the matrix display border style. The default is `border(all)`. See [\[P\] matlist](#).

`left(#)` sets the matrix display left indent. The default is `left(2)`. See [\[P\] matlist](#).

## Syntax for estat facweights

```
estat facweights [, format(string) border(string) left(#)]
```

## Menu

Statistics > Postestimation > Reports and statistics

## Options for estat facweights

`format(string)` sets the matrix display format. The default is `format(%9.0f)`.

`border(string)` sets the matrix display border style. The default is `border(all)`. See [\[P\] matlist](#).

`left(#)` sets the matrix display left indent. The default is `left(2)`. See [\[P\] matlist](#).

## Syntax for estat mfx

```
estat mfx [if] [in] [, options]
```



| options                                          | description                                                               |
|--------------------------------------------------|---------------------------------------------------------------------------|
| <b>Main</b>                                      |                                                                           |
| <code>varlist(varlist)</code>                    | display marginal effects for <i>varlist</i>                               |
| <code>at(mean [atlist]   median [atlist])</code> | calculate marginal effects at these values                                |
| <b>Options</b>                                   |                                                                           |
| <code>level(#)</code>                            | set confidence interval level; default is <code>level(95)</code>          |
| <code>nodiscrete</code>                          | treat indicator variables as continuous                                   |
| <code>noesample</code>                           | do not restrict calculation of means and medians to the estimation sample |
| <code>nowght</code>                              | ignore weights when calculating means and medians                         |

## Menu

Statistics > Postestimation > Reports and statistics

## Options for estat mfx

### Main

`varlist(varlist)` specifies the variables for which to display marginal effects. The default is all variables.

`at(mean [atlist] | median [atlist])` specifies the values at which the marginal effects are to be calculated. *atlist* is

```
[[alternative:variable = #] [variable = #] [...]]
```

The default is to calculate the marginal effects at the means of the independent variables at the estimation sample, `at(mean)`.

After specifying the summary statistic, you can specify a series of specific values for variables. You can specify values for alternative-specific variables by `alternative`, or you can specify one value for all alternatives. You can specify only one value for case-specific variables. For example, in the `travel` dataset, `income` is a case-specific variable, whereas `termtime` and `travelcost` are alternative-specific variables. The following would be a legal syntax for `estat mfx`:

```
. estat mfx, at(mean air:termtime=50 travelcost=100 income=60)
```

When `nodiscrete` is not specified, `at(mean [atlist])` or `at(median [atlist])` has no effect on computing marginal effects for indicator variables, which are calculated as the discrete change in the simulated probability as the indicator variable changes from 0 to 1.

The mean and median computations respect any `if` and `in` qualifiers, so you can restrict the data over which the means or medians are computed. You can even restrict the values to a specific case; e.g.,

```
. estat mfx if case==21
```

### Options

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see [\[U\] 20.7 Specifying the width of confidence intervals](#).

`nodiscrete` specifies that indicator variables be treated as continuous variables. An indicator variable is one that takes on the value 0 or 1 in the estimation sample. By default, the discrete change in the simulated probability is computed as the indicator variable changes from 0 to 1.

`noesample` specifies that the whole dataset be considered instead of only those marked in the `e(sample)` defined by the `asmpbprobit` command.

`nowght` specifies that weights be ignored when calculating the means or medians.

## Remarks

Remarks are presented under the following headings:

- [Predicted probabilities](#)
- [Obtaining estimation statistics](#)
- [Obtaining marginal effects](#)

## Predicted probabilities

After fitting an alternative-specific multinomial probit model, you can use `predict` to obtain the simulated probabilities that an individual will choose each of the alternatives. When evaluating the multivariate normal probabilities via Monte Carlo simulation, `predict` uses the same method to generate the random sequence of numbers as the previous call to `asmpbprobit`. For example, if you specified `intmethod(Halton)` when fitting the model, `predict` also uses the Halton sequence.

### ► Example 1

In [example 1](#) of [\[R\] asmpbprobit](#), we fit a model of individuals' travel-mode choices. We can obtain the simulated probabilities that an individual chooses each alternative by using `predict`:

```
. use http://www.stata-press.com/data/r11/travel
. asmpbprobit choice travelcost termtime, casevars(income) case(id)
> alternatives(mode)
(output omitted)
. predict prob
(option pr assumed; Pr(mode))
. list id mode prob choice in 1/12, sepby(id)
```

|     | id | mode  | prob     | choice |
|-----|----|-------|----------|--------|
| 1.  | 1  | air   | .1494373 | 0      |
| 2.  | 1  | train | .3292305 | 0      |
| 3.  | 1  | bus   | .1319851 | 0      |
| 4.  | 1  | car   | .3898138 | 1      |
| 5.  | 2  | air   | .2566063 | 0      |
| 6.  | 2  | train | .2761403 | 0      |
| 7.  | 2  | bus   | .0116148 | 0      |
| 8.  | 2  | car   | .4556371 | 1      |
| 9.  | 3  | air   | .2098737 | 0      |
| 10. | 3  | train | .1082147 | 0      |
| 11. | 3  | bus   | .1671145 | 0      |
| 12. | 3  | car   | .5147865 | 1      |

## Obtaining estimation statistics

Once you have fit a multinomial probit model, you can obtain the estimated variance or correlation matrices for the model alternatives by using the `estat` command.

### ► Example 2

To display the correlations of the errors in the latent-variable equations, we type

```
. estat correlation
```

|       | train  | bus    | car    |
|-------|--------|--------|--------|
| train | 1.0000 |        |        |
| bus   | 0.8909 | 1.0000 |        |
| car   | 0.7896 | 0.8952 | 1.0000 |

Note: correlations are for alternatives differenced with air

The covariance matrix can be displayed by typing

```
. estat covariance
```

|       | train    | bus      | car      |
|-------|----------|----------|----------|
| train | 2        |          |          |
| bus   | 1.600309 | 1.613382 |          |
| car   | 1.374712 | 1.39983  | 1.515656 |

Note: covariances are for alternatives differenced with air

◀

## Obtaining marginal effects

The marginal effects are computed as the derivative of the simulated probability for an alternative with respect to an independent variable. A table of marginal effects is displayed for each alternative, with the table containing the marginal effect for each case-specific variable and the alternative for each alternative-specific variable.

By default, the marginal effects are computed at the means of each continuous independent variable over the estimation sample. For indicator variables, the difference in the simulated probability evaluated at 0 and 1 is computed by default. Indicator variables will be treated as continuous variables if the `nodiscrete` option is used.

### ► Example 3

Continuing with our model from [example 1](#), we obtain the marginal effects for alternatives `air`, `train`, `bus`, and `car` evaluated at the mean values of each independent variable. Recall that the `travelcost` and `termtime` variables are alternative specific, taking on different values for each alternative, so they have a separate marginal effect for each alternative.

```
. estat mfx
Pr(choice = air) = .29437379
```

| variable   | dp/dx    | Std. Err. | z     | P> z  | [        | 95% C.I. | ] | X      |
|------------|----------|-----------|-------|-------|----------|----------|---|--------|
| travelcost |          |           |       |       |          |          |   |        |
| air        | -.002689 | .000677   | -3.97 | 0.000 | -.004015 | -.001362 |   | 102.65 |
| train      | .000901  | .000436   | 2.07  | 0.039 | .000046  | .001755  |   | 130.2  |
| bus        | .000376  | .000271   | 1.39  | 0.166 | -.000156 | .000907  |   | 115.26 |
| car        | .001412  | .00051    | 2.77  | 0.006 | .000412  | .002412  |   | 95.414 |
| termtime   |          |           |       |       |          |          |   |        |
| air        | -.010375 | .00271    | -3.83 | 0.000 | -.015687 | -.005063 |   | 61.01  |
| train      | .003475  | .001638   | 2.12  | 0.034 | .000265  | .006686  |   | 35.69  |
| bus        | .00145   | .001008   | 1.44  | 0.150 | -.000524 | .003425  |   | 41.657 |
| car        | .005449  | .002164   | 2.52  | 0.012 | .001207  | .00969   |   | 0      |
| casevars   |          |           |       |       |          |          |   |        |
| income     | .00389   | .001847   | 2.11  | 0.035 | .00027   | .00751   |   | 34.548 |

Pr(choice = train) = .29535865

| variable   | dp/dx    | Std. Err. | z     | P> z  | [        | 95% C.I. | ] | X      |
|------------|----------|-----------|-------|-------|----------|----------|---|--------|
| travelcost |          |           |       |       |          |          |   |        |
| air        | .000899  | .000436   | 2.06  | 0.039 | .000045  | .001753  |   | 102.65 |
| train      | -.004081 | .001466   | -2.78 | 0.005 | -.006954 | -.001209 |   | 130.2  |
| bus        | .001278  | .00063    | 2.03  | 0.042 | .000043  | .002513  |   | 115.26 |
| car        | .001904  | .000887   | 2.15  | 0.032 | .000166  | .003642  |   | 95.414 |
| termtime   |          |           |       |       |          |          |   |        |
| air        | .003469  | .001638   | 2.12  | 0.034 | .000259  | .00668   |   | 61.01  |
| train      | -.015749 | .00247    | -6.38 | 0.000 | -.020589 | -.010908 |   | 35.69  |
| bus        | .004932  | .001593   | 3.10  | 0.002 | .00181   | .008053  |   | 41.657 |
| car        | .007348  | .002229   | 3.30  | 0.001 | .00298   | .011716  |   | 0      |
| casevars   |          |           |       |       |          |          |   |        |
| income     | -.009568 | .002223   | -4.30 | 0.000 | -.013925 | -.005212 |   | 34.548 |

Pr(choice = bus) = .08879634

| variable   | dp/dx    | Std. Err. | z     | P> z  | [        | 95% C.I. | ] | X      |
|------------|----------|-----------|-------|-------|----------|----------|---|--------|
| travelcost |          |           |       |       |          |          |   |        |
| air        | .00038   | .000274   | 1.39  | 0.165 | -.000157 | .000916  |   | 102.65 |
| train      | .001279  | .00063    | 2.03  | 0.042 | .000044  | .002514  |   | 130.2  |
| bus        | -.003182 | .001174   | -2.71 | 0.007 | -.005484 | -.00088  |   | 115.26 |
| car        | .001524  | .000675   | 2.26  | 0.024 | .0002    | .002848  |   | 95.414 |
| termtime   |          |           |       |       |          |          |   |        |
| air        | .001465  | .001016   | 1.44  | 0.149 | -.000527 | .003457  |   | 61.01  |
| train      | .004935  | .001591   | 3.10  | 0.002 | .001817  | .008053  |   | 35.69  |
| bus        | -.01228  | .002803   | -4.38 | 0.000 | -.017774 | -.006786 |   | 41.657 |
| car        | .00588   | .002255   | 2.61  | 0.009 | .001461  | .010299  |   | 0      |
| casevars   |          |           |       |       |          |          |   |        |
| income     | .000434  | .001461   | 0.30  | 0.766 | -.002429 | .003297  |   | 34.548 |

Pr(choice = car) = .32161847

| variable   | dp/dx    | Std. Err. | z     | P> z  | [        | 95% C.I. | ] | X      |
|------------|----------|-----------|-------|-------|----------|----------|---|--------|
| travelcost |          |           |       |       |          |          |   |        |
| air        | .00141   | .00051    | 2.77  | 0.006 | .000411  | .002409  |   | 102.65 |
| train      | .001904  | .000886   | 2.15  | 0.032 | .000166  | .003641  |   | 130.2  |
| bus        | .001523  | .000676   | 2.25  | 0.024 | .000199  | .002848  |   | 115.26 |
| car        | -.004837 | .001539   | -3.14 | 0.002 | -.007854 | -.00182  |   | 95.414 |
| termtime   |          |           |       |       |          |          |   |        |
| air        | .005441  | .002162   | 2.52  | 0.012 | .001204  | .009678  |   | 61.01  |
| train      | .007347  | .002228   | 3.30  | 0.001 | .00298   | .011713  |   | 35.69  |
| bus        | .005879  | .002256   | 2.61  | 0.009 | .001456  | .010301  |   | 41.657 |
| car        | -.018666 | .003939   | -4.74 | 0.000 | -.026387 | -.010945 |   | 0      |
| casevars   |          |           |       |       |          |          |   |        |
| income     | .005246  | .002165   | 2.42  | 0.015 | .001002  | .00949   |   | 34.548 |

First, we note that there is a separate marginal effects table for each alternative and that table begins by reporting the overall probability of choosing the alternative, e.g., 0.2944 for air travel. We see in the first table that a unit increase in terminal time for air travel from 61.01 minutes will result in a decrease in probability of choosing air travel (when the probability is evaluated at the mean of all variables) by approximately 0.01, with a 95% confidence interval of about  $-0.016$  to  $-0.005$ . Travel cost has a less negative effect of choosing air travel (at the average cost of 102.65). Alternatively, an increase in terminal time and travel cost for train, bus, or car from these mean values will increase the chance for air travel to be chosen. Also, with an increase in income from 34.5, it would appear that an individual would be more likely to choose air or automobile travel over bus or train. (While the marginal effect for bus travel is positive, it is not significant.)

◀

## ► Example 4

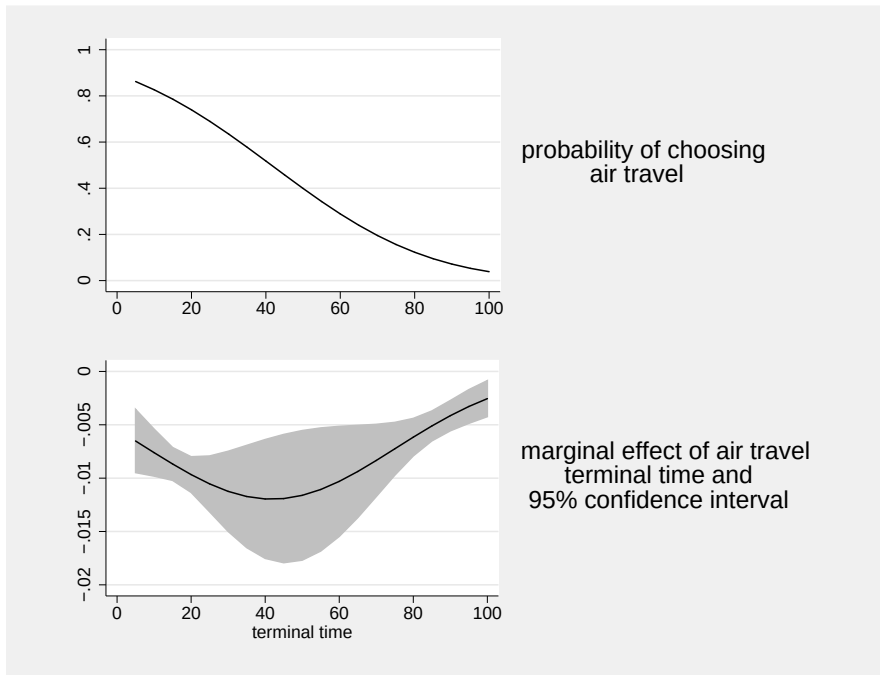
Plotting the simulated probability marginal effect evaluated over a range of values for an independent variable may be more revealing than a table of values. Below are the commands for generating the simulated probability marginal effect of air travel for increasing air travel terminal time. We fix all other independent variables at their medians.

```
. qui gen meff = .
. qui gen tt = .
. qui gen lb = .
. qui gen ub = .
. forvalues i=0/19 {
2. local termtime = 5+5*'i'
3. qui replace tt = 'termtime' if _n == 'i'+1
4. qui estat mfx, at(median air:termtime='termtime') var(termtime)
5. mat air = r(air)
6. qui replace meff = air[1,1] if _n == 'i'+1
7. qui replace lb = air[1,5] if _n == 'i'+1
8. qui replace ub = air[1,6] if _n == 'i'+1
9. qui replace prob = r(pr_air) if _n == 'i'+1
10. }
. label variable tt "terminal time"
```

```
. twoway (rarea lb ub tt, pstyle(ci)) (line meff tt, lpattern(solid)), name(meff)
> legend(off) title(" marginal effect of air travel" "terminal time and"
> "95% confidence interval", position(3))

. twoway line prob tt, name(prob) title(" probability of choosing" "air travel",
> position(3)) graphregion(margin(r+9)) ytitle("") xtitle("")

. graph combine prob meff, cols(1) graphregion(margin(l+5 r+5))
```



From the graphs, we see that the simulated probability of choosing air travel decreases in an sigmoid fashion. The marginal effects display the rate of change in the simulated probability as a function of the air travel terminal time. The rate of change in the probability of choosing air travel decreases until the air travel terminal time reaches about 45; thereafter, it increases.

◀

## Saved results

`estat mfx` saves the following in `r()`:

### Scalars

`r(pr_ult)` scalars containing the computed probability of each alternative evaluated at the value that is labeled `X` in the table output. Here `ult` are the labels in the macro `e(alteqs)`.

### Matrices

`r(alt)` matrices containing the computed marginal effects and associated statistics. There is one matrix for each alternative, where `alt` are the labels in the macro `e(alteqs)`. Column 1 of each matrix contains the marginal effects; column 2, their standard errors; columns 3 and 4, their  $z$  statistics and the  $p$ -values for the  $z$  statistics; and columns 5 and 6, the confidence intervals. Column 7 contains the values of the independent variables used to compute the probabilities `r(pr_ult)`.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

### Marginal effects

The marginal effects are computed as the derivative of the simulated probability with respect to each independent variable. A set of marginal effects is computed for each alternative; thus, for  $J$  alternatives, there will be  $J$  tables. Moreover, the alternative-specific variables will have  $J$  entries, one for each alternative in each table. The details of computing the effects are different for alternative-specific variables and case-specific variables, as well as for continuous and indicator variables.

We use the latent-variable notation of `asmprobit` (see [R] [asmprobit](#)) for a  $J$ -alternative model and, for notational convenience, we will drop any subscripts involving observations. We then have the following linear functions  $\eta_j = \mathbf{x}_j\beta + \mathbf{z}\alpha_j$ , for  $j = 1, \dots, J$ . Let  $k$  index the alternative of interest, and then

$$\begin{aligned} v_{j'} &= \eta_j - \eta_k \\ &= (\mathbf{x}_j - \mathbf{x}_k)\beta + \mathbf{z}(\alpha_j - \alpha_k) + \epsilon_{j'} \end{aligned}$$

where  $j' = j$  if  $j < k$  and  $j' = j - 1$  if  $j > k$ , so that  $j' = 1, \dots, J - 1$  and  $\epsilon_{j'} \sim \text{MVN}(\mathbf{0}, \Sigma)$ . Denote  $p_k = \Pr(v_1 \leq 0, \dots, v_{J-1} \leq 0)$  as the simulated probability of choosing alternative  $k$  given profile  $\mathbf{x}_k$  and  $\mathbf{z}$ . The marginal effects are then  $\partial p_k / \partial \mathbf{x}_k$ ,  $\partial p_k / \partial \mathbf{x}_j$ , and  $\partial p_k / \partial \mathbf{z}$ , where  $k = 1, \dots, J$ ,  $j \neq k$ . `asmprobit` analytically computes the first-order derivatives of the simulated probability with respect to the  $v$ 's, and the marginal effects for  $\mathbf{x}$ 's and  $\mathbf{z}$  are obtained via the chain rule. The standard errors for the marginal effects are computed using the delta method.

### Also see

[R] [asmprobit](#) — Alternative-specific multinomial probit regression

[U] [20 Estimation and postestimation commands](#)

## Syntax

```
asroprobit depvar [indepvars] [if] [in] [weight] , case(varname)

 alternatives(varname) [options]
```

| <i>options</i>                                         | description                                                                                                                                                                 |
|--------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                                  |                                                                                                                                                                             |
| * <u>case</u> ( <i>varname</i> )                       | use <i>varname</i> to identify cases                                                                                                                                        |
| * <u>alternatives</u> ( <i>varname</i> )               | use <i>varname</i> to identify the alternatives available for each case                                                                                                     |
| <u>casevars</u> ( <i>varlist</i> )                     | case-specific variables                                                                                                                                                     |
| <u>constraints</u> ( <i>constraints</i> )              | apply specified linear constraints                                                                                                                                          |
| <u>collinear</u>                                       | keep collinear variables                                                                                                                                                    |
| Model 2                                                |                                                                                                                                                                             |
| <u>correlation</u> ( <i>correlation</i> )              | correlation structure of the latent-variable errors                                                                                                                         |
| <u>stddev</u> ( <i>stddev</i> )                        | variance structure of the latent-variable errors                                                                                                                            |
| <u>structural</u>                                      | use the structural covariance parameterization; default is the differenced covariance parameterization                                                                      |
| <u>factor</u> (#)                                      | use the factor covariance structure with dimension #                                                                                                                        |
| <u>noconstant</u>                                      | suppress the alternative-specific constant terms                                                                                                                            |
| <u>basealternative</u> (#   <i>lbl</i>   <i>str</i> )  | alternative used for normalizing location                                                                                                                                   |
| <u>scalealternative</u> (#   <i>lbl</i>   <i>str</i> ) | alternative used for normalizing scale                                                                                                                                      |
| <u>altwise</u>                                         | use alternative-wise deletion instead of casewise deletion                                                                                                                  |
| <u>reverse</u>                                         | interpret the lowest rank in <i>depvar</i> as the best; the default is the highest rank is the best                                                                         |
| SE/Robust                                              |                                                                                                                                                                             |
| <u>vce</u> ( <i>vcetype</i> )                          | <i>vcetype</i> may be <code>oim</code> , <code>robust</code> , <code>cluster</code> <i>clustvar</i> , <code>opg</code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                                              |                                                                                                                                                                             |
| <u>level</u> (#)                                       | set confidence level; default is <code>level(95)</code>                                                                                                                     |
| <u>notransform</u>                                     | do not transform variance–covariance estimates to the standard deviation and correlation metric                                                                             |
| <u>nocnsreport</u>                                     | do not display constraints                                                                                                                                                  |



## Integration

|                                              |                                                                |
|----------------------------------------------|----------------------------------------------------------------|
| <u>intmethod</u> ( <i>seqtype</i> )          | type of quasi- or pseudouniform sequence                       |
| <u>intpoints</u> (#)                         | number of points in each sequence                              |
| <u>intburn</u> (#)                           | starting index in the Hammersley or Halton sequence            |
| <u>intseed</u> ( <i>code</i>   #)            | pseudouniform random-number seed                               |
| <u>antithetics</u>                           | use antithetic draws                                           |
| <u>nopivot</u>                               | do not use integration interval pivoting                       |
| <u>initbhhh</u> (#)                          | use the BHHH optimization algorithm for the first # iterations |
| <u>favor</u> ( <u>speed</u>   <u>space</u> ) | favor speed or space when generating integration points        |

## Maximization

|                         |                                  |
|-------------------------|----------------------------------|
| <u>maximize_options</u> | control the maximization process |
|-------------------------|----------------------------------|

|                                |                                                           |
|--------------------------------|-----------------------------------------------------------|
| <sup>†</sup> <u>coeflegend</u> | display coefficients' legend instead of coefficient table |
|--------------------------------|-----------------------------------------------------------|

| <i>correlation</i>            | description                                                                                                                         |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <u>unstructured</u>           | one correlation parameter for each pair of alternatives; correlations with the <code>basealternative()</code> are zero; the default |
| <u>exchangeable</u>           | one correlation parameter common to all pairs of alternatives; correlations with the <code>basealternative()</code> are zero        |
| <u>independent</u>            | constrain all correlation parameters to zero                                                                                        |
| <u>pattern</u> <i>matname</i> | user-specified matrix identifying the correlation pattern                                                                           |
| <u>fixed</u> <i>matname</i>   | user-specified matrix identifying the fixed and free correlation parameters                                                         |

| <i>stddev</i>                 | description                                                                                                                                             |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>heteroskedastic</u>        | estimate standard deviation for each alternative; standard deviations for <code>basealternative()</code> and <code>scalealternative()</code> set to one |
| <u>homoskedastic</u>          | all standard deviations are one                                                                                                                         |
| <u>pattern</u> <i>matname</i> | user-specified matrix identifying the standard deviation pattern                                                                                        |
| <u>fixed</u> <i>matname</i>   | user-specified matrix identifying the fixed and free standard deviations                                                                                |

| <i>seqtype</i>    | description                    |
|-------------------|--------------------------------|
| <u>hammersley</u> | Hammersley point set           |
| <u>halton</u>     | Halton point set               |
| <u>random</u>     | uniform pseudorandom point set |

\* `case(varname)` and `alternatives(varname)` are required.

<sup>†</sup> `coeflegend` does not appear in the dialog box.

`bootstrap`, `by`, `jackknife`, `statsby`, and `xi` are allowed; see [U] [11.1.10 Prefix commands](#).

Weights are not allowed with the `bootstrap` prefix.

`fweights`, `iweights`, and `pweights` are allowed; see [U] [11.1.6 weight](#).

See [U] [20 Estimation and postestimation commands](#) for more capabilities of estimation commands.

## Menu

Statistics > Ordinal outcomes > Rank-ordered probit regression

## Description

`asprobit` fits rank-ordered probit (ROP) models by using maximum simulated likelihood (MSL). The model allows you to relax the independence of irrelevant alternatives (IIA) property that is characteristic of the rank-ordered logistic model by estimating the variance–covariance parameters of the latent-variable errors. Each unique identifier in the `case()` variable has multiple alternatives identified in the `alternatives()` variable, and `depvar` contains the ranked alternatives made by each case. Only the order in the ranks, not the magnitude of their differences, is assumed to be relevant. By default, the largest rank indicates the more desirable alternative. Use the `reverse` option if the lowest rank should be interpreted as the more desirable alternative. Tied ranks are allowed, but they increase the computation time because all permutations of the tied ranks are used in computing the likelihood for each case. `asprobit` allows two types of independent variables: alternative-specific variables, in which the values of each variable vary with each alternative, and case-specific variables, which vary with each case.

The estimation technique of `asprobit` is nearly identical to that of `asmprobit`, and the two routines share many of the same options; see [\[R\] `asmprobit`](#).

## Options

### Model

`case(varname)` specifies the variable that identifies each case. This variable identifies the individuals or entities making a choice. `case()` is required.

`alternatives(varname)` specifies the variable that identifies the alternatives available for each case. The number of alternatives can vary with each case; the maximum number of alternatives is 20. `alternatives()` is required.

`casevars(varlist)` specifies the case-specific variables that are constant for each `case()`. If there are a maximum of  $J$  alternatives, there will be  $J - 1$  sets of coefficients associated with `casevars()`.

`constraints(constraints)`, `collinear`; see [\[R\] estimation options](#).

### Model 2

`correlation(correlation)` specifies the correlation structure of the latent-variable errors.

`correlation(unstructured)` is the most general and has  $J(J - 3)/2 + 1$  unique correlation parameters. This is the default unless `stddev()` or `structural` are specified.

`correlation(exchangeable)` provides for one correlation coefficient common to all latent variables, except the latent variable associated with the `basealternative()`.

`correlation(independent)` assumes that all correlations are zero.

`correlation(pattern matname)` and `correlation(fixed matname)` give you more flexibility in defining the correlation structure. See [Variance structures](#) in [\[R\] `asmprobit`](#) for more information.

`stddev(stddev)` specifies the variance structure of the latent-variable errors.

`stddev(heteroskedastic)` is the most general and has  $J - 2$  estimable parameters. The standard deviations of the latent-variable errors for the alternatives specified in `basealternative()` and `scalealternative()` are fixed to one.

`stddev(homoskedastic)` constrains all the standard deviations to equal one.

`stddev(pattern matname)` and `stddev(fixed matname)` give you added flexibility in defining the standard deviation parameters. See [Variance structures](#) in [R] **asmprobit** for more information.

`structural` requests the  $J \times J$  structural covariance parameterization instead of the default  $J-1 \times J-1$  differenced covariance parameterization (the covariance of the latent errors differenced with that of the base alternative). The differenced covariance parameterization will achieve the same maximum simulated likelihood regardless of the choice of `basealternative()` and `scalealternative()`. On the other hand, the structural covariance parameterization imposes more normalizations that may bound the model away from its maximum likelihood and thus prevent convergence with some datasets or choices of `basealternative()` and `scalealternative()`.

`factor(#)` requests that the factor covariance structure of dimension `#` be used. The `factor()` option can be used with the `structural` option but cannot be used with `stddev()` or `correlation()`. A  $\# \times J$  (or  $\# \times J-1$ ) matrix,  $\mathbf{C}$ , is used to factor the covariance matrix as  $I + \mathbf{C}'\mathbf{C}$ , where  $I$  is the identity matrix of dimension  $J$  (or  $J-1$ ). The column dimension of  $\mathbf{C}$  depends on whether the covariance is structural or differenced. The row dimension of  $\mathbf{C}$ , `#`, must be less than or equal to  $\text{floor}((J(J-1)/2 - 1)/(J-2))$ , because there are only  $J(J-1)/2 - 1$  identifiable variance-covariance parameters. This covariance parameterization may be useful for reducing the number of covariance parameters that need to be estimated.

If the covariance is structural, the column of  $\mathbf{C}$  corresponding to the base alternative contains zeros. The column corresponding to the scale alternative has a one in the first row and zeros elsewhere. If the covariance is differenced, the column corresponding to the scale alternative (differenced with the base) has a one in the first row and zeros elsewhere.

`noconstant` suppresses the  $J-1$  alternative-specific constant terms.

`basealternative(#|lbl|str)` specifies the alternative used to normalize the latent-variable location (also referred to as the level of utility). The base alternative may be specified as a number, label, or string. The standard deviation for the latent-variable error associated with the base alternative is fixed to one, and its correlations with all other latent-variable errors are set to zero. The default is the first alternative when sorted. If a `fixed` or `pattern` matrix is given in the `stddev()` and `correlation()` options, the `basealternative()` will be implied by the fixed standard deviations and correlations in the matrix specifications. `basealternative()` cannot be equal to `scalealternative()`.

`scalealternative(#|lbl|str)` specifies the alternative used to normalize the latent-variable scale (also referred to as the scale of utility). The scale alternative may be specified as a number, label, or string. The default is to use the second alternative when sorted. If a `fixed` or `pattern` matrix is given in the `stddev()` option, the `scalealternative()` will be implied in the fixed standard deviations in the matrix specification. `scalealternative()` cannot be equal to `basealternative()`.

If a `fixed` or `pattern` matrix is given for the `stddev()` option, the base alternative and scale alternative are implied by the standard deviations and correlations in the matrix specifications, and they need not be specified in the `basealternative()` and `scalealternative()` options.

`altwise` specifies that alternative-wise deletion be used when marking out observations due to missing values in your variables. The default is to use casewise deletion; that is, the entire group of observations making up a case is deleted if any missing values are encountered. This option does not apply to observations that are marked out by the `if` or `in` qualifier or the `by` prefix.

`reverse` directs `asroprobit` to interpret the rank in `depvar` that is smallest in value as the preferred alternative. By default, the rank that is largest in value is the favored alternative.

## SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

If specifying `vce(bootstrap)` or `vce(jackknife)`, you must also specify `basealternative()` and `scalealternative()`.

## Reporting

`level(#)`; see [R] [estimation options](#).

`notransform` prevents retransforming the Cholesky-factored variance–covariance estimates to the correlation and standard deviation metric.

This option has no effect if `structural` is not specified because the default differenced variance–covariance estimates have no interesting interpretation as correlations and standard deviations. `notransform` also has no effect if the `correlation()` and `stddev()` options are specified with anything other than their default values. Here it is generally not possible to factor the variance–covariance matrix, so optimization is already performed using the standard deviation and correlation representations.

`nocnsreport`; see [R] [estimation options](#).

## Integration

`intmethod(hammersley | halton | random)` specifies the method of generating the point sets used in the quasi–Monte Carlo integration of the multivariate normal density. `intmethod(hammersley)`, the default, uses the Hammersley sequence; `intmethod(halton)` uses the Halton sequence; and `intmethod(random)` uses a sequence of uniform random numbers.

`intpoints(#)` specifies the number of points to use in the quasi–Monte Carlo integration. If this option is not specified, the number of points is  $50 \times J$  if `intmethod(hammersley)` or `intmethod(halton)` is used and  $100 \times J$  if `intmethod(random)` is used. Larger values of `intpoints()` provide better approximations of the log likelihood, but at the cost of added computation time.

`intburn(#)` specifies where in the Halton or Hammersley sequence to start, which helps reduce the correlation between the sequences of each dimension. The default is 0. This option may not be specified with `intmethod(random)`.

`intseed(code | #)` specifies the seed to use for generating the uniform pseudorandom sequence. This option may be specified only with `intmethod(random)`. `code` refers to a string that records the state of the random-number generator `runiform()`; see [R] [set seed](#). An integer value `#` may be used also. The default is to use the current seed value from Stata’s uniform random-number generator, which can be obtained from `c(seed)`.

`antithetics` specifies that antithetic draws be used. The antithetic draw for the  $J - 1$  vector uniform-random variables,  $\mathbf{x}$ , is  $1 - \mathbf{x}$ .

`nopivot` turns off integration interval pivoting. By default, `asroprobit` will pivot the wider intervals of integration to the interior of the multivariate integration. This improves the accuracy of the quadrature estimate. However, discontinuities may result in the computation of numerical second-order derivatives using finite differencing (for the Newton–Raphson optimize technique, `tech(nr)`) when few simulation points are used, resulting in a nonpositive-definite Hessian. `asroprobit` uses the Broyden–Fletcher–Goldfarb–Shanno optimization algorithm, by default, which does not require computing the Hessian numerically using finite differencing.

`initbhhh(#)` specifies that the Berndt–Hall–Hall–Hausman (BHHH) algorithm be used for the initial # optimization steps. This option is the only way to use the BHHH algorithm along with other optimization techniques. The algorithm switching feature of `ml`'s `technique()` option cannot include `bhhh`.

`favor(speed|space)` instructs `asroprobit` to favor either speed or space when generating the integration points. `favor(speed)` is the default. When favoring speed, the integration points are generated once and stored in memory, thus increasing the speed of evaluating the likelihood. This speed increase can be seen when there are many cases or when the user specifies a large number of integration points, `intpoints(#)`. When favoring space, the integration points are generated repeatedly with each likelihood evaluation.

For unbalanced data, where the number of alternatives varies with each case, the estimates computed using `intmethod(random)` will vary slightly between `favor(speed)` and `favor(space)`. This is because the uniform sequences will not be identical, even when initiating the sequences using the same uniform seed, `intseed(code|#)`. For `favor(speed)`, `ncase` blocks of `intpoints(#)`  $\times$   $J - 2$  uniform points are generated, where  $J$  is the maximum number of alternatives. For `favor(space)`, the column dimension of the matrices of points varies with the number of alternatives that each case has.

#### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [R] [maximize](#).

The following options may be particularly useful in obtaining convergence with `asroprobit`: `difficult`, `technique(algorithm_spec)`, `nrtolerance(#)`, `nonrtolerance`, and `from(init_specs)`.

If `technique()` contains more than one algorithm specification, `bhhh` cannot be one of them. To use the BHHH algorithm with another algorithm, use the `initbhhh()` option and specify the other algorithm in `technique()`.

Setting the optimization type to `technique(bhhh)` resets the default `vcetype` to `vce(opg)`.

When specifying `from(matname [, copy])`, the values in `matname` associated with the latent-variable error variances must be for the log-transformed standard deviations and inverse-hyperbolic tangent-transformed correlations. This option makes using the coefficient vector from a previously fitted `asroprobit` model convenient as a starting point.

The following option is available with `asroprobit` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

The mathematical description and numerical computations of the rank-ordered probit model are similar to that of the multinomial probit model. The only difference is that the dependent variable of the rank-ordered probit model is ordinal, showing preferences among alternatives, as opposed to the binary dependent variable of the multinomial probit model, indicating a chosen alternative. We will describe how the likelihood of a ranking is computed using the latent-variable framework here, but for details of the latent-variable parameterization of these models and the method of maximum simulated likelihood, see [R] [asmprobit](#).

Consider the latent-variable parameterization of a  $J$  alternative rank-ordered probit model. Using the notation from `asmprobit`, we have variables  $\eta_{ij}$ ,  $j = 1, \dots, J$ , such that

$$\eta_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_i\boldsymbol{\alpha}_j + \xi_{ij}$$

Here the  $\mathbf{x}_{ij}$  are the alternative-specific independent variables, the  $\mathbf{z}_i$  are the case-specific variables, and the  $\xi_{ij}$  are multivariate normal with mean zero and covariance  $\boldsymbol{\Omega}$ . Without loss of generality, assume that individual  $i$  ranks the alternatives in order of the alternative indices  $j = 1, 2, \dots, J$ , so the alternative  $J$  is the preferred alternative and alternative 1 is the least preferred alternative. The probability of this ranking given  $\boldsymbol{\beta}$  and  $\boldsymbol{\alpha}_j$  is the probability that  $\eta_{i,J-1} - \eta_{i,J} \leq 0$  and  $\eta_{i,J-2} - \eta_{i,J-1} \leq 0, \dots$ , and  $\eta_{i,1} - \eta_{i,2} \leq 0$ .

► Example 1

Long and Freese (2006) provide an example of a rank-ordered logit model with alternative-specific variables. We use this dataset to demonstrate `asroprobit`. The data come from the Wisconsin Longitudinal Study. This is a study of 1957 Wisconsin high school graduates that were asked to rate their relative preference of four job characteristics: esteem, a job other people regard highly; variety, a job that is not repetitive and allows you to do a variety of things; autonomy, a job where your supervisor does not check on you frequently; and security, a job with a low risk of being laid off. The case-specific covariates are gender, `female`, an indicator variable for females, and `score`, a score on a general mental ability test measured in standard deviations. The alternative-specific variables are `high` and `low`, which indicate whether the respondent's current job is high or low in esteem, variety, autonomy, or security. This approach provides three states for a respondent's current job status for each alternative, (1, 0), (0, 1), and (0, 0), using the notation (`high`, `low`). The score (1, 1) is omitted because the respondent's current job cannot be considered both high and low in one of the job characteristics. The (0, 0) score would indicate that the respondent's current job does not rank high or low (is neutral) in a job characteristic. The alternatives are ranked such that 1 is the preferred alternative and 4 is the least preferred.

```
. use http://www.stata-press.com/data/r11/wlsrank
(1992 Wisconsin Longitudinal Study data on job values)
. list id jobchar rank female score high low in 1/12, sepby(id)
```

|     | id | jobchar  | rank | female | score    | high | low |
|-----|----|----------|------|--------|----------|------|-----|
| 1.  | 1  | security | 1    | 1      | .0492111 | 0    | 0   |
| 2.  | 1  | autonomy | 4    | 1      | .0492111 | 0    | 0   |
| 3.  | 1  | variety  | 1    | 1      | .0492111 | 0    | 0   |
| 4.  | 1  | esteem   | 3    | 1      | .0492111 | 0    | 0   |
| 5.  | 5  | security | 2    | 1      | 2.115012 | 1    | 0   |
| 6.  | 5  | variety  | 2    | 1      | 2.115012 | 1    | 0   |
| 7.  | 5  | esteem   | 2    | 1      | 2.115012 | 1    | 0   |
| 8.  | 5  | autonomy | 1    | 1      | 2.115012 | 0    | 0   |
| 9.  | 7  | autonomy | 1    | 0      | 1.701852 | 1    | 0   |
| 10. | 7  | variety  | 1    | 0      | 1.701852 | 0    | 1   |
| 11. | 7  | esteem   | 4    | 0      | 1.701852 | 0    | 0   |
| 12. | 7  | security | 1    | 0      | 1.701852 | 0    | 0   |

The three cases listed have tied ranks. `asroprobit` will allow ties, but at the cost of increased computation time. To evaluate the likelihood of the first observation, `asroprobit` must compute

$$\begin{aligned} &\Pr(\text{esteem} = 3, \text{variety} = 1, \text{autonomy} = 4, \text{security} = 2) + \\ &\Pr(\text{esteem} = 3, \text{variety} = 2, \text{autonomy} = 4, \text{security} = 1) \end{aligned}$$

and both of these probabilities are estimated using simulation. In fact, the full dataset contains 7,237 tied ranks and asroprobit takes a great deal of time to estimate the parameters. For exposition, we estimate the rank-ordered probit model by using the cases without ties. These cases are marked in the variable `noties`.

The model of job preference is

$$\eta_{ij} = \beta_1 \text{high}_{ij} + \beta_2 \text{low}_{ij} + \alpha_{1j} \text{female}_i + \alpha_{2j} \text{score}_i + \alpha_{0j} + \xi_{ij}$$

for  $j = 1, 2, 3, 4$ . The base alternative will be `esteem`, so  $\alpha_{01} = \alpha_{11} = \alpha_{21} = 0$ .

```
. asroprobit rank high low if noties, casevars(female score) case(id)
> alternatives(jobchar) reverse
note: variable high has 107 cases that are not alternative-specific: there is
 no within-case variability
note: variable low has 193 cases that are not alternative-specific: there is
 no within-case variability

Iteration 0: log simulated-likelihood = -1103.2768
Iteration 1: log simulated-likelihood = -1089.3361 (backed up)
 (output omitted)

Alternative-specific rank-ordered probit Number of obs = 1660
Case variable: id Number of cases = 415
Alternative variable: jobchar Alts per case: min = 4
 avg = 4.0
 max = 4

Integration sequence: Hammersley
Integration points: 200
Log simulated-likelihood = -1080.2206 Wald chi2(8) = 34.01
 Prob > chi2 = 0.0000
```

| rank     | Coef.              | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|----------|--------------------|-----------|-------|-------|----------------------|----------|
| jobchar  |                    |           |       |       |                      |          |
| high     | .3741029           | .0925685  | 4.04  | 0.000 | .192672              | .5555337 |
| low      | -.0697443          | .1093317  | -0.64 | 0.524 | -.2840305            | .1445419 |
| esteem   | (base alternative) |           |       |       |                      |          |
| variety  |                    |           |       |       |                      |          |
| female   | .1351487           | .1843088  | 0.73  | 0.463 | -.2260899            | .4963873 |
| score    | .1405482           | .0977567  | 1.44  | 0.151 | -.0510515            | .3321479 |
| _cons    | 1.735016           | .1451343  | 11.95 | 0.000 | 1.450558             | 2.019474 |
| autonomy |                    |           |       |       |                      |          |
| female   | .2561828           | .1679565  | 1.53  | 0.127 | -.0730059            | .5853715 |
| score    | .1898853           | .0875668  | 2.17  | 0.030 | .0182575             | .361513  |
| _cons    | .7009797           | .1227336  | 5.71  | 0.000 | .4604262             | .9415333 |
| security |                    |           |       |       |                      |          |
| female   | .232622            | .2057547  | 1.13  | 0.258 | -.1706497            | .6358938 |
| score    | -.1780076          | .1102115  | -1.62 | 0.106 | -.3940181            | .038003  |
| _cons    | 1.343766           | .1600059  | 8.40  | 0.000 | 1.030161             | 1.657372 |
| /lnl2_2  | .1805151           | .0757296  | 2.38  | 0.017 | .0320878             | .3289424 |
| /lnl3_3  | .4843091           | .0793343  | 6.10  | 0.000 | .3288168             | .6398014 |
| /12_1    | .6062037           | .1169368  | 5.18  | 0.000 | .3770117             | .8353957 |
| /13_1    | .4509217           | .1431183  | 3.15  | 0.002 | .1704151             | .7314283 |
| /13_2    | .2289447           | .1226081  | 1.87  | 0.062 | -.0113627            | .4692521 |

```
(jobchar=esteem is the alternative normalizing location)
(jobchar=variety is the alternative normalizing scale)
```

We specified the `reverse` option because a rank of 1 is the highest preference. The variance–covariance estimates are for the Cholesky-factored variance–covariance for the latent-variable errors differenced with that of alternative `esteem`. We can view the estimated correlations by entering

```
. estat correlation
```

|          | variety | autonomy | security |
|----------|---------|----------|----------|
| variety  | 1.0000  |          |          |
| autonomy | 0.4516  | 1.0000   |          |
| security | 0.2652  | 0.2399   | 1.0000   |

Note: correlations are for alternatives differenced with `esteem`

and typing

```
. estat covariance
```

|          | variety  | autonomy | security |
|----------|----------|----------|----------|
| variety  | 2        |          |          |
| autonomy | .8573015 | 1.80229  |          |
| security | .6376996 | .5475882 | 2.890048 |

Note: covariances are for alternatives differenced with `esteem`

gives the (co)variances. [R] [mprobit](#) explains that if the latent-variable errors are independent, then the correlations in the differenced parameterization should be  $\sim 0.5$  and the variances should be  $\sim 2.0$ , which seems to be the case here.

The coefficient estimates for the probit models can be difficult to interpret because of the normalization for location and scale. The regression estimates for the case-specific variables will be relative to the base alternative and the regression estimates for both the case-specific and alternative-specific variables are affected by the scale normalization. The more pronounced the heteroskedasticity and correlations, the more pronounced the resulting estimate differences when choosing alternatives to normalize for location and scale. However, when using the differenced covariance structure, you will obtain the same model likelihood regardless of which alternatives you choose as the base and scale alternatives. For model interpretation, you can examine the estimated probabilities and marginal effects by using postestimation routines `predict` and `estat mfx`. See [R] [asroprobit postestimation](#).





## Saved results

asroprobit saves the following in `e()`:

### Scalars

|                                |                                                      |
|--------------------------------|------------------------------------------------------|
| <code>e(N)</code>              | number of observations                               |
| <code>e(N_case)</code>         | number of cases                                      |
| <code>e(N_ties)</code>         | number of ties                                       |
| <code>e(k)</code>              | number of parameters                                 |
| <code>e(k_alt)</code>          | number of alternatives                               |
| <code>e(k_indvars)</code>      | number of alternative-specific variables             |
| <code>e(k_casevars)</code>     | number of case-specific variables                    |
| <code>e(k_sigma)</code>        | number of variance estimates                         |
| <code>e(k_rho)</code>          | number of correlation estimates                      |
| <code>e(k_eq)</code>           | number of equations in <code>e(b)</code>             |
| <code>e(k_eq_model)</code>     | number of equations in model Wald test               |
| <code>e(df_m)</code>           | model degrees of freedom                             |
| <code>e(k_autoCns)</code>      | number of base, empty, and omitted constraints       |
| <code>e(ll)</code>             | log simulated-likelihood                             |
| <code>e(N_clust)</code>        | number of clusters                                   |
| <code>e(const)</code>          | constant indicator                                   |
| <code>e(i_base)</code>         | base alternative index                               |
| <code>e(i_scale)</code>        | scale alternative index                              |
| <code>e(mc_points)</code>      | number of Monte Carlo replications                   |
| <code>e(mc_burn)</code>        | starting sequence index                              |
| <code>e(mc_antithetics)</code> | antithetics indicator                                |
| <code>e(reverse)</code>        | 1 if minimum rank is best, 0 if maximum rank is best |
| <code>e(chi2)</code>           | $\chi^2$                                             |
| <code>e(p)</code>              | significance                                         |
| <code>e(fullcov)</code>        | unstructured covariance indicator                    |
| <code>e(structcov)</code>      | 1 if structured covariance; 0 otherwise              |
| <code>e(cholesky)</code>       | Cholesky-factored covariance indicator               |
| <code>e(alt_min)</code>        | minimum number of alternatives                       |
| <code>e(alt_avg)</code>        | average number of alternatives                       |
| <code>e(alt_max)</code>        | maximum number of alternatives                       |
| <code>e(rank)</code>           | rank of <code>e(V)</code>                            |
| <code>e(ic)</code>             | number of iterations                                 |
| <code>e(rc)</code>             | return code                                          |
| <code>e(converged)</code>      | 1 if converged, 0 otherwise                          |

*(Continued on next page)*

## Macros

|                                   |                                                                          |
|-----------------------------------|--------------------------------------------------------------------------|
| <code>e(cmd)</code>               | <code>asroprobit</code>                                                  |
| <code>e(cmd2)</code>              | <code>asroprobit</code>                                                  |
| <code>e(cmdline)</code>           | command as typed                                                         |
| <code>e(depvar)</code>            | name of dependent variable                                               |
| <code>e(indvars)</code>           | alternative-specific independent variable                                |
| <code>e(casevars)</code>          | case-specific variables                                                  |
| <code>e(case)</code>              | variable defining cases                                                  |
| <code>e(altvar)</code>            | variable defining alternatives                                           |
| <code>e(alteqs)</code>            | alternative equation names                                               |
| <code>e(alt#)</code>              | alternative labels                                                       |
| <code>e(wtype)</code>             | weight type                                                              |
| <code>e(wexp)</code>              | weight expression                                                        |
| <code>e(title)</code>             | title in estimation output                                               |
| <code>e(clustvar)</code>          | name of cluster variable                                                 |
| <code>e(correlation)</code>       | correlation structure                                                    |
| <code>e(stddev)</code>            | variance structure                                                       |
| <code>e(cov_class)</code>         | class of the covariance structure                                        |
| <code>e(chi2type)</code>          | Wald, type of model $\chi^2$ test                                        |
| <code>e(vce)</code>               | <i>vcetype</i> specified in <code>vce()</code>                           |
| <code>e(vcetype)</code>           | title used to label Std. Err.                                            |
| <code>e(opt)</code>               | type of optimization                                                     |
| <code>e(which)</code>             | max or min; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>         | type of ml method                                                        |
| <code>e(mc_method)</code>         | Hammersley, Halton, or uniform random; technique to generate sequences   |
| <code>e(mc_seed)</code>           | random-number generator seed                                             |
| <code>e(user)</code>              | name of likelihood-evaluator program                                     |
| <code>e(technique)</code>         | maximization technique                                                   |
| <code>e(singularHmethod)</code>   | m-marquardt or hybrid; method used when Hessian is singular              |
| <code>e(crittype)</code>          | optimization criterion                                                   |
| <code>e(datasignature)</code>     | the checksum                                                             |
| <code>e(datasignaturevars)</code> | variables used in calculation of checksum                                |
| <code>e(properties)</code>        | b V                                                                      |
| <code>e(estat_cmd)</code>         | program used to implement <code>estat</code>                             |
| <code>e(mfx_dlg)</code>           | program used to implement <code>estat mfx</code> dialog                  |
| <code>e(predict)</code>           | program used to implement <code>predict</code>                           |
| <code>e(marginsnotok)</code>      | predictions disallowed by <code>margins</code>                           |

## Matrices

|                              |                                                                                            |
|------------------------------|--------------------------------------------------------------------------------------------|
| <code>e(b)</code>            | coefficient vector                                                                         |
| <code>e(Cns)</code>          | constraints matrix                                                                         |
| <code>e(stats)</code>        | alternative statistics                                                                     |
| <code>e(stdpattern)</code>   | variance pattern                                                                           |
| <code>e(stdfixed)</code>     | fixed and free standard deviations                                                         |
| <code>e(altvals)</code>      | alternative values                                                                         |
| <code>e(altfreq)</code>      | alternative frequencies                                                                    |
| <code>e(alt_casevars)</code> | indicators for estimated case-specific coefficients— <code>e(k_alt) × e(k_casevars)</code> |
| <code>e(corpattern)</code>   | correlation structure                                                                      |
| <code>e(corfixed)</code>     | fixed and free correlations                                                                |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)                                                        |
| <code>e(gradient)</code>     | gradient vector                                                                            |
| <code>e(V)</code>            | variance–covariance matrix of the estimators                                               |
| <code>e(V_modelbased)</code> | model-based variance                                                                       |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`asroprobit` is implemented as an ado-file.

From a computational perspective, `asroprobit` is similar to `asmprobit` and the two programs share many numerical tools. Therefore, we will use the notation from [Methods and formulas](#) in [R] `asmprobit` to discuss the rank-ordered probit probability model.

The latent variables for a  $J$ -alternative model are  $\eta_{ij} = \mathbf{x}_{ij}\beta + \mathbf{z}_i\alpha_j + \xi_{ij}$ , for  $j = 1, \dots, J$ ,  $i = 1, \dots, n$ , and  $\xi'_i = (\xi_{i,1}, \dots, \xi_{i,J}) \sim \text{MVN}(\mathbf{0}, \mathbf{\Omega})$ . Without loss of generality, assume for the  $i$ th observation that an individual ranks the alternatives in the order of their numeric indices,  $\mathbf{y}_i = (J, J-1, \dots, 1)$ , so the first alternative is the most preferred and the last alternative is the least preferred. We can then difference the latent variables such that

$$\begin{aligned}
 v_{ik} &= \eta_{i,k+1} - \eta_{i,k} \\
 &= (\mathbf{x}_{i,k+1} - \mathbf{x}_{i,k})\beta + \mathbf{z}_i(\alpha_{k+1} - \alpha_k) + \xi_{i,k+1} - \xi_{ik} \\
 &= \delta_{ik}\beta + \mathbf{z}_i\gamma_k + \epsilon_{ik}
 \end{aligned}$$

for  $k = 1, \dots, J-1$  and where  $\epsilon_i = (\epsilon_{i1}, \dots, \epsilon_{i,J-1}) \sim \text{MVN}(\mathbf{0}, \mathbf{\Sigma}_{(i)})$ .  $\mathbf{\Sigma}$  is indexed by  $i$  because it is specific to the ranking of individual  $i$ . We denote the deterministic part of the model as  $\lambda_{ik} = \delta_{ik}\beta + \mathbf{z}_i\gamma_k$ , and the probability of this event is

$$\begin{aligned}
 \Pr(\mathbf{y}_i) &= \Pr(v_{i1} \leq 0, \dots, v_{i,J-1} \leq 0) \\
 &= \Pr(\epsilon_{i1} \leq -\lambda_{i1}, \dots, \epsilon_{i,J-1} \leq -\lambda_{i,J-1}) \\
 &= (2\pi)^{-(J-1)/2} |\mathbf{\Sigma}_{(i)}|^{-1/2} \int_{-\infty}^{-\lambda_{i1}} \dots \int_{-\infty}^{-\lambda_{i,J-1}} \exp\left(-\frac{1}{2}\mathbf{z}'\mathbf{\Sigma}_{(i)}^{-1}\mathbf{z}\right) d\mathbf{z}
 \end{aligned}$$

The integral has the same form as (3) of [Methods and formulas](#) in [R] `asmprobit`. See [R] `asmprobit` for details on evaluating this integral numerically by using simulation.

`asroprobit` handles tied ranks by enumeration. For  $k$  tied ranks, it will generate  $k!$  rankings, where  $!$  is the factorial operator  $k! = k(k-1)(k-2) \cdots (2)(1)$ . For two sets of tied ranks of size  $k_1$  and  $k_2$ , `asroprobit` will generate  $k_1!k_2!$  rankings. The total probability is the sum of the probability of each ranking. For example, if there are two tied ranks such that  $\mathbf{y}_i = (J, J, J-2, \dots, 1)$ , then `asroprobit` will evaluate  $\Pr(\mathbf{y}_i) = \Pr(\mathbf{y}_i^{(1)}) + \Pr(\mathbf{y}_i^{(2)})$ , where  $\mathbf{y}_i^{(1)} = (J, J-1, J-2, \dots, 1)$  and  $\mathbf{y}_i^{(2)} = (J-1, J, J-2, \dots, 1)$ .

This command supports the clustered version of the Huber/White/sandwich estimator of the variance using `vce(robust)` and `vce(cluster clustvar)`. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#). Specifying `vce(robust)` is equivalent to specifying `vce(cluster casevar)`, where `casevar` is the variable that identifies the cases.

## Reference

Long, J. S., and J. Freese. 2006. [Regression Models for Categorical Dependent Variables Using Stata](#). 2nd ed. College Station, TX: Stata Press.

## Also see

- [R] [asroprobit postestimation](#) — Postestimation tools for `asroprobit`
- [R] [asmprobit](#) — Alternative-specific multinomial probit regression
- [R] [mprobit](#) — Multinomial probit regression
- [R] [mlogit](#) — Multinomial (polytomous) logistic regression
- [R] [oprobit](#) — Ordered probit regression
- [U] [20 Estimation and postestimation commands](#)

## Description

The following postestimation commands are of special interest after `asroprobit`:

| command                         | description                                    |
|---------------------------------|------------------------------------------------|
| <code>estat alternatives</code> | alternative summary statistics                 |
| <code>estat covariance</code>   | variance–covariance matrix of the alternatives |
| <code>estat correlation</code>  | correlation matrix of the alternatives         |
| <code>estat facweights</code>   | covariance factor weights matrix               |
| <code>estat mfx</code>          | marginal effects                               |

For information about these commands, see below.

The following standard postestimation commands are also available:

| command                | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>     | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estimates</code> | cataloging estimation results                                                                       |
| <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code>    | likelihood-ratio test                                                                               |
| <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>   | predicted probabilities, estimated linear predictor and its standard error                          |
| <code>predictnl</code> | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details.

## Special-interest postestimation commands

- `estat alternatives` displays summary statistics about the alternatives in the estimation sample. The command also provides a mapping between the index numbers that label the covariance parameters of the model and their associated values and labels for the alternative variable.
- `estat covariance` computes the estimated variance–covariance matrix for the alternatives. The estimates are displayed, and the variance–covariance matrix is stored in `r(cov)`.
- `estat correlation` computes the estimated correlation matrix for the alternatives. The estimates are displayed, and the correlation matrix is stored in `r(cor)`.
- `estat facweights` displays the covariance factor weights matrix and stores it in `r(C)`.

`estat mfx` computes marginal effects of a simulated probability of a set of ranked alternatives. The probability is stored in `r(pr)`, the matrix of rankings is stored in `r(ranks)`, and the matrix of marginal-effect statistics is stored in `r(mfx)`.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic altwise]

predict [type] { stub* | newvarlist } [if] [in], scores
```

| statistic | description                                       |
|-----------|---------------------------------------------------|
| Main      |                                                   |
| pr        | probability of each ranking, by case; the default |
| pr1       | probability that each alternative is preferred    |
| xb        | linear prediction                                 |
| stdp      | standard error of the linear prediction           |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

Menu

Statistics > Postestimation > Predictions, residuals, etc.

Options for predict

Main

`pr`, the default, calculates the probability of each ranking. For each case, one probability is computed for the ranks in `e(depvar)`.

`pr1` calculates the probability that each alternative is preferred.

`xb` calculates the linear prediction  $\mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_i\boldsymbol{\alpha}_j$  for alternative  $j$  and case  $i$ .

`stdp` calculates the standard error of the linear predictor.

`altwise` specifies that alternative-wise deletion be used when marking out observations due to missing values in your variables. The default is to use casewise deletion. The `xb` and `stdp` options always use alternative-wise deletion.

`scores` calculates the scores for each coefficient in `e(b)`. This option requires a new variable list of length equal to the number of columns in `e(b)`. Otherwise, use the `stub*` option to have `predict` generate enumerated variables with prefix `stub`.

Syntax for estat alternatives

```
estat alternatives
```

## Menu

Statistics > Postestimation > Reports and statistics

## Syntax for estat covariance

```
estat covariance [, format(string) border(string) left(#)]
```

## Menu

Statistics > Postestimation > Reports and statistics

## Options for estat covariance

`format(string)` sets the matrix display format. The default is `format(%9.0g)`.

`border(string)` sets the matrix display border style. The default is `border(all)`. See [\[P\] matlist](#).

`left(#)` sets the matrix display left indent. The default is `left(2)`. See [\[P\] matlist](#).

## Syntax for estat correlation

```
estat correlation [, format(string) border(string) left(#)]
```

## Menu

Statistics > Postestimation > Reports and statistics

## Options for estat correlation

`format(string)` sets the matrix display format. The default is `format(%9.4f)`.

`border(string)` sets the matrix display border style. The default is `border(all)`. See [\[P\] matlist](#).

`left(#)` sets the matrix display left indent. The default is `left(2)`. See [\[P\] matlist](#).

## Syntax for estat facweights

```
estat facweights [, format(string) border(string) left(#)]
```

## Menu

Statistics > Postestimation > Reports and statistics

## Options for estat facweights

`format(string)` sets the matrix display format. The default is `format(%9.0f)`.

`border(string)` sets the matrix display border style. The default is `border(all)`. See [\[P\] matlist](#).

`left(#)` sets the matrix display left indent. The default is `left(2)`. See [\[P\] matlist](#).

## Syntax for estat mfx

```
estat mfx [if] [in] [, options]
```

| <i>options</i>                          | description                                                                           |
|-----------------------------------------|---------------------------------------------------------------------------------------|
| <b>Main</b>                             |                                                                                       |
| <code>varlist(<i>varlist</i>)</code>    | display marginal effects for <i>varlist</i>                                           |
| <code>at(median [<i>atlist</i>])</code> | calculate marginal effects at these values                                            |
| <code>rank(<i>atlist</i>)</code>        | calculate marginal effects for the simulated probability of these ranked alternatives |
| <b>Options</b>                          |                                                                                       |
| <code>level(#)</code>                   | set confidence interval level; default is level(95)                                   |
| <code>nodiscrete</code>                 | treat indicator variables as continuous                                               |
| <code>noesample</code>                  | do not restrict calculation of the medians to the estimation sample                   |
| <code>nowght</code>                     | ignore weights when calculating medians                                               |

## Menu

Statistics > Postestimation > Reports and statistics

## Options for estat mfx

### Main

`varlist(varlist)` specifies the variables for which to display marginal effects. The default is all variables.

`at(median [alternative:variable = #] [variable = #] [...])` specifies the values at which the marginal effects are to be calculated. The default is to compute the marginal effects at the medians of the independent variables by using the estimation sample, `at(median)`. You can also specify specific values for variables. Values for alternative-specific variables can be specified by alternative, or you can specify one value for all alternatives. You can specify only one value for case-specific variables. For example, in the `wlsrank` dataset, `female` and `score` are case-specific variables, whereas `high` and `low` are alternative-specific variables. The following would be a legal syntax for `estat mfx`:

```
. estat mfx, at(median high=0 esteem:high=1 low=0 security:low=1 female=1)
```

When `nodiscrete` is not specified, `at(median [atlist])` has no effect on computing marginal effects for indicator variables, which are calculated as the discrete change in the simulated probability as the indicator variable changes from 0 to 1.

The median computations respect any `if` or `in` qualifiers, so you can restrict the data over which the medians are computed. You can even restrict the values to a specific case, e.g.,

```
. estat mfx if case==13
```

`rank(alternative = # alternative = # [...])` specifies the ranks for the alternatives. The default is to rank the calculated latent variables. Alternatives excluded from `rank()` are omitted from the analysis. You must therefore specify at least two alternatives in `rank()`. You may have tied ranks in the rank specification. Only the order in the ranks is relevant.



## Options

`level(#)` sets the confidence level; default is `level(95)`.

`nodiscrete` specifies that indicator variables be treated as continuous variables. An indicator variable is one that takes on the value 0 or 1 in the estimation sample. By default, the discrete change in the simulated probability is computed as the indicator variable changes from 0 to 1.

`noesample` specifies that the whole dataset be considered instead of only those marked in the `e(sample)` defined by the `asroprobit` command.

`nowght` specifies that weights be ignored when calculating the medians.

## Remarks

Remarks are presented under the following headings:

*Predicted probabilities*

*Obtaining estimation statistics*

## Predicted probabilities

After fitting an alternative-specific rank-ordered probit model, you can use `predict` to obtain the probabilities of alternative rankings or the probabilities of each alternative being preferred. When evaluating the multivariate normal probabilities via (quasi) Monte Carlo, `predict` uses the same method to generate the (quasi) random sequence of numbers as the previous call to `asroprobit`. For example, if you specified `intmethod(halton)` when fitting the model, `predict` also uses Halton sequences.

### ► Example 1

In [example 1](#) of [\[R\] asroprobit](#), we fit a model of job characteristic preferences. This is a study of 1957 Wisconsin high school graduates that were asked to rate their relative preference of four job characteristics: esteem, a job other people regard highly; variety, a job that is not repetitive and allows you to do a variety of things; autonomy, a job where your supervisor does not check on you frequently; and security, a job with a low risk of being laid off. The case-specific covariates are gender, `female`, an indicator variable for females, and `score`, a score on a general mental ability test measured in standard deviations. The alternative-specific variables are `high` and `low`, which indicate whether the respondent's current job is high or low in esteem, variety, autonomy, or security. This approach provides three states for a respondent's current job status for each alternative, (1,0), (0,1), and (0,0), using the notation (high, low). The score (1,1) is omitted because the respondent's current job cannot be considered both high and low in one of the job characteristics. The (0,0) score would indicate that the respondent's current job does not rank high or low (is neutral) in a job characteristic. The alternatives are ranked such that 1 is the preferred alternative and 4 is the least preferred.

We can obtain the probabilities of the observed alternative rankings, the `pr` option, and the probability of each alternative being preferred, the `pr1` option, by using `predict`:

```
. use http://www.stata-press.com/data/r11/wlsrank
(1992 Wisconsin Longitudinal Study data on job values)

. asroprobit rank high low if noties, casevars(female score) case(id)
> alternatives(jobchar) reverse
(output omitted)
```

```
. keep if e(sample)
(11244 observations deleted)
. predict prob, pr
. predict prob1, pr1
. list id jobchar prob prob1 rank female score high low in 1/12
```

|     | id | jobchar  | prob     | prob1    | rank | female | score    | high | low |
|-----|----|----------|----------|----------|------|--------|----------|------|-----|
| 1.  | 13 | security | .0421807 | .2784269 | 3    | 0      | .3246512 | 0    | 1   |
| 2.  | 13 | autonomy | .0421807 | .1029036 | 1    | 0      | .3246512 | 0    | 0   |
| 3.  | 13 | variety  | .0421807 | .6026725 | 2    | 0      | .3246512 | 1    | 0   |
| 4.  | 13 | esteem   | .0421807 | .0160111 | 4    | 0      | .3246512 | 0    | 1   |
| 5.  | 19 | autonomy | .0942025 | .1232488 | 4    | 1      | .0492111 | 0    | 0   |
| 6.  | 19 | esteem   | .0942025 | .0140261 | 3    | 1      | .0492111 | 0    | 0   |
| 7.  | 19 | security | .0942025 | .4601368 | 1    | 1      | .0492111 | 1    | 0   |
| 8.  | 19 | variety  | .0942025 | .4025715 | 2    | 1      | .0492111 | 0    | 0   |
| 9.  | 22 | esteem   | .1414177 | .0255264 | 4    | 1      | 1.426412 | 1    | 0   |
| 10. | 22 | variety  | .1414177 | .4549441 | 1    | 1      | 1.426412 | 0    | 0   |
| 11. | 22 | security | .1414177 | .2629494 | 3    | 1      | 1.426412 | 0    | 0   |
| 12. | 22 | autonomy | .1414177 | .2566032 | 2    | 1      | 1.426412 | 1    | 0   |

The prob variable is constant for each case because it contains the probability of the ranking in the rank variable. On the other hand, the prob1 variable contains the estimated probability of each alternative being preferred. For each case, the sum of the values in prob1 will be approximately 1.0. They do not add up to exactly 1.0 because of approximations due to the GHK algorithm.



Obtaining estimation statistics

For examples of the specialized estat subcommands covariance and correlation, see [R] [asm-probit postestimation](#). The entry also has a good example of computing marginal effects after asm-probit that is applicable to asroprobit. Below we will elaborate further on marginal effects after asroprobit where we manipulate the rank() option.

Example 2

We will continue with the preferred job characteristics example where we first compute the marginal effects for case id = 13.

```
. estat mfx if id==13, rank(security=3 autonomy=1 variety=2 esteem=4)
Pr(estee=4 variety=2 autonomy=1 security=3) = .04218068
```

| variable | dp/dx    | Std. Err. | z     | P> z  | [        | 95% C.I. | ] | X      |
|----------|----------|-----------|-------|-------|----------|----------|---|--------|
| high*    |          |           |       |       |          |          |   |        |
| esteem   | -.008713 | .001964   | -4.44 | 0.000 | -.012562 | -.004864 |   | 0      |
| variety  | -.009102 | .003127   | -2.91 | 0.004 | -.015231 | -.002973 |   | 1      |
| autonomy | .025535  | .007029   | 3.63  | 0.000 | .011758  | .039313  |   | 0      |
| security | -.003745 | .001394   | -2.69 | 0.007 | -.006477 | -.001013 |   | 0      |
| low*     |          |           |       |       |          |          |   |        |
| esteem   | .001614  | .002646   | 0.61  | 0.542 | -.003572 | .0068    |   | 1      |
| variety  | .001809  | .003012   | 0.60  | 0.548 | -.004094 | .007712  |   | 0      |
| autonomy | -.003849 | .006104   | -0.63 | 0.528 | -.015813 | .008115  |   | 0      |
| security | .000582  | .000985   | 0.59  | 0.554 | -.001348 | .002513  |   | 1      |
| casevars |          |           |       |       |          |          |   |        |
| female*  | .009767  | .009064   | 1.08  | 0.281 | -.007998 | .027533  |   | 0      |
| score    | .008587  | .004488   | 1.91  | 0.056 | -.00021  | .017384  |   | .32465 |

(\*) dp/dx is for discrete change of indicator variable from 0 to 1

Next we compute the marginal effects for the probability that autonomy is preferred given the profile of case id = 13.

```
. estat mfx if id==13, rank(security=2 autonomy=1 variety=2 esteem=2)
Pr(estee=3 variety=4 autonomy=1 security=2) +
Pr(estee=4 variety=3 autonomy=1 security=2) +
Pr(estee=2 variety=4 autonomy=1 security=3) +
Pr(estee=4 variety=2 autonomy=1 security=3) +
Pr(estee=2 variety=3 autonomy=1 security=4) +
Pr(estee=3 variety=2 autonomy=1 security=4) = .10276103
```

| variable | dp/dx    | Std. Err. | z     | P> z  | [        | 95% C.I. | ] | X      |
|----------|----------|-----------|-------|-------|----------|----------|---|--------|
| high*    |          |           |       |       |          |          |   |        |
| esteem   | -.003524 | .001258   | -2.80 | 0.005 | -.005989 | -.001059 |   | 0      |
| variety  | -.036203 | .00894    | -4.05 | 0.000 | -.053724 | -.018681 |   | 1      |
| autonomy | .057279  | .013801   | 4.15  | 0.000 | .030231  | .084328  |   | 0      |
| security | -.0128   | .002665   | -4.80 | 0.000 | -.018024 | -.007576 |   | 0      |
| low*     |          |           |       |       |          |          |   |        |
| esteem   | .000518  | .000833   | 0.62  | 0.534 | -.001116 | .002151  |   | 1      |
| variety  | .006409  | .010588   | 0.61  | 0.545 | -.014343 | .027161  |   | 0      |
| autonomy | -.008818 | .013766   | -0.64 | 0.522 | -.035799 | .018163  |   | 0      |
| security | .002314  | .003697   | 0.63  | 0.531 | -.004932 | .009561  |   | 1      |
| casevars |          |           |       |       |          |          |   |        |
| female*  | .013839  | .021607   | 0.64  | 0.522 | -.028509 | .056188  |   | 0      |
| score    | .017917  | .011062   | 1.62  | 0.105 | -.003764 | .039598  |   | .32465 |

(\*) dp/dx is for discrete change of indicator variable from 0 to 1

The probability computed by `estat mfx` matches the probability computed by `predict, pr1` only within three digits. This outcome is because of how the computation is carried out and the numeric inaccuracy of the GHK simulator using a Hammersley point set of length 200. The computation carried out by `estat mfx` literally computes all six probabilities listed in the header of the MFX table and sums them. The computation by `predict, pr1` is the same as `predict after asrobit` (multinomial probit): it computes the probability that autonomy is chosen, thus requiring only one

call to the GHK simulator. Hence, there is a difference in the reported values even though the two probability statements are equivalent.



## Saved results

`estat mfx` saves the following in `r()`:

### Scalars

`r(pr)` scalar containing the computed probability of the ranked alternatives.

### Matrices

`r(ranks)` column vector containing the alternative ranks. The rownames identify the alternatives.

`r(mfx)` matrix containing the computed marginal effects and associated statistics. Column 1 of the matrix contains the marginal effects; column 2, their standard errors; column 3, their  $z$  statistics; and columns 4 and 5, the confidence intervals. Column 6 contains the values of the independent variables used to compute the probabilities `r(pr)`.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [asroprobit](#) — Alternative-specific rank-ordered probit regression

[R] [asmprobit](#) — Alternative-specific multinomial probit regression

[U] [20 Estimation and postestimation commands](#)

## Description

This entry discusses a statistical issue that arises when using the Bayesian information criterion (BIC) to compare models.

Stata calculates BIC, assuming  $N = e(N)$ —we will explain—but sometimes it would be better if a different  $N$  were used. Commands that calculate BIC have an `n()` option, allowing you to specify the  $N$  to be used.

In summary,

1. If you are comparing results estimated by the same estimation command, using the default BIC calculation is probably fine. There is an issue, but most researchers would ignore it.
2. If you are comparing results estimated by different estimation commands, you need to be on your guard.
  - a. If the different estimation commands share the same definitions of observations, independence, and the like, you are back in case 1.
  - b. If they differ in these regards, you need to think about the value of  $N$  that should be used. For example, `logit` and `xtlogit` differ in that the former assumes independent observations and the latter, independent panels.
  - c. If estimation commands differ in the events being used over which the likelihood function is calculated, the information criteria may not be comparable at all. We say information *criteria* because this would apply equally to the Akaike information criterion (AIC), as well as to BIC. For instance, `streg` and `stcox` produce such incomparable results. The events used by `streg` are the actual survival times, whereas the events used by `stcox` are failures within risk pools, conditional on the times at which failures occurred.

## Remarks

Remarks are presented under the following headings:

*Background*

*The problem of determining  $N$*

*The problem of conformable likelihoods*

*The first problem does not arise with AIC; the second problem does*  
*Calculating BIC correctly*

## Background

The AIC and the BIC are two popular measures for comparing maximum likelihood models. AIC and BIC are defined as

$$\text{AIC} = -2 \times \ln(\text{likelihood}) + 2 \times k$$

$$\text{BIC} = -2 \times \ln(\text{likelihood}) + \ln(N) \times k$$

where

$k$  = model degrees of freedom

$N$  = number of observations

We are going to discuss AIC along with BIC because AIC has some of the problems that BIC has, but not all.

AIC and BIC can be viewed as measures that combine fit and complexity. Fit is measured negatively by  $-2 \times \ln(\text{likelihood})$ ; the larger the value, the worse the fit. Complexity is measured positively, either by  $2 \times k$  (AIC) or  $\ln(N) \times k$  (BIC).

Given two models fit on the same data, the model with the smaller value of the information criterion is considered to be better.

There is a substantial literature on these measures: see [Akaike \(1974\)](#); [Raftery \(1995\)](#); [Sakamoto, Ishiguro, and Kitagawa \(1986\)](#); and [Schwarz \(1978\)](#).

When Stata calculates the above measures, it uses the rank of  $\mathbf{e(V)}$  for  $k$  and it uses  $\mathbf{e(N)}$  for  $N$ .  $\mathbf{e(V)}$  and  $\mathbf{e(N)}$  are Stata notation for results stored by the estimation command.  $\mathbf{e(V)}$  is the variance–covariance matrix of the estimated parameters, and  $\mathbf{e(N)}$  is the number of observations in the dataset used in calculating the result.

## The problem of determining $N$

The difference between AIC and BIC is that AIC uses the constant 2 to weight  $k$ , whereas BIC uses  $\ln(N)$ .

Determining what value of  $N$  should be used is problematic. Despite appearances, the definition “ $N$  is the number of observations” is not easy to make operational.  $N$  does not appear in the likelihood function itself,  $N$  is not the output of a standard statistical formula, and what is an observation is often subjective.

### ► Example 1

Often what is meant by  $N$  is obvious. Consider a simple logit model. What is meant by  $N$  is the number of observations that are statistically independent and that corresponds to  $M$ , the number of observations in the dataset used in the calculation. We will write  $N = M$ .

But now assume that the same dataset has a grouping variable and the data are thought to be clustered within group. To keep the problem simple, let’s pretend that there are  $G$  groups and  $m$  observations within group, so that  $M = G \times m$ . Because you are worried about intragroup correlation, you fit your model with `xtlogit`, grouping on the grouping variable. Now you wish to calculate BIC. What is the  $N$  that should be used?  $N = M$  or  $N = G$ ?

That is a deep question. If the observations really are independent, then you should use  $N = M$ . If the observations within group are not just correlated but are duplicates of one another, and they had to be so, then you should use  $N = G$ . Between those two extremes, you should probably use a number between  $N$  and  $G$ , but determining what that number should be from measured correlations is difficult. Using  $N = M$  is conservative in that, if anything, it overweights complexity. Conservativeness, however, is subjective, too: using  $N = G$  could be considered more conservative in that fewer constraints are being placed on the data.

When the estimated correlation is high, our reaction would be that using  $N = G$  is probably more reasonable. Our first reaction, however, would be that using BIC to compare models is probably a misuse of the measure.

Stata uses  $N = M$ . An informal survey of web-based literature suggests that  $N = M$  is the popular choice.

There is another reason, not so good, to choose  $N = M$ . It makes across-model comparisons more likely to be valid when performed without thinking about the issue. Say that you wish to compare the `logit` and `xtlogit` results. Thus, you need to calculate

$$\text{BIC}_p = -2 \times \ln(\text{likelihood}_p) + \ln(N_p) \times k$$

$$\text{BIC}_x = -2 \times \ln(\text{likelihood}_x) + \ln(N_x) \times k$$

Whatever  $N$  you use, you must use the same  $N$  in both formulas. Stata's choice of  $N = M$  at least meets that test.

◀

## ► Example 2

In the above example, using  $N = M$  is reasonable. Now let's look at when using  $N = M$  is wrong, even if popular.

Consider a model fit by `stcox`. Using  $N = M$  is certainly wrong if for no other reason than  $M$  is not even a well-defined number. The same data can be represented by different datasets with different numbers of observations. For example, in one dataset, there might be 1 observation per subject. In another, the same subjects could have two records each, the first recording the first half of the time at risk and the second recording the remaining part. All statistics calculated by Stata on either dataset would be the same, but  $M$  would be different.

Deciding on the right definition, however, is difficult. Viewed one way,  $N$  in the Cox regression case should be the number of risk pools,  $R$ , because the Cox regression calculation is made on the basis of the independent risk pools. Viewed another way,  $N$  should be the number of subjects,  $N_{\text{subj}}$ , because, even though the likelihood function is based on risk pools, the parameters estimated are at the subject level.

You can decide which argument you prefer.

For parametric survival models, in single-record data,  $N = M$  is unambiguously correct. For multirecord data, there is an argument for  $N = M$  and for  $N = N_{\text{subj}}$ .

◀

## The problem of conformable likelihoods

The problem of conformable likelihoods does not concern  $N$ . Researchers sometimes use information criteria such as BIC and AIC to make comparisons across models. For that to be valid, the likelihoods must be conformable; i.e., the likelihoods must all measure the same thing.

It is common to think of the likelihood function as the  $\text{Pr}(\text{data} | \text{parameters})$ , but in fact, the likelihood is

$$\text{Pr}(\text{particular events in the data} | \text{parameters})$$

You must ensure that the events are the same.

For instance, they are not the same in the semiparametric Cox regression and the various parametric survival models. In Cox regression, the events are, at each failure time, that the subjects observed to fail in fact failed, given that failures occurred at those times. In the parametric models, the events are that each subject failed exactly when the subject was observed to fail.

The formula for AIC and BIC is

$$\text{measure} = -2 \times \ln(\text{likelihood}) + \text{complexity}$$

When you are comparing models, if the likelihoods are measuring different events, even if the models obtain estimates of the same parameters, differences in the information measures are irrelevant.

The first problem does not arise with AIC; the second problem does

Regardless of model, the problem of defining  $N$  never arises with AIC because  $N$  is not used in the AIC calculation. AIC uses a constant 2 to weight complexity as measured by  $k$ , rather than  $\ln(N)$ .

For both AIC and BIC, however, the likelihood functions must be conformable; i.e., they must be measuring the same event.

Calculating BIC correctly

When using BIC to compare results, and especially when using BIC to compare results from different models, you should think carefully about how  $N$  should be defined. Then specify that number by using the `n()` option:

```
. estimates stats full sub, n(74)
```

| Model | Obs | ll(null)  | ll(model) | df | AIC      | BIC      |
|-------|-----|-----------|-----------|----|----------|----------|
| full  | 102 | -45.03321 | -20.59083 | 4  | 49.18167 | 58.39793 |
| sub   | 102 | -45.03321 | -27.17516 | 3  | 60.35031 | 67.26251 |

Note: N = 74 used in calculating BIC

Both `estimates stats` and `estat ic` allow the `n()` option; see [\[R\] estimates stats](#) and [\[R\] estat](#).

Methods and formulas

AIC and BIC are defined as

$$\text{AIC} = -2 \times \ln(\text{likelihood}) + 2 \times k$$

$$\text{BIC} = -2 \times \ln(\text{likelihood}) + \ln(N) \times k$$

where  $k$  is the model degrees of freedom calculated as the rank of variance–covariance matrix of the parameters  $\mathbf{e}(V)$  and  $N$  is the number of observations used in estimation or, more precisely, the number of independent terms in the likelihood. Operationally,  $N$  is defined as  $\mathbf{e}(N)$  unless the `n()` option is specified.

References

Akaike, H. 1974. A new look at the statistical model identification. *IEEE Transactions on Automatic Control* 19: 716–723.

Raftery, A. 1995. Bayesian model selection in social research. In Vol. 25 of *Sociological Methodology*, ed. P. V. Marsden, 111–163. Oxford: Blackwell.



Sakamoto, Y., M. Ishiguro, and G. Kitagawa. 1986. *Akaike Information Criterion Statistics*. Dordrecht, The Netherlands: Reidel.

Schwarz, G. 1978. Estimating the dimension of a model. *Annals of Statistics* 6: 461–464.

## Also see

[R] [estat](#) — Postestimation statistics

[R] [estimates stats](#) — Model statistics

## Syntax

```
binreg depvar [indepvars] [if] [in] [weight] [, options]
```

| <i>options</i>                              | description                                                                                                                                                                              |
|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                       |                                                                                                                                                                                          |
| <u>no</u> constant                          | suppress constant term                                                                                                                                                                   |
| or                                          | use logit link and report odds ratios                                                                                                                                                    |
| rr                                          | use log link and report risk ratios                                                                                                                                                      |
| hr                                          | use log-complement link and report health ratios                                                                                                                                         |
| rd                                          | use identity link and report risk differences                                                                                                                                            |
| n( <i>#</i>   <i>varname</i> )              | use <i>#</i> or <i>varname</i> for number of trials                                                                                                                                      |
| <u>exposure</u> ( <i>varname</i> )          | include ln( <i>varname</i> ) in model with coefficient constrained to 1                                                                                                                  |
| <u>offset</u> ( <i>varname</i> )            | include <i>varname</i> in model with coefficient constrained to 1                                                                                                                        |
| <u>constraints</u> ( <i>constraints</i> )   | apply specified linear constraints                                                                                                                                                       |
| <u>collinear</u>                            | keep collinear variables                                                                                                                                                                 |
| mu( <i>varname</i> )                        | use <i>varname</i> as the initial estimate for the mean of <i>depvar</i>                                                                                                                 |
| <u>init</u> ( <i>varname</i> )              | synonym for mu( <i>varname</i> )                                                                                                                                                         |
| SE/Robust                                   |                                                                                                                                                                                          |
| vce( <i>vcetype</i> )                       | <i>vcetype</i> may be <u>e</u> im, <u>r</u> obust, <u>c</u> luster <i>clustvar</i> , oim, opg, <u>b</u> ootstrap, <u>j</u> ackknife, hac <i>kernel</i> , jackknife1, or <u>u</u> nbiased |
| t( <i>varname</i> )                         | variable name corresponding to time                                                                                                                                                      |
| <u>v</u> factor( <i>#</i> )                 | multiply variance matrix by scalar <i>#</i>                                                                                                                                              |
| disp( <i>#</i> )                            | quasi-likelihood multiplier                                                                                                                                                              |
| <u>scale</u> ( <i>x2</i>   dev   <i>#</i> ) | set the scale parameter; default is <code>scale(1)</code>                                                                                                                                |
| Reporting                                   |                                                                                                                                                                                          |
| <u>level</u> ( <i>#</i> )                   | set confidence level; default is <code>level(95)</code>                                                                                                                                  |
| <u>coefficients</u>                         | report nonexponentiated coefficients                                                                                                                                                     |
| <u>nocns</u> report                         | do not display constraints                                                                                                                                                               |
| <i>display_options</i>                      | control spacing and display of omitted variables and base and empty cells                                                                                                                |
| Maximization                                |                                                                                                                                                                                          |
| irls                                        | use iterated, reweighted least-squares optimization; the default                                                                                                                         |
| ml                                          | use maximum likelihood optimization                                                                                                                                                      |
| <i>maximize_options</i>                     | control the maximization process; seldom used                                                                                                                                            |
| fisher( <i>#</i> )                          | Fisher scoring steps                                                                                                                                                                     |
| search                                      | search for good starting values                                                                                                                                                          |
| <sup>†</sup> <u>coeflegend</u>              | display coefficients' legend instead of coefficient table                                                                                                                                |

<sup>†</sup>`coeflegend` does not appear in the dialog box.

`indepvars` may contain factor variables; see [U] 11.4.3 [Factor variables](#).

`depvar` and `indepvars` may contain time-series operators; see [U] 11.4.4 [Time-series varlists](#).

`bootstrap`, `by`, `jackknife`, `mi estimate`, `rolling`, and `statsby` are allowed; see [U] 11.1.10 [Prefix commands](#). `vce(bootstrap)`, `vce(jackknife)`, and `vce(jackknife1)` are not allowed with the `mi estimate` prefix.

Weights are not allowed with the `bootstrap` prefix.

`aweight`s are not allowed with the `jackknife` prefix.

`fweight`s, `aweight`s, `iweight`s, and `pweight`s are allowed; see [U] 11.1.6 [weight](#).

See [U] 20 [Estimation and postestimation commands](#) for more capabilities of estimation commands.

## Menu

Statistics > Generalized linear models > GLM for the binomial family

## Description

`binreg` fits generalized linear models for the binomial family. It estimates odds ratios, risk ratios, health ratios, and risk differences. The available links are

| Option          | Implied link                | Parameter                     |
|-----------------|-----------------------------|-------------------------------|
| <code>or</code> | <code>logit</code>          | odds ratios = $\exp(\beta)$   |
| <code>rr</code> | <code>log</code>            | risk ratios = $\exp(\beta)$   |
| <code>hr</code> | <code>log complement</code> | health ratios = $\exp(\beta)$ |
| <code>rd</code> | <code>identity</code>       | risk differences = $\beta$    |

Estimates of odds, risk, and health ratios are obtained by exponentiating the appropriate coefficients. The `or` option produces the same results as Stata’s `logistic` command, and `or coefficients` yields the same results as the `logit` command. When no link is specified, `or` is assumed.

## Options

Model

`noconstant`; see [R] [estimation options](#).

`or` requests the logit link and results in odds ratios if `coefficients` is not specified.

`rr` requests the log link and results in risk ratios if `coefficients` is not specified.

`hr` requests the log-complement link and results in health ratios if `coefficients` is not specified.

`rd` requests the identity link and results in risk differences.

`n(#|varname)` specifies either a constant integer to use as the denominator for the binomial family or a variable that holds the denominator for each observation.

`exposure(varname)`, `offset(varname)`, `constraints(constraints)`, `collinear`; see [R] [estimation options](#). `constraints(constraints)` and `collinear` are not allowed with `irls`.

`mu(varname)` specifies `varname` containing an initial estimate for the mean of `depvar`. This option can be useful if you encounter convergence difficulties. `init(varname)` is a synonym.

SE/Robust

`vce(vctype)` specifies the type of standard error reported, which includes types that are robust to some kinds of misspecification, that allow for intragroup correlation, that are derived from asymptotic theory, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

`vce(eim)`, the default, uses the expected information matrix (EIM) for the variance estimator.

`binreg` also allows the following:

`vce(hac kernel [#])` specifies that a heteroskedasticity- and autocorrelation-consistent (HAC) variance estimate be used. HAC refers to the general form for combining weighted matrices to form the variance estimate. There are three kernels built into `binreg`. `kernel` is a user-written program or one of

`nwest` | `gallant` | `anderson`

If `#` is not specified,  $N - 2$  is assumed.

`vce(jackknife1)` specifies that the one-step jackknife estimate of variance be used.

`vce(unbiased)` specifies that the unbiased sandwich estimate of variance be used.

`t(varname)` specifies the variable name corresponding to time; see [TS] [tsset](#). `binreg` does not always need to know `t()`, though it does if `vce(hac ...)` is specified. Then you can either specify the time variable with `t()`, or you can `tsset` your data before calling `binreg`. When the time variable is required, `binreg` assumes that the observations are spaced equally over time.

`vfactor(#)` specifies a scalar by which to multiply the resulting variance matrix. This option allows users to match output with other packages, which may apply degrees of freedom or other small-sample corrections to estimates of variance.

`disp(#)` multiplies the variance of `depvar` by `#` and divides the deviance by `#`. The resulting distributions are members of the quasiliikelihood family.

`scale(x2|dev|#)` overrides the default scale parameter. This option is allowed only with Hessian (information matrix) variance estimates.

By default, `scale(1)` is assumed for the discrete distributions (binomial, Poisson, and negative binomial), and `scale(x2)` is assumed for the continuous distributions (Gaussian, gamma, and inverse Gaussian).

`scale(x2)` specifies that the scale parameter be set to the Pearson chi-squared (or generalized chi-squared) statistic divided by the residual degrees of freedom, which was recommended by McCullagh and Nelder (1989) as a good general choice for continuous distributions.

`scale(dev)` sets the scale parameter to the deviance divided by the residual degrees of freedom. This option provides an alternative to `scale(x2)` for continuous distributions and overdispersed or underdispersed discrete distributions.

`scale(#)` sets the scale parameter to `#`.

Reporting

`level(#)`, `noconstant`; see [R] [estimation options](#).

`coefficients` displays the nonexponentiated coefficients and corresponding standard errors and confidence intervals. This option has no effect when the `rd` option is specified, because it always presents the nonexponentiated coefficients.

`nocnsreport`; see [R] [estimation options](#).

*display\_options*: noomitted, vsquish, noemptycells, baselevels, allbaselevels; see [R] [estimation options](#).

#### Maximization

*irls* requests iterated, reweighted least-squares (IRLS) optimization of the deviance instead of Newton–Raphson optimization of the log likelihood. This option is the default.

*ml* requests that optimization be carried out by using Stata’s *ml* command.

*maximize\_options*: technique(*algorithm\_spec*), [no]log, trace, gradient, showstep, hessian, showtolerance, difficult, iterate(#), tolerance(#), ltolerance(#), nrtolerance(#), nonrtolerance, from(*init\_specs*); see [R] [maximize](#). These options are seldom used.

Setting the optimization method to *ml*, with technique() set to something other than *BHHH*, changes the *vcetype* to *vce(oim)*. Specifying technique(*bhhh*) changes *vcetype* to *vce(opg)*.

*fisher*(#) specifies the number of Newton–Raphson steps that should use the Fisher scoring Hessian or EIM before switching to the observed information matrix (OIM). This option is available only if *ml* is specified and is useful only for Newton–Raphson optimization.

*search* specifies that the command search for good starting values. This option is available only if *ml* is specified and is useful only for Newton–Raphson optimization.

The following option is available with *binreg* but is not shown in the dialog box:

*coeflegend*; see [R] [estimation options](#).

## Remarks

[Wacholder \(1986\)](#) suggests methods for estimating risk ratios and risk differences from prospective binomial data. These estimates are obtained by selecting the proper link functions in the generalized linear-model framework. (See [Methods and formulas](#) for details; also see [R] [glm](#).)

### ► Example 1

[Wacholder \(1986\)](#) presents an example, using data from [Wright et al. \(1983\)](#), of an investigation of the relationship between alcohol consumption and the risk of a low-birthweight baby. Covariates examined included whether the mother smoked (yes or no), mother’s social class (three levels), and drinking frequency (light, moderate, or heavy). The data for the 18 possible categories determined by the covariates are illustrated below.

(Continued on next page)

Let’s first describe the data and list a few observations.

```
. use http://www.stata-press.com/data/r11/binreg
. list
```

|     | cat | d  | n   | alc | smo | soc |
|-----|-----|----|-----|-----|-----|-----|
| 1.  | 1   | 11 | 84  | 3   | 1   | 1   |
| 2.  | 2   | 5  | 79  | 2   | 1   | 1   |
| 3.  | 3   | 11 | 169 | 1   | 1   | 1   |
| 4.  | 4   | 6  | 28  | 3   | 2   | 1   |
| 5.  | 5   | 3  | 13  | 2   | 2   | 1   |
| 6.  | 6   | 1  | 26  | 1   | 2   | 1   |
| 7.  | 7   | 4  | 22  | 3   | 1   | 2   |
| 8.  | 8   | 3  | 25  | 2   | 1   | 2   |
| 9.  | 9   | 12 | 162 | 1   | 1   | 2   |
| 10. | 10  | 4  | 17  | 3   | 2   | 2   |
| 11. | 11  | 2  | 7   | 2   | 2   | 2   |
| 12. | 12  | 6  | 38  | 1   | 2   | 2   |
| 13. | 13  | 0  | 14  | 3   | 1   | 3   |
| 14. | 14  | 1  | 18  | 2   | 1   | 3   |
| 15. | 15  | 12 | 91  | 1   | 1   | 3   |
| 16. | 16  | 7  | 19  | 3   | 2   | 3   |
| 17. | 17  | 2  | 18  | 2   | 2   | 3   |
| 18. | 18  | 8  | 70  | 1   | 2   | 3   |

Each observation corresponds to one of the 18 covariate structures. The number of low-birthweight babies from n in each category is given by the d variable.

We begin by estimating risk ratios:

```
. binreg d i.soc i.alc i.smo, n(n) rr
Iteration 1: deviance = 14.2879
Iteration 2: deviance = 13.607
Iteration 3: deviance = 13.60503
Iteration 4: deviance = 13.60503
```

|                                     |               |                 |   |           |
|-------------------------------------|---------------|-----------------|---|-----------|
| Generalized linear models           |               | No. of obs      | = | 18        |
| Optimization : MQL Fisher scoring   |               | Residual df     | = | 12        |
| (IRLS EIM)                          |               | Scale parameter | = | 1         |
| Deviance                            | = 13.6050268  | (1/df) Deviance | = | 1.133752  |
| Pearson                             | = 11.51517095 | (1/df) Pearson  | = | .9595976  |
| Variance function: V(u) = u*(1-u/n) |               | [Binomial]      |   |           |
| Link function : g(u) = ln(u/n)      |               | [Log]           |   |           |
|                                     |               | BIC             | = | -21.07943 |

|       | d | EIM        |           | z    | P> z  | [95% Conf. Interval] |          |
|-------|---|------------|-----------|------|-------|----------------------|----------|
|       |   | Risk Ratio | Std. Err. |      |       |                      |          |
| soc   |   |            |           |      |       |                      |          |
|       | 2 | 1.340001   | .3127382  | 1.25 | 0.210 | .848098              | 2.11721  |
|       | 3 | 1.349487   | .3291488  | 1.23 | 0.219 | .8366715             | 2.176619 |
| alc   |   |            |           |      |       |                      |          |
|       | 2 | 1.191157   | .3265354  | 0.64 | 0.523 | .6960276             | 2.038503 |
|       | 3 | 1.974078   | .4261751  | 3.15 | 0.002 | 1.293011             | 3.013884 |
| 2.smo |   | 1.648444   | .332875   | 2.48 | 0.013 | 1.109657             | 2.448836 |

By default, Stata reports the risk ratios (the exponentiated regression coefficients) estimated by the model. We can see that the risk ratio comparing heavy drinkers with light drinkers, after adjusting for smoking and social class, is 1.974078. That is, mothers who drink heavily during their pregnancy have approximately twice the risk of delivering low-birthweight babies as mothers who are light drinkers.

The nonexponentiated coefficients can be obtained with the `coefficients` option:

```
. binreg d i.soc i.alc i.smo, n(n) rr coefficients

Iteration 1: deviance = 14.2879
Iteration 2: deviance = 13.607
Iteration 3: deviance = 13.60503
Iteration 4: deviance = 13.60503

Generalized linear models No. of obs = 18
Optimization : MQL Fisher scoring Residual df = 12
 (IRLS EIM) Scale parameter = 1
Deviance = 13.6050268 (1/df) Deviance = 1.133752
Pearson = 11.51517095 (1/df) Pearson = .9595976
Variance function: V(u) = u*(1-u/n) [Binomial]
Link function : g(u) = ln(u/n) [Log]
 BIC = -21.07943
```

| d     | EIM       |           | z      | P> z  | [95% Conf. Interval] |           |
|-------|-----------|-----------|--------|-------|----------------------|-----------|
|       | Coef.     | Std. Err. |        |       |                      |           |
| soc   |           |           |        |       |                      |           |
| 2     | .2926702  | .2333866  | 1.25   | 0.210 | -.1647591            | .7500994  |
| 3     | .2997244  | .2439066  | 1.23   | 0.219 | -.1783238            | .7777726  |
| alc   |           |           |        |       |                      |           |
| 2     | .1749248  | .274133   | 0.64   | 0.523 | -.362366             | .7122156  |
| 3     | .6801017  | .2158856  | 3.15   | 0.002 | .2569737             | 1.10323   |
| 2.smo | .4998317  | .2019329  | 2.48   | 0.013 | .1040505             | .8956129  |
| _cons | -2.764079 | .2031606  | -13.61 | 0.000 | -3.162266            | -2.365891 |

(Continued on next page)

Risk differences are obtained with the `rd` option:

```
. binreg d i.soc i.alc i.smo, n(n) rd
Iteration 1: deviance = 18.67277
Iteration 2: deviance = 14.94364
Iteration 3: deviance = 14.9185
Iteration 4: deviance = 14.91762
Iteration 5: deviance = 14.91758
Iteration 6: deviance = 14.91758
Iteration 7: deviance = 14.91758

Generalized linear models
Optimization : MQL Fisher scoring
 (IRLS EIM)
Deviance = 14.91758277
Pearson = 12.60353235
Variance function: V(u) = u*(1-u/n)
Link function : g(u) = u/n

No. of obs = 18
Residual df = 12
Scale parameter = 1
(1/df) Deviance = 1.243132
(1/df) Pearson = 1.050294
[Binomial]
[Identity]
BIC = -19.76688
```

| d     | EIM        |           | z    | P> z  | [95% Conf. Interval] |          |
|-------|------------|-----------|------|-------|----------------------|----------|
|       | Risk Diff. | Std. Err. |      |       |                      |          |
| soc   |            |           |      |       |                      |          |
| 2     | .0263817   | .0232124  | 1.14 | 0.256 | -.0191137            | .0718771 |
| 3     | .0365553   | .0268668  | 1.36 | 0.174 | -.0161026            | .0892132 |
| alc   |            |           |      |       |                      |          |
| 2     | .0122539   | .0257713  | 0.48 | 0.634 | -.0382569            | .0627647 |
| 3     | .0801291   | .0302878  | 2.65 | 0.008 | .020766              | .1394921 |
| 2.smo | .0542415   | .0270838  | 2.00 | 0.045 | .0011582             | .1073248 |
| _cons | .059028    | .0160693  | 3.67 | 0.000 | .0275327             | .0905232 |

The risk difference between heavy drinkers and light drinkers is simply the value of the coefficient for `3.alc` = 0.0801291. Because the risk differences are obtained directly from the coefficients estimated by using the identity link, the `coefficients` option has no effect here.

Health ratios are obtained with the `hr` option. The health ratios (exponentiated coefficients for the log-complement link) are reported directly.



```
. binreg d i.soc i.alc i.smo, n(n) hr
Iteration 1: deviance = 21.15233
Iteration 2: deviance = 15.16467
Iteration 3: deviance = 15.13205
Iteration 4: deviance = 15.13114
Iteration 5: deviance = 15.13111
Iteration 6: deviance = 15.13111
Iteration 7: deviance = 15.13111

Generalized linear models
Optimization : MQL Fisher scoring
 (IRLS EIM)
Deviance = 15.13110545
Pearson = 12.84203917
Variance function: V(u) = u*(1-u/n)
Link function : g(u) = ln(1-u/n)

No. of obs = 18
Residual df = 12
Scale parameter = 1
(1/df) Deviance = 1.260925
(1/df) Pearson = 1.07017
[Binomial]
[Log complement]
BIC = -19.55336
```

| d     | EIM      |           | z     | P> z  | [95% Conf. Interval] |          |
|-------|----------|-----------|-------|-------|----------------------|----------|
|       | HR       | Std. Err. |       |       |                      |          |
| soc   |          |           |       |       |                      |          |
| 2     | .9720541 | .024858   | -1.11 | 0.268 | .9245342             | 1.022017 |
| 3     | .9597182 | .0290412  | -1.36 | 0.174 | .9044535             | 1.01836  |
| alc   |          |           |       |       |                      |          |
| 2     | .9871517 | .0278852  | -0.46 | 0.647 | .9339831             | 1.043347 |
| 3     | .9134243 | .0325726  | -2.54 | 0.011 | .8517631             | .9795493 |
| 2.smo | .9409983 | .0296125  | -1.93 | 0.053 | .8847125             | 1.000865 |

(HR) Health ratios

To see the nonexponentiated coefficients, we can specify the `coefficients` option.



(Continued on next page)

## Saved results

binreg, irls saves the following in `e()`:

### Scalars

|                             |                                                     |
|-----------------------------|-----------------------------------------------------|
| <code>e(N)</code>           | number of observations                              |
| <code>e(k)</code>           | number of parameters                                |
| <code>e(k_eq_model)</code>  | number of equations in model Wald test              |
| <code>e(df_m)</code>        | model degrees of freedom                            |
| <code>e(df)</code>          | residual degrees of freedom                         |
| <code>e(phi)</code>         | model scale parameter                               |
| <code>e(disp)</code>        | dispersion parameter                                |
| <code>e(bic)</code>         | model BIC                                           |
| <code>e(N_clust)</code>     | number of clusters                                  |
| <code>e(deviance)</code>    | deviance                                            |
| <code>e(deviance_s)</code>  | scaled deviance                                     |
| <code>e(deviance_p)</code>  | Pearson deviance                                    |
| <code>e(deviance_ps)</code> | scaled Pearson deviance                             |
| <code>e(dispers)</code>     | dispersion                                          |
| <code>e(dispers_s)</code>   | scaled dispersion                                   |
| <code>e(dispers_p)</code>   | Pearson dispersion                                  |
| <code>e(dispers_ps)</code>  | scaled Pearson dispersion                           |
| <code>e(vf)</code>          | factor set by <code>vfactor()</code> , 1 if not set |
| <code>e(rank)</code>        | rank of <code>e(V)</code>                           |
| <code>e(rc)</code>          | return code                                         |

## Macros

|                 |                                             |
|-----------------|---------------------------------------------|
| e(cmd)          | binreg                                      |
| e(cmdline)      | command as typed                            |
| e(depvar)       | name of dependent variable                  |
| e(eform)        | eform() option implied by or, rr, hr, or rd |
| e(varfunc)      | name of variance function used              |
| e(varfunct)     | Binomial                                    |
| e(varfuncf)     | variance function                           |
| e(link)         | link function used by glm                   |
| e(linkt)        | link title                                  |
| e(linkf)        | link form                                   |
| e(m)            | number of binomial trials                   |
| e(wtype)        | weight type                                 |
| e(wexp)         | weight expression                           |
| e(title_fl)     | family-link title                           |
| e(clustvar)     | name of cluster variable                    |
| e(offset)       | offset                                      |
| e(const)        | noconstant or not set                       |
| e(hac_kernel)   | HAC kernel                                  |
| e(hac_lag)      | HAC lag                                     |
| e(vce)          | vcetype specified in vce()                  |
| e(vcetype)      | title used to label Std. Err.               |
| e(opt)          | type of optimization                        |
| e(opt1)         | optimization title, line 1                  |
| e(opt2)         | optimization title, line 2                  |
| e(crittype)     | optimization criterion                      |
| e(properties)   | b V                                         |
| e(predict)      | program used to implement predict           |
| e(marginsnotok) | predictions disallowed by margins           |
| e(asbalanced)   | factor variables fvset as asbalanced        |
| e(asobserved)   | factor variables fvset as asobserved        |

## Matrices

|                 |                                              |
|-----------------|----------------------------------------------|
| e(b)            | coefficient vector                           |
| e(V)            | variance-covariance matrix of the estimators |
| e(V_modelbased) | model-based variance                         |

## Functions

|           |                         |
|-----------|-------------------------|
| e(sample) | marks estimation sample |
|-----------|-------------------------|

(Continued on next page)

`binreg, ml` saves the following in `e()`:

Scalars

|                             |                                                     |
|-----------------------------|-----------------------------------------------------|
| <code>e(N)</code>           | number of observations                              |
| <code>e(k)</code>           | number of parameters                                |
| <code>e(k_eq)</code>        | number of equations in <code>e(b)</code>            |
| <code>e(k_eq_model)</code>  | number of equations in model Wald test              |
| <code>e(k_dv)</code>        | number of dependent variables                       |
| <code>e(k_autoCns)</code>   | number of base, empty, and omitted constraints      |
| <code>e(df_m)</code>        | model degrees of freedom                            |
| <code>e(df)</code>          | residual degrees of freedom                         |
| <code>e(phi)</code>         | model scale parameter                               |
| <code>e(aic)</code>         | model AIC, if <code>ml</code>                       |
| <code>e(bic)</code>         | model BIC                                           |
| <code>e(ll)</code>          | log likelihood, if <code>ml</code>                  |
| <code>e(N_clust)</code>     | number of clusters                                  |
| <code>e(chi2)</code>        | $\chi^2$                                            |
| <code>e(p)</code>           | significance of model test                          |
| <code>e(deviance)</code>    | deviance                                            |
| <code>e(deviance_s)</code>  | scaled deviance                                     |
| <code>e(deviance_p)</code>  | Pearson deviance                                    |
| <code>e(deviance_ps)</code> | scaled Pearson deviance                             |
| <code>e(dispers)</code>     | dispersion                                          |
| <code>e(dispers_s)</code>   | scaled dispersion                                   |
| <code>e(dispers_p)</code>   | Pearson dispersion                                  |
| <code>e(dispers_ps)</code>  | scaled Pearson dispersion                           |
| <code>e(vf)</code>          | factor set by <code>vfactor()</code> , 1 if not set |
| <code>e(rank)</code>        | rank of <code>e(V)</code>                           |
| <code>e(ic)</code>          | number of iterations                                |
| <code>e(rc)</code>          | return code                                         |
| <code>e(converged)</code>   | 1 if converged, 0 otherwise                         |

## Macros

|                    |                                                                          |
|--------------------|--------------------------------------------------------------------------|
| e(cmd)             | binreg                                                                   |
| e(cmdline)         | command as typed                                                         |
| e(depvar)          | name of dependent variable                                               |
| e(eform)           | eform() option implied by or, rr, hr, or rd                              |
| e(varfunc)         | name of variance function used                                           |
| e(varfunct)        | Binomial                                                                 |
| e(varfunctf)       | variance function                                                        |
| e(link)            | link function used by glm                                                |
| e(linkt)           | link title                                                               |
| e(linkf)           | link form                                                                |
| e(m)               | number of binomial trials                                                |
| e(wtype)           | weight type                                                              |
| e(wexp)            | weight expression                                                        |
| e(title)           | title in estimation output                                               |
| e(title_fl)        | family-link title                                                        |
| e(clustvar)        | name of cluster variable                                                 |
| e(offset)          | offset                                                                   |
| e(cons)            | noconstant or not set                                                    |
| e(hac_kernel)      | HAC kernel                                                               |
| e(hac_lag)         | HAC lag                                                                  |
| e(chi2type)        | LR; type of model $\chi^2$ test                                          |
| e(vce)             | vcetype specified in vce()                                               |
| e(vcetype)         | title used to label Std. Err.                                            |
| e(opt)             | type of optimization                                                     |
| e(opt1)            | optimization title, line 1                                               |
| e(which)           | max or min; whether optimizer is to perform maximization or minimization |
| e(ml_method)       | type of ml method                                                        |
| e(user)            | name of likelihood-evaluator program                                     |
| e(technique)       | maximization technique                                                   |
| e(singularHmethod) | m-marquardt or hybrid; method used when Hessian is singular              |
| e(crittype)        | optimization criterion                                                   |
| e(properties)      | b V                                                                      |
| e(predict)         | program used to implement predict                                        |
| e(marginsnotok)    | predictions disallowed by margins                                        |
| e(asbalanced)      | factor variables fvset as asbalanced                                     |
| e(asobserved)      | factor variables fvset as asobserved                                     |

## Matrices

|                 |                                              |
|-----------------|----------------------------------------------|
| e(b)            | coefficient vector                           |
| e(Cns)          | constraints matrix                           |
| e(ilog)         | iteration log (up to 20 iterations)          |
| e(gradient)     | gradient vector                              |
| e(V)            | variance-covariance matrix of the estimators |
| e(V_modelbased) | model-based variance                         |

## Functions

|           |                         |
|-----------|-------------------------|
| e(sample) | marks estimation sample |
|-----------|-------------------------|

## Methods and formulas

`binreg` is implemented as an ado-file.

Let  $\pi_i$  be the probability of success for the  $i$ th observation,  $i = 1, \dots, N$ , and let  $X\beta$  be the linear predictor. The link function relates the covariates of each observation to its respective probability through the linear predictor.

In logistic regression, the logit link is used:

$$\ln\left(\frac{\pi}{1-\pi}\right) = X\beta$$

The regression coefficient  $\beta_k$  represents the change in the logarithm of the odds associated with a one-unit change in the value of the  $X_k$  covariate; thus  $\exp(\beta_k)$  is the ratio of the odds associated with a change of one unit in  $X_k$ .

For risk differences, the identity link  $\pi = X\beta$  is used. The regression coefficient  $\beta_k$  represents the risk difference associated with a change of one unit in  $X_k$ . When using the identity link, you can obtain fitted probabilities outside the interval  $(0, 1)$ . As suggested by Wacholder, at each iteration, fitted probabilities are checked for range conditions (and put back in range if necessary). For example, if the identity link results in a fitted probability that is smaller than  $1e-4$ , the probability is replaced with  $1e-4$  before the link function is calculated.

A similar adjustment is made for the logarithmic link, which is used for estimating the risk ratio,  $\ln(\pi) = X\beta$ , where  $\exp(\beta_k)$  is the risk ratio associated with a change of one unit in  $X_k$ , and for the log-complement link used to estimate the probability of no disease or health, where  $\exp(\beta_k)$  represents the “health ratio” associated with a change of one unit in  $X_k$ .

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

## References

- Hardin, J. W., and M. A. Cleves. 1999. [sbe29: Generalized linear models: Extensions to the binomial family](#). *Stata Technical Bulletin* 50: 21–25. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 140–146. College Station, TX: Stata Press.
- Kleinbaum, D. G., and M. Klein. 2002. [Logistic Regression: A Self-Learning Text](#). 2nd ed. New York: Springer.
- McCullagh, P., and J. A. Nelder. 1989. [Generalized Linear Models](#). 2nd ed. London: Chapman & Hall/CRC.
- Wacholder, S. 1986. Binomial regression in GLIM: Estimating risk ratios and risk differences. *American Journal of Epidemiology* 123: 174–184.
- Wright, J. T., I. G. Barrison, I. G. Lewis, K. D. MacRae, E. J. Waterson, P. J. Toplis, M. G. Gordon, N. F. Morris, and I. M. Murray-Lyon. 1983. Alcohol consumption, pregnancy and low birthweight. *Lancet* 1: 663–665.

## Also see

- [R] [binreg postestimation](#) — Postestimation tools for `binreg`
- [R] [glm](#) — Generalized linear models
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `binreg`:

| command                | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>     | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estimates</code> | cataloging estimation results                                                                       |
| <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>  | link test for model specification                                                                   |
| <code>margins</code>   | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>   | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code> | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details.

# Syntax for predict

```
predict [type] newvar [if] [in] [, statistic options]
```

| statistic           | description                                                                                 |
|---------------------|---------------------------------------------------------------------------------------------|
| Main                |                                                                                             |
| <u>mu</u>           | expected value of $y$ ; the default                                                         |
| <u>xb</u>           | linear prediction $\eta = \mathbf{x}\hat{\beta}$                                            |
| <u>eta</u>          | synonym for <code>xb</code>                                                                 |
| <u>stdp</u>         | standard error of the linear prediction                                                     |
| <u>anscombe</u>     | <a href="#">Anscombe (1953)</a> residuals                                                   |
| <u>cooksd</u>       | Cook's distance                                                                             |
| <u>deviance</u>     | deviance residuals                                                                          |
| <u>hat</u>          | diagonals of the “hat” matrix as an analog to simple linear regression                      |
| <u>likelihood</u>   | weighted average of the standardized deviance and standard Pearson residuals                |
| <u>pearson</u>      | Pearson residuals                                                                           |
| <u>response</u>     | differences between the observed and fitted outcomes                                        |
| <u>score</u>        | first derivative of the log likelihood with respect to $\mathbf{x}_j\beta$                  |
| <u>working</u>      | working residuals                                                                           |
|                     |                                                                                             |
| options             | description                                                                                 |
| Options             |                                                                                             |
| <u>nooffset</u>     | modify calculations to ignore the offset variable                                           |
| <u>standardized</u> | multiply residual by the factor $(1 - h)^{1/2}$                                             |
| <u>studentized</u>  | multiply residual by one over the square root of the estimated scale parameter              |
| <u>modified</u>     | modify denominator of residual to be a reasonable estimate of the variance of $depvar$      |
| <u>adjusted</u>     | adjust deviance residual to make the convergence to the limiting normal distribution faster |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

- `mu`, the default, specifies that `predict` calculate the expected value of  $y$ , equal to  $g^{-1}(\mathbf{x}\hat{\beta})$  [ $ng^{-1}(\mathbf{x}\hat{\beta})$  for the binomial family].
- `xb` calculates the linear prediction  $\eta = \mathbf{x}\hat{\beta}$ .
- `eta` is a synonym for `xb`.



`stdp` calculates the standard error of the linear prediction.

`anscombe` calculates the [Anscombe \(1953\)](#) residuals to produce residuals that closely follow a normal distribution.

`cooksd` calculates Cook’s distance, which measures the aggregate change in the estimated coefficients when each observation is left out of the estimation.

`deviance` calculates the deviance residuals, which are recommended by [McCullagh and Nelder \(1989\)](#) and others as having the best properties for examining goodness of fit of a GLM. They are approximately normally distributed if the model is correct and may be plotted against the fitted values or against a covariate to inspect the model’s fit. Also see the `pearson` option below.

`hat` calculates the diagonals of the “hat” matrix as an analog to simple linear regression.

`likelihood` calculates a weighted average of the standardized deviance and standardized Pearson (described below) residuals.

`pearson` calculates the Pearson residuals, which often have markedly skewed distributions for nonnormal family distributions. Also see the `deviance` option above.

`response` calculates the differences between the observed and fitted outcomes.

`score` calculates the equation-level score,  $\partial \ln L / \partial (\mathbf{x}_j \boldsymbol{\beta})$ .

`working` calculates the working residuals, which are response residuals weighted according to the derivative of the link function.

#### Options

`nooffset` is relevant only if you specified `offset(varname)` for `binreg`. It modifies the calculations made by `predict` so that they ignore the offset variable; the linear prediction is treated as  $\mathbf{x}_j \mathbf{b}$  rather than as  $\mathbf{x}_j \mathbf{b} + \text{offset}_j$ .

`standardized` requests that the residual be multiplied by the factor  $(1 - h)^{-1/2}$ , where  $h$  is the diagonal of the hat matrix. This step is done to take into account the correlation between `devar` and its predicted value.

`studentized` requests that the residual be multiplied by one over the square root of the estimated scale parameter.

`modified` requests that the denominator of the residual be modified to be a reasonable estimate of the variance of `devar`. The base residual is multiplied by the factor  $(k/w)^{-1/2}$ , where  $k$  is either one or the user-specified dispersion parameter and  $w$  is the specified weight (or one if left unspecified).

`adjusted` adjusts the deviance residual to make the convergence to the limiting normal distribution faster. The adjustment deals with adding to the deviance residual a higher-order term depending on the variance function family. This option is allowed only when `deviance` is specified.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## References

- Anscombe, F. J. 1953. Contribution of discussion paper by H. Hotelling “New light on the correlation coefficient and its transforms”. *Journal of the Royal Statistical Society, Series B* 15: 229–230.
- McCullagh, P., and J. A. Nelder. 1989. *Generalized Linear Models*. 2nd ed. London: Chapman & Hall/CRC.

## Also see

[R] [binreg](#) — Generalized linear models: Extensions to the binomial family

[U] [20 Estimation and postestimation commands](#)

Syntax

*Bivariate probit model*

```
biprobit depvar1 depvar2 [varlist] [if] [in] [weight] [, options]
```

*Seemingly unrelated bivariate probit model*

```
biprobit equation1 equation2 [if] [in] [weight] [, su_options]
```

where *equation*<sub>1</sub> and *equation*<sub>2</sub> are specified as

```
([eqname:] depvar [=] [varlist] [, noconstant offset(varname)])
```

| <i>options</i>                            | description                                                                                                             |
|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| Model                                     |                                                                                                                         |
| <u>noconstant</u>                         | suppress constant term                                                                                                  |
| <u>partial</u>                            | fit partial observability model                                                                                         |
| <u>offset1</u> ( <i>varname</i> )         | offset variable for first equation                                                                                      |
| <u>offset2</u> ( <i>varname</i> )         | offset variable for second equation                                                                                     |
| <u>constraints</u> ( <i>constraints</i> ) | apply specified linear constraints                                                                                      |
| <u>collinear</u>                          | keep collinear variables                                                                                                |
| SE/Robust                                 |                                                                                                                         |
| <u>vce</u> ( <i>vcetype</i> )             | <i>vcetype</i> may be oim, <u>robust</u> , <u>cluster</u> <i>clustvar</i> , opg, <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting                                 |                                                                                                                         |
| <u>level</u> (#)                          | set confidence level; default is level(95)                                                                              |
| <u>noskip</u>                             | perform likelihood-ratio test                                                                                           |
| <u>nocnsreport</u>                        | do not display constraints                                                                                              |
| <u>display_options</u>                    | control spacing and display of omitted variables and base and empty cells                                               |
| Maximization                              |                                                                                                                         |
| <u>maximize_options</u>                   | control the maximization process; seldom used                                                                           |
| <sup>†</sup> <u>coeflegend</u>            | display coefficients' legend instead of coefficient table                                                               |

| <i>su_options</i>                     | description                                                                                                                                                          |
|---------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                 |                                                                                                                                                                      |
| <code>partial</code>                  | fit partial observability model                                                                                                                                      |
| <code>constraints(constraints)</code> | apply specified linear constraints                                                                                                                                   |
| <code>collinear</code>                | keep collinear variables                                                                                                                                             |
| SE/Robust                             |                                                                                                                                                                      |
| <code>vce(vctype)</code>              | <i>vcetype</i> may be <code>oim</code> , <code>robust</code> , <code>cluster clustvar</code> , <code>opg</code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                             |                                                                                                                                                                      |
| <code>level(#)</code>                 | set confidence level; default is <code>level(95)</code>                                                                                                              |
| <code>noskip</code>                   | perform likelihood-ratio test                                                                                                                                        |
| <code>nocnsreport</code>              | do not display constraints                                                                                                                                           |
| <code>display_options</code>          | control spacing and display of omitted variables and base and empty cells                                                                                            |
| Maximization                          |                                                                                                                                                                      |
| <code>maximize_options</code>         | control the maximization process; seldom used                                                                                                                        |
| <sup>†</sup> <code>coeflegend</code>  | display coefficients' legend instead of coefficient table                                                                                                            |

<sup>†</sup>`coeflegend` does not appear in the dialog box.

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar1*, *depvar2*, *varlist*, and *depvar* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

`bootstrap`, `by`, `jackknife`, `rolling`, `statsby`, and `svy` are allowed; see [U] 11.1.10 Prefix commands.

Weights are not allowed with the `bootstrap` prefix.

`vce()`, `noskip`, and weights are not allowed with the `svy` prefix.

`pweights`, `fweights`, and `iweights` are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

## Menu

### biprobit

Statistics > Binary outcomes > Bivariate probit regression

### seemingly unrelated biprobit

Statistics > Binary outcomes > Seemingly unrelated bivariate probit regression

## Description

`biprobit` fits maximum-likelihood two-equation probit models—either a bivariate probit or a seemingly unrelated probit (limited to two equations).

## Options

### Model

`noconstant`; see [R] [estimation options](#).

`partial` specifies that the partial observability model be fit. This particular model commonly has poor convergence properties, so we recommend that you use the `difficult` option if you want to fit the Poirier partial observability model; see [R] [ml](#).

This model computes the product of the two dependent variables so that you do not have to replace each with the product.

`offset1(varname)`, `offset2(varname)`, `constraints(constraints)`, `collinear`; see [R] [estimation options](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

### Reporting

`level(#)`; see [R] [estimation options](#).

`noskip` specifies that a full maximum-likelihood model with only a constant for the regression equation be fit. This model is not displayed but is used as the base model to compute a likelihood-ratio test for the model test statistic displayed in the estimation header. By default, the overall model test statistic is an asymptotically equivalent Wald test of all the parameters in the regression equation being zero (except the constant). For many models, this option can substantially increase estimation time.

`nocnsreport`; see [R] [estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [R] [maximize](#). These options are seldom used.

Setting the optimization type to `technique(bhhh)` resets the default `vcetype` to `vce(opg)`.

The following option is available with `biprobit` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

For a good introduction to the bivariate probit models, see [Greene \(2008, 817–826\)](#) and [Pindyck and Rubinfeld \(1998\)](#). [Poirier \(1980\)](#) explains the partial observability model. [Van de Ven and Van Praag \(1981\)](#) explain the probit model with sample selection; see [R] [heckprob](#) for details.

➤ Example 1

We use the data from [Pindyck and Rubinfeld \(1998, 332\)](#). In this dataset, the variables are whether children attend private school (`private`), number of years the family has been at the present residence (`years`), log of property tax (`logptax`), log of income (`loginc`), and whether the head of the household voted for an increase in property taxes (`vote`).

We wish to model the bivariate outcomes of whether children attend private school and whether the head of the household voted for an increase in property tax based on the other covariates.

```
. use http://www.stata-press.com/data/r11/school
. biprobit private vote years logptax loginc
Fitting comparison equation 1:
Iteration 0: log likelihood = -31.967097
Iteration 1: log likelihood = -31.452424
Iteration 2: log likelihood = -31.448958
Iteration 3: log likelihood = -31.448958
Fitting comparison equation 2:
Iteration 0: log likelihood = -63.036914
Iteration 1: log likelihood = -58.534843
Iteration 2: log likelihood = -58.497292
Iteration 3: log likelihood = -58.497288
Comparison: log likelihood = -89.946246
Fitting full model:
Iteration 0: log likelihood = -89.946246
Iteration 1: log likelihood = -89.258897
Iteration 2: log likelihood = -89.254028
Iteration 3: log likelihood = -89.254028
Bivariate probit regression
Log likelihood = -89.254028
Number of obs = 95
Wald chi2(6) = 9.59
Prob > chi2 = 0.1431
```

|         | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|---------|-----------|-----------|-------|-------|----------------------|-----------|
| private |           |           |       |       |                      |           |
| years   | -.0118884 | .0256778  | -0.46 | 0.643 | -.0622159            | .0384391  |
| logptax | -.1066962 | .6669782  | -0.16 | 0.873 | -1.413949            | 1.200557  |
| loginc  | .3762037  | .5306484  | 0.71  | 0.478 | -.663848             | 1.416255  |
| _cons   | -4.184694 | 4.837817  | -0.86 | 0.387 | -13.66664            | 5.297253  |
| vote    |           |           |       |       |                      |           |
| years   | -.0168561 | .0147834  | -1.14 | 0.254 | -.0458309            | .0121188  |
| logptax | -1.288707 | .5752266  | -2.24 | 0.025 | -2.416131            | -.1612839 |
| loginc  | .998286   | .4403565  | 2.27  | 0.023 | .1352031             | 1.861369  |
| _cons   | -.5360573 | 4.068509  | -0.13 | 0.895 | -8.510188            | 7.438073  |
| /athrho | -.2764525 | .2412099  | -1.15 | 0.252 | -.7492153            | .1963102  |
| rho     | -.2696186 | .2236753  |       |       | -.6346806            | .1938267  |

```
Likelihood-ratio test of rho=0: chi2(1) = 1.38444 Prob > chi2 = 0.2393
```

The output shows several iteration logs. The first iteration log corresponds to running the univariate probit model for the first equation, and the second log corresponds to running the univariate probit for the second model. If  $\rho = 0$ , the sum of the log likelihoods from these two models will equal the log likelihood of the bivariate probit model; this sum is printed in the iteration log as the comparison log likelihood.

The final iteration log is for fitting the full bivariate probit model. A likelihood-ratio test of the log likelihood for this model and the comparison log likelihood is presented at the end of the output. If we had specified the `vce(robust)` option, this test would be presented as a Wald test instead of as a likelihood-ratio test.

We could have fit the same model by using the seemingly unrelated syntax as

```
. biprobit (private=years logptax loginc) (vote=years logptax loginc)
```



## Saved results

`biprobit` saves the following in `e()`:

### Scalars

|                            |                                                   |
|----------------------------|---------------------------------------------------|
| <code>e(N)</code>          | number of observations                            |
| <code>e(k)</code>          | number of parameters                              |
| <code>e(k_eq)</code>       | number of equations                               |
| <code>e(k_aux)</code>      | number of auxiliary parameters                    |
| <code>e(k_eq_model)</code> | number of equations in model Wald test            |
| <code>e(k_dv)</code>       | number of dependent variables                     |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints    |
| <code>e(df_m)</code>       | model degrees of freedom                          |
| <code>e(ll)</code>         | log likelihood                                    |
| <code>e(ll_0)</code>       | log likelihood, constant-only model (noskip only) |
| <code>e(ll_c)</code>       | log likelihood, comparison model                  |
| <code>e(N_clust)</code>    | number of clusters                                |
| <code>e(chi2)</code>       | $\chi^2$                                          |
| <code>e(chi2_c)</code>     | $\chi^2$ for comparison test                      |
| <code>e(p)</code>          | significance                                      |
| <code>e(rho)</code>        | $\rho$                                            |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                         |
| <code>e(rank0)</code>      | rank of <code>e(V)</code> for constant-only model |
| <code>e(ic)</code>         | number of iterations                              |
| <code>e(rc)</code>         | return code                                       |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                       |

*(Continued on next page)*

**Macros**

|                                 |                                                                                 |
|---------------------------------|---------------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>biprobit</code>                                                           |
| <code>e(cmdline)</code>         | command as typed                                                                |
| <code>e(depvar)</code>          | names of dependent variables                                                    |
| <code>e(wtype)</code>           | weight type                                                                     |
| <code>e(wexp)</code>            | weight expression                                                               |
| <code>e(title)</code>           | title in estimation output                                                      |
| <code>e(clustvar)</code>        | name of cluster variable                                                        |
| <code>e(offset1)</code>         | offset for first equation                                                       |
| <code>e(offset2)</code>         | offset for second equation                                                      |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                         |
| <code>e(chi2_ct)</code>         | Wald or LR; type of model $\chi^2$ test corresponding to <code>e(chi2_c)</code> |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                                  |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                                   |
| <code>e(diparm#)</code>         | display transformed parameter #                                                 |
| <code>e(opt)</code>             | type of optimization                                                            |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization        |
| <code>e(ml_method)</code>       | type of ml method                                                               |
| <code>e(user)</code>            | name of likelihood-evaluator program                                            |
| <code>e(technique)</code>       | maximization technique                                                          |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular                     |
| <code>e(crittype)</code>        | optimization criterion                                                          |
| <code>e(properties)</code>      | b v                                                                             |
| <code>d(predict)</code>         | program used to implement <code>predict</code>                                  |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>                  |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>                  |

**Matrices**

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(gradient)</code>     | gradient vector                              |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

**Functions**

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`biprobit` is implemented as an ado-file.



The log likelihood,  $\ln L$ , is given by

$$\begin{aligned}\xi_j^\beta &= x_j\beta + \text{offset}_j^\beta \\ \xi_j^\gamma &= z_j\gamma + \text{offset}_j^\gamma \\ q_{1j} &= \begin{cases} 1 & \text{if } y_{1j} \neq 0 \\ -1 & \text{otherwise} \end{cases} \\ q_{2j} &= \begin{cases} 1 & \text{if } y_{2j} \neq 0 \\ -1 & \text{otherwise} \end{cases} \\ \rho_j^* &= q_{1j}q_{2j}\rho \\ \ln L &= \sum_{j=1}^n w_j \ln \Phi_2 \left( q_{1j}\xi_j^\beta, q_{2j}\xi_j^\gamma, \rho_j^* \right)\end{aligned}$$

where  $\Phi_2()$  is the cumulative bivariate normal distribution function (with mean  $[0 \ 0]'$ ) and  $w_j$  is an optional weight for observation  $j$ . This derivation assumes that

$$\begin{aligned}y_{1j}^* &= x_j\beta + \epsilon_{1j} + \text{offset}_j^\beta \\ y_{2j}^* &= z_j\gamma + \epsilon_{2j} + \text{offset}_j^\gamma \\ E(\epsilon_1) &= E(\epsilon_2) = 0 \\ \text{Var}(\epsilon_1) &= \text{Var}(\epsilon_2) = 1 \\ \text{Cov}(\epsilon_1, \epsilon_2) &= \rho\end{aligned}$$

where  $y_{1j}^*$  and  $y_{2j}^*$  are the unobserved latent variables; instead, we observe only  $y_{ij} = 1$  if  $y_{ij}^* > 0$  and  $y_{ij} = 0$  otherwise (for  $i = 1, 2$ ).

In the maximum likelihood estimation,  $\rho$  is not directly estimated, but  $\text{atanh } \rho$  is

$$\text{atanh } \rho = \frac{1}{2} \ln \left( \frac{1 + \rho}{1 - \rho} \right)$$

From the form of the likelihood, if  $\rho = 0$ , then the log likelihood for the bivariate probit models is equal to the sum of the log likelihoods of the two univariate probit models. A likelihood-ratio test may therefore be performed by comparing the likelihood of the full bivariate model with the sum of the log likelihoods for the univariate probit models.

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

`biprobit` also supports estimation with survey data. For details on VCEs with survey data, see [\[SVY\] variance estimation](#).

## References

- De Luca, G. 2008. [SNP and SML estimation of univariate and bivariate binary-choice models](#). *Stata Journal* 8: 190–220.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- Hardin, J. W. 1996. [sg61: Bivariate probit models](#). *Stata Technical Bulletin* 33: 15–20. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 152–158. College Station, TX: Stata Press.

Heckman, J. 1979. Sample selection bias as a specification error. *Econometrica* 47: 153–161.

Pindyck, R. S., and D. L. Rubinfeld. 1998. *Econometric Models and Economic Forecasts*. 4th ed. New York: McGraw–Hill.

Poirier, D. J. 1980. Partial observability in bivariate probit models. *Journal of Econometrics* 12: 209–217.

Van de Ven, W. P. M. M., and B. M. S. Van Pragg. 1981. The demand for deductibles in private health insurance: A probit model with sample selection. *Journal of Econometrics* 17: 229–252.

## Also see

[R] **biprobit postestimation** — Postestimation tools for biprobit

[R] **mprobit** — Multinomial probit regression

[R] **probit** — Probit regression

[SVY] **svy estimation** — Estimation commands for survey data

[U] **20 Estimation and postestimation commands**

Description

The following postestimation commands are available for `biprobit`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

## Syntax for predict

```
predict [type] newvar [if] [in] [, statistic nooffset]

predict [type] { stub* | newvareq1 newvareq2 newvarathrho } [if] [in] , scores
```

| statistic | description                                                                                                                                     |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| Main      |                                                                                                                                                 |
| p11       | $\Phi_2(\mathbf{x}_j\mathbf{b}, \mathbf{z}_j\mathbf{g}, \rho)$ , predicted probability $\Pr(y_{1j} = 1, y_{2j} = 1)$ ; the default              |
| p10       | $\Phi_2(\mathbf{x}_j\mathbf{b}, -\mathbf{z}_j\mathbf{g}, -\rho)$ , predicted probability $\Pr(y_{1j} = 1, y_{2j} = 0)$                          |
| p01       | $\Phi_2(-\mathbf{x}_j\mathbf{b}, \mathbf{z}_j\mathbf{g}, -\rho)$ , predicted probability $\Pr(y_{1j} = 0, y_{2j} = 1)$                          |
| p00       | $\Phi_2(-\mathbf{x}_j\mathbf{b}, -\mathbf{z}_j\mathbf{g}, \rho)$ , predicted probability $\Pr(y_{1j} = 0, y_{2j} = 0)$                          |
| pmarg1    | $\Phi(\mathbf{x}_j\mathbf{b})$ , marginal success probability for equation 1                                                                    |
| pmarg2    | $\Phi(\mathbf{z}_j\mathbf{g})$ , marginal success probability for equation 2                                                                    |
| pcond1    | $\Phi_2(\mathbf{x}_j\mathbf{b}, \mathbf{z}_j\mathbf{g}, \rho)/\Phi(\mathbf{z}_j\mathbf{g})$ , conditional probability of success for equation 1 |
| pcond2    | $\Phi_2(\mathbf{x}_j\mathbf{b}, \mathbf{z}_j\mathbf{g}, \rho)/\Phi(\mathbf{x}_j\mathbf{b})$ , conditional probability of success for equation 2 |
| xb1       | $\mathbf{x}_j\mathbf{b}$ , linear prediction for equation 1                                                                                     |
| xb2       | $\mathbf{z}_j\mathbf{g}$ , linear prediction for equation 2                                                                                     |
| stdp1     | standard error of the linear prediction for equation 1                                                                                          |
| stdp2     | standard error of the linear prediction for equation 2                                                                                          |

where  $\Phi()$  is the standard normal-distribution function and  $\Phi_2()$  is the bivariate standard normal-distribution function.

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

|                                                                                                                                               |
|-----------------------------------------------------------------------------------------------------------------------------------------------|
| Main                                                                                                                                          |
| p11, the default, calculates the bivariate predicted probability $\Pr(y_{1j} = 1, y_{2j} = 1)$ .                                              |
| p10 calculates the bivariate predicted probability $\Pr(y_{1j} = 1, y_{2j} = 0)$ .                                                            |
| p01 calculates the bivariate predicted probability $\Pr(y_{1j} = 0, y_{2j} = 1)$ .                                                            |
| p00 calculates the bivariate predicted probability $\Pr(y_{1j} = 0, y_{2j} = 0)$ .                                                            |
| pmarg1 calculates the univariate (marginal) predicted probability of success $\Pr(y_{1j} = 1)$ .                                              |
| pmarg2 calculates the univariate (marginal) predicted probability of success $\Pr(y_{2j} = 1)$ .                                              |
| pcond1 calculates the conditional (on success in equation 2) predicted probability of success $\Pr(y_{1j} = 1, y_{2j} = 1)/\Pr(y_{2j} = 1)$ . |
| pcond2 calculates the conditional (on success in equation 1) predicted probability of success $\Pr(y_{1j} = 1, y_{2j} = 1)/\Pr(y_{1j} = 1)$ . |

`xb1` calculates the probit linear prediction  $\mathbf{x}_j\mathbf{b}$ .

`xb2` calculates the probit linear prediction  $\mathbf{z}_j\mathbf{g}$ .

`stdp1` calculates the standard error of the linear prediction for equation 1.

`stdp2` calculates the standard error of the linear prediction for equation 2.

`nooffset` is relevant only if you specified `offset1(varname)` or `offset2(varname)` for `biprobit`.

It modifies the calculations made by `predict` so that they ignore the offset variables; the linear predictions are treated as  $\mathbf{x}_j\mathbf{b}$  rather than as  $\mathbf{x}_j\mathbf{b} + \text{offset}_{1j}$  and  $\mathbf{z}_j\boldsymbol{\gamma}$  rather than as  $\mathbf{z}_j\boldsymbol{\gamma} + \text{offset}_{2j}$ .

`scores` calculates equation-level score variables.

The first new variable will contain  $\partial \ln L / \partial (\mathbf{x}_j\boldsymbol{\beta})$ .

The second new variable will contain  $\partial \ln L / \partial (\mathbf{z}_j\boldsymbol{\gamma})$ .

The third new variable will contain  $\partial \ln L / \partial (\text{atanh } \rho)$ .

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [biprobit](#) — Bivariate probit regression

[U] [20 Estimation and postestimation commands](#)

# Title

bittest — Binomial probability test

## Syntax

*Binomial probability test*

```
bittest varname== #p [if] [in] [weight] [, detail]
```

*Immediate form of binomial probability test*

```
bittesti #N #succ #p [, detail]
```

*by* is allowed with `bittest`; see [D] [by](#).

*fweights* are allowed with `bittest`; see [U] [11.1.6 weight](#).

## Menu

### bittest

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > Binomial probability test

### bittesti

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > Binomial probability test calculator

## Description

`bittest` performs exact hypothesis tests for binomial random variables. The null hypothesis is that the probability of a success on a trial is  $\#_p$ . The total number of trials is the number of nonmissing values of *varname* (in `bittest`) or  $\#_N$  (in `bittesti`). The number of observed successes is the number of 1s in *varname* (in `bittest`) or  $\#_{\text{succ}}$  (in `bittesti`). *varname* must contain only 0s, 1s, and missing.

`bittesti` is the immediate form of `bittest`; see [U] [19 Immediate commands](#) for a general introduction to immediate commands.

## Option

### Advanced

*detail* shows the probability of the observed number of successes,  $k_{\text{obs}}$ ; the probability of the number of successes on the opposite tail of the distribution that is used to compute the two-sided *p*-value,  $k_{\text{opp}}$ ; and the probability of the point next to  $k_{\text{opp}}$ . This information can be safely ignored. See the technical note below for details.

## Remarks

Remarks are presented under the following headings:

[bittest](#)  
[bittesti](#)

## bitest

### ► Example 1

We test 15 university students for high levels of one measure of visual quickness which, from other evidence, we believe is present in 30% of the nonuniversity population. Included in our data is quick, taking on the values 1 (“success”) or 0 (“failure”) depending on the outcome of the test.

```
. use http://www.stata-press.com/data/r11/quick
. bitest quick == 0.3
```

| Variable             | N                           | Observed k | Expected k | Assumed p | Observed p |
|----------------------|-----------------------------|------------|------------|-----------|------------|
| quick                | 15                          | 7          | 4.5        | 0.30000   | 0.46667    |
| Pr(k >= 7)           | = 0.131143 (one-sided test) |            |            |           |            |
| Pr(k <= 7)           | = 0.949987 (one-sided test) |            |            |           |            |
| Pr(k <= 1 or k >= 7) | = 0.166410 (two-sided test) |            |            |           |            |

The first part of the output reveals that, assuming a true probability of success of 0.3, the expected number of successes is 4.5 and that we observed seven. Said differently, the assumed frequency under the null hypothesis  $H_0$  is 0.3, and the observed frequency is 0.47.

The first line under the table is a one-sided test; it is the probability of observing seven or more successes conditional on  $p = 0.3$ . It is a test of  $H_0: p = 0.3$  versus the alternative hypothesis  $H_A: p > 0.3$ . Said in English, the alternative hypothesis is that more than 30% of university students score at high levels on this test of visual quickness. The  $p$ -value for this hypothesis test is 0.13.

The second line under the table is a one-sided test of  $H_0$  versus the opposite alternative hypothesis  $H_A: p < 0.3$ .

The third line is the two-sided test. It is a test of  $H_0$  versus the alternative hypothesis  $H_A: p \neq 0.3$ .

◀

### □ Technical note

The  $p$ -value of a hypothesis test is the probability (calculated assuming  $H_0$  is true) of observing any outcome as extreme or more extreme than the observed outcome, with extreme meaning in the direction of the alternative hypothesis. In example 1, the outcomes  $k = 8, 9, \dots, 15$  are clearly “more extreme” than the observed outcome  $k_{\text{obs}} = 7$  when considering the alternative hypothesis  $H_A: p \neq 0.3$ . However, outcomes with only a few successes are also in the direction of this alternative hypothesis. For two-sided hypotheses, outcomes with  $k$  successes are considered “as extreme or more extreme” than the observed outcome  $k_{\text{obs}}$  if  $\Pr(k) \leq \Pr(k_{\text{obs}})$ . Here  $\Pr(k = 0)$  and  $\Pr(k = 1)$  are both less than  $\Pr(k = 7)$ , so they are included in the two-sided  $p$ -value.

The `detail` option allows you to see the probability (assuming that  $H_0$  is true) of the observed successes ( $k = 7$ ) and the probability of the boundary point ( $k = 1$ ) of the opposite tail used for the two-sided  $p$ -value.

```
. bitest quick == 0.3, detail
```

| Variable             | N                             | Observed k | Expected k | Assumed p | Observed p |
|----------------------|-------------------------------|------------|------------|-----------|------------|
| quick                | 15                            | 7          | 4.5        | 0.30000   | 0.46667    |
| Pr(k >= 7)           | = 0.131143 (one-sided test)   |            |            |           |            |
| Pr(k <= 7)           | = 0.949987 (one-sided test)   |            |            |           |            |
| Pr(k <= 1 or k >= 7) | = 0.166410 (two-sided test)   |            |            |           |            |
| Pr(k == 7)           | = 0.081130 (observed)         |            |            |           |            |
| Pr(k == 2)           | = 0.091560                    |            |            |           |            |
| Pr(k == 1)           | = 0.030520 (opposite extreme) |            |            |           |            |

Also shown is the probability of the point next to the boundary point. This probability, namely,  $\Pr(k = 2) = 0.092$ , is certainly close to the probability of the observed outcome  $\Pr(k = 7) = 0.081$ , so some people might argue that  $k = 2$  should be included in the two-sided  $p$ -value. Statisticians (at least some we know) would reply that the  $p$ -value is a precisely defined concept and that this is an arbitrary “fuzzification” of its definition. When you compute exact  $p$ -values according to the precise definition of a  $p$ -value, your type I error is never more than what you say it is—so no one can criticize you for being anticonservative. Including the point  $k = 2$  is being overly conservative because it makes the  $p$ -value larger yet. But it is your choice; being overly conservative, at least in statistics, is always safe. Know that `bitest` and `bitesti` always keep to the precise definition of a  $p$ -value, so if you wish to include this extra point, you must do so by hand or by using the `r()` saved results; see [Saved results](#) below.



bitesti

Example 2

The binomial test is a function of two statistics and one parameter:  $N$ , the number of observations;  $k_{\text{obs}}$ , the number of observed successes; and  $p$ , the assumed probability of a success on a trial. For instance, in a city of  $N = 2,500,000$ , we observe  $k_{\text{obs}} = 36$  cases of a particular disease when the population rate for the disease is  $p = 0.00001$ .

```
. bitesti 2500000 36 .00001
```

| N                      | Observed k                  | Expected k | Assumed p | Observed p |
|------------------------|-----------------------------|------------|-----------|------------|
| 2500000                | 36                          | 25         | 0.00001   | 0.00001    |
| Pr(k >= 36)            | = 0.022458 (one-sided test) |            |           |            |
| Pr(k <= 36)            | = 0.985448 (one-sided test) |            |           |            |
| Pr(k <= 14 or k >= 36) | = 0.034859 (two-sided test) |            |           |            |



Example 3

Boice Jr. and Monson (1977) present data on breast cancer cases and person-years of observations for women with tuberculosis who were repeatedly exposed to multiple x-ray fluoroscopies and for women with tuberculosis who were not. The data are

|               | Exposed | Not exposed | Total  |
|---------------|---------|-------------|--------|
| Breast cancer | 41      | 15          | 56     |
| Person-years  | 28,010  | 19,017      | 47,027 |



We can thus test whether x-ray fluoroscopic examinations are associated with breast cancer; the assumed rate of exposure is  $p = 28010/47027$ .

```
. bitesti 56 41 28010/47027
```

|                        | N  | Observed k | Expected k | Assumed p        | Observed p |
|------------------------|----|------------|------------|------------------|------------|
|                        | 56 | 41         | 33.35446   | 0.59562          | 0.73214    |
| Pr(k >= 41)            |    |            | = 0.023830 | (one-sided test) |            |
| Pr(k <= 41)            |    |            | = 0.988373 | (one-sided test) |            |
| Pr(k <= 25 or k >= 41) |    |            | = 0.040852 | (two-sided test) |            |

◀

## Saved results

`bitest` and `bitesti` save the following in `r()`:

Scalars

|                     |                                    |                         |                                                           |
|---------------------|------------------------------------|-------------------------|-----------------------------------------------------------|
| <code>r(N)</code>   | number $N$ of trials               | <code>r(k_opp)</code>   | opposite extreme $k$                                      |
| <code>r(P_p)</code> | assumed probability $p$ of success | <code>r(P_k)</code>     | probability of observed $k$ (detail only)                 |
| <code>r(k)</code>   | observed number $k$ of successes   | <code>r(P_oppk)</code>  | probability of opposite extreme $k$ (detail only)         |
| <code>r(p_l)</code> | lower one-sided $p$ -value         | <code>r(k_nopp)</code>  | $k$ next to opposite extreme (detail only)                |
| <code>r(p_u)</code> | upper one-sided $p$ -value         | <code>r(P_noppk)</code> | probability of $k$ next to opposite extreme (detail only) |
| <code>r(p)</code>   | two-sided $p$ -value               |                         |                                                           |

## Methods and formulas

`bitest` and `bitesti` are implemented as ado-files.

Let  $N$ ,  $k_{\text{obs}}$ , and  $p$  be, respectively, the number of observations, the observed number of successes, and the assumed probability of success on a trial. The expected number of successes is  $Np$ , and the observed probability of success on a trial is  $k_{\text{obs}}/N$ .

`bitest` and `bitesti` compute exact  $p$ -values based on the binomial distribution. The upper one-sided  $p$ -value is

$$\Pr(k \geq k_{\text{obs}}) = \sum_{m=k_{\text{obs}}}^N \binom{N}{m} p^m (1-p)^{N-m}$$

The lower one-sided  $p$ -value is

$$\Pr(k \leq k_{\text{obs}}) = \sum_{m=0}^{k_{\text{obs}}} \binom{N}{m} p^m (1-p)^{N-m}$$

If  $k_{\text{obs}} \geq Np$ , the two-sided  $p$ -value is

$$\Pr(k \leq k_{\text{opp}} \text{ or } k \geq k_{\text{obs}})$$

where  $k_{\text{opp}}$  is the largest number  $\leq Np$  such that  $\Pr(k = k_{\text{opp}}) \leq \Pr(k = k_{\text{obs}})$ . If  $k_{\text{obs}} < Np$ , the two-sided  $p$ -value is

$$\Pr(k \leq k_{\text{obs}} \text{ or } k \geq k_{\text{opp}})$$

where  $k_{\text{opp}}$  is the smallest number  $\geq Np$  such that  $\Pr(k = k_{\text{opp}}) \leq \Pr(k = k_{\text{obs}})$ .

## References

- Boice Jr., J. D., and R. R. Monson. 1977. Breast cancer in women after repeated fluoroscopic examinations of the chest. *Journal of the National Cancer Institute* 59: 823–832.
- Hoel, P. G. 1984. *Introduction to Mathematical Statistics*. 5th ed. New York: Wiley.

## Also see

- [R] **ci** — Confidence intervals for means, proportions, and counts
- [R] **prtest** — One- and two-sample tests of proportions

## Syntax

```
bootstrap exp_list [, options eform_option] : command
```

| options                                         | description                                                                                                                    |
|-------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| Main                                            |                                                                                                                                |
| <code>reps(#)</code>                            | perform # bootstrap replications; default is <code>reps(50)</code>                                                             |
| Options                                         |                                                                                                                                |
| <code>strata(varlist)</code>                    | variables identifying strata                                                                                                   |
| <code>size(#)</code>                            | draw samples of size #; default is <code>_N</code>                                                                             |
| <code>cluster(varlist)</code>                   | variables identifying resampling clusters                                                                                      |
| <code>idcluster(newvar)</code>                  | create new cluster ID variable                                                                                                 |
| <code>saving(filename, ...)</code>              | save results to <i>filename</i> ; save statistics in double precision;<br>save results to <i>filename</i> every # replications |
| <code>bca</code>                                | compute acceleration for $BC_a$ confidence intervals                                                                           |
| <code>mse</code>                                | use MSE formula for variance estimation                                                                                        |
| Reporting                                       |                                                                                                                                |
| <code>level(#)</code>                           | set confidence level; default is <code>level(95)</code>                                                                        |
| <code>notable</code>                            | suppress table of results                                                                                                      |
| <code>noheader</code>                           | suppress table header                                                                                                          |
| <code>nolegend</code>                           | suppress table legend                                                                                                          |
| <code>verbose</code>                            | display the full table legend                                                                                                  |
| <code>nodots</code>                             | suppress the replication dots                                                                                                  |
| <code>noisily</code>                            | display any output from <i>command</i>                                                                                         |
| <code>trace</code>                              | trace the <i>command</i>                                                                                                       |
| <code>title(text)</code>                        | use <i>text</i> as title for bootstrap results                                                                                 |
| <code>display_options</code>                    | control spacing and display of omitted variables and base and empty cells                                                      |
| Advanced                                        |                                                                                                                                |
| <code>nodrop</code>                             | do not drop observations                                                                                                       |
| <code>nowarn</code>                             | do not warn when <code>e(sample)</code> is not set                                                                             |
| <code>force</code>                              | do not check for <i>weights</i> or <code>svy</code> commands; seldom used                                                      |
| <code>reject(exp)</code>                        | identify invalid results                                                                                                       |
| <code>seed(#)</code>                            | set random-number seed to #                                                                                                    |
| <sup>†</sup> <code>group(varname)</code>        | ID variable for groups within <code>cluster()</code>                                                                           |
| <sup>†</sup> <code>jackknifeopts(jkopts)</code> | options for jackknife                                                                                                          |
| <sup>†</sup> <code>coeflegend</code>            | display coefficients' legend instead of coefficient table                                                                      |

<sup>†</sup>`group()`, `jackknifeopts()`, and `coeflegend` do not appear in the dialog box.

*weights* are not allowed in *command*.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

|                          |                                                                                |
|--------------------------|--------------------------------------------------------------------------------|
| <i>exp_list</i> contains | ( <i>name</i> : <i>elist</i> )<br><i>elist</i><br><i>eexp</i>                  |
| <i>elist</i> contains    | <i>newvar</i> = ( <i>exp</i> )<br>( <i>exp</i> )                               |
| <i>eexp</i> is           | <i>specname</i><br>[ <i>eqno</i> ] <i>specname</i>                             |
| <i>specname</i> is       | <code>_b</code><br><code>_b[]</code><br><code>_se</code><br><code>_se[]</code> |
| <i>eqno</i> is           | <code>##</code><br><i>name</i>                                                 |

*exp* is a standard Stata expression; see [U] 13 **Functions and expressions**.

Distinguish between `[]`, which are to be typed, and `[]`, which indicate optional arguments.

## Menu

Statistics > Resampling > Bootstrap estimation

## Description

**bootstrap** performs bootstrap estimation. Typing

```
. bootstrap exp_list, reps(#): command
```

executes *command* multiple times, bootstrapping the statistics in *exp\_list* by resampling observations (with replacement) from the data in memory *#* times. This method is commonly referred to as the nonparametric bootstrap.

*command* defines the statistical command to be executed. Most Stata commands and user-written programs can be used with **bootstrap**, as long as they follow standard Stata syntax; see [U] 11 **Language syntax**. If the `bca` option is supplied, *command* must also work with **jackknife**; see [R] **jackknife**. The `by` prefix may not be part of *command*.

*exp\_list* specifies the statistics to be collected from the execution of *command*. If *command* changes the contents in `e(b)`, *exp\_list* is optional and defaults to `_b`.

Because bootstrapping is a random process, if you want to be able to reproduce results, set the random-number seed by specifying the `seed(#)` option or by typing

```
. set seed #
```

where *#* is a seed of your choosing, before running **bootstrap**; see [R] **set seed**.

Many estimation commands allow the `vce(bootstrap)` option. For those commands, we recommend using `vce(bootstrap)` over `bootstrap` because the estimation command already handles clustering and other model-specific details for you. The `bootstrap` prefix command is intended for use with nonestimation commands, such as `summarize`, user-written commands, or functions of coefficients.

`bs` and `bstrap` are synonyms for `bootstrap`.

## Options

### Main

`reps(#)` specifies the number of bootstrap replications to be performed. The default is 50. A total of 50–200 replications are generally adequate for estimates of standard error and thus are adequate for normal-approximation confidence intervals; see [Mooney and Duval \(1993, 11\)](#). Estimates of confidence intervals using the percentile or bias-corrected methods typically require 1,000 or more replications.

### Options

`strata(varlist)` specifies the variables that identify strata. If this option is specified, bootstrap samples are taken independently within each stratum.

`size(#)` specifies the size of the samples to be drawn. The default is `_N`, meaning to draw samples of the same size as the data. If specified, `#` must be less than or equal to the number of observations within `strata()`.

If `cluster()` is specified, the default size is the number of clusters in the original dataset. For unbalanced clusters, resulting sample sizes will differ from replication to replication. For cluster sampling, `#` must be less than or equal to the number of clusters within `strata()`.

`cluster(varlist)` specifies the variables that identify resampling clusters. If this option is specified, the sample drawn during each replication is a bootstrap sample of clusters.

`idcluster(newvar)` creates a new variable containing a unique identifier for each resampled cluster. This option requires that `cluster()` also be specified.

`saving(filename[, suboptions])` creates a Stata data file (`.dta` file) consisting of, for each statistic in `exp_list`, a variable containing the bootstrap replicates.

`double` specifies that the results for each replication be stored as doubles, meaning 8-byte reals. By default, they are stored as floats, meaning 4-byte reals. This option may be used without the `saving()` option to compute the variance estimates by using double precision.

`every(#)` specifies that results be written to disk every `#`th replication. `every()` should be specified only in conjunction with `saving()` when `command` takes a long time for each replication. This option will allow recovery of partial results should some other software crash your computer. See [\[P\] postfile](#).

`replace` specifies that `filename` be overwritten, if it exists. This option does not appear in the dialog box.

`bca` specifies that `bootstrap` estimate the acceleration of each statistic in `exp_list`. This estimate is used to construct  $BC_a$  confidence intervals. Type `estat bootstrap, bca` to display the  $BC_a$  confidence interval generated by the `bootstrap` command.

`mse` specifies that `bootstrap` compute the variance by using deviations of the replicates from the observed value of the statistics based on the entire dataset. By default, `bootstrap` computes the variance by using deviations from the average of the replicates.

## Reporting

`level(#)`; see [R] [estimation options](#).

`notable` suppresses the display of the table of results.

`noheader` suppresses the display of the table header. This option implies `nolegend`. This option may also be specified when replaying estimation results.

`nolegend` suppresses the display of the table legend. This option may also be specified when replaying estimation results.

`verbose` specifies that the full table legend be displayed. By default, coefficients and standard errors are not displayed. This option may also be specified when replaying estimation results.

`nodots` suppresses display of the replication dots. By default, one dot character is displayed for each successful replication. A red 'x' is displayed if *command* returns an error or if one of the values in *exp\_list* is missing.

`noisily` specifies that any output from *command* be displayed. This option implies the `nodots` option.

`trace` causes a trace of the execution of *command* to be displayed. This option implies the `noisily` option.

`title(text)` specifies a title to be displayed above the table of bootstrap results. The default title is the title saved in `e(title)` by an estimation command, or if `e(title)` is not filled in, *Bootstrap results* is used. `title()` may also be specified when replaying estimation results.

*display\_options*: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

## Advanced

`nodrop` prevents observations outside `e(sample)` and the `if` and `in` conditions from being dropped before the data are resampled.

`nowarn` suppresses the display of a warning message when *command* does not set `e(sample)`.

`force` suppresses the restriction that *command* not specify weights or be a `svy` command. This is a rarely used option. Use it only if you know what you are doing.

`reject(exp)` identifies an expression that indicates when results should be rejected. When *exp* is true, the resulting values are reset to missing values.

`seed(#)` sets the random-number seed. Specifying this option is equivalent to typing the following command prior to calling `bootstrap`:

```
. set seed #
```

The following options are available with `bootstrap` but are not shown in the dialog box:

*eform\_option* causes the coefficient table to be displayed in exponentiated form: for each coefficient, `exp(b)` rather than *b* is displayed. Standard errors and confidence intervals are also transformed. Display of the intercept, if any, is suppressed.

*command* determines which of the following are allowed (`eform(string)` and `eform` are always allowed):

| <i>eform_option</i>        | description                                                     |
|----------------------------|-----------------------------------------------------------------|
| <code>eform(string)</code> | use <i>string</i> for the column title                          |
| <code>eform</code>         | exponentiated coefficient, <i>string</i> is <code>exp(b)</code> |
| <code>hr</code>            | hazard ratio, <i>string</i> is Haz. Ratio                       |
| <code>shr</code>           | subhazard ratio, <i>string</i> is SHR                           |
| <code>irr</code>           | incidence-rate ratio, <i>string</i> is IRR                      |
| <code>or</code>            | odds ratio, <i>string</i> is Odds Ratio                         |
| <code>rrr</code>           | relative-risk ratio, <i>string</i> is RRR                       |

`group(varname)` re-creates *varname* containing a unique identifier for each group across the resampled clusters. This option requires that `idcluster()` also be specified.

This option is useful for maintaining unique group identifiers when sampling clusters with replacement. Suppose that cluster 1 contains 3 groups. If the `idcluster(newclid)` option is specified and cluster 1 is sampled multiple times, *newclid* uniquely identifies each copy of cluster 1. If `group(newgroupid)` is also specified, *newgroupid* uniquely identifies each copy of each group.

`jackknifeopts(jkopts)` identifies options that are to be passed to `jackknife` when it computes the acceleration values for the  $BC_a$  confidence intervals; see [R] [jackknife](#). This option requires the `bca` option and is mostly used for passing the `eclass`, `rclass`, or `n(#)` option to `jackknife`.

`coeflegend`; see [R] [estimation options](#).

## Remarks

Remarks are presented under the following headings:

- [Introduction](#)
- [Regression coefficients](#)
- [Expressions](#)
- [Combining bootstrap datasets](#)
- [A note about macros](#)
- [Achieved significance level](#)
- [Bootstrapping a ratio](#)
- [Warning messages and `e\(sample\)`](#)
- [Bootstrapping statistics from data with a complex structure](#)

## Introduction

With few assumptions, bootstrapping provides a way of estimating standard errors and other measures of statistical precision (Efron 1979; Efron and Stein 1981; Efron 1982; Efron and Tibshirani 1986; Efron and Tibshirani 1993; also see Davison and Hinkley [1997]; Guan [2003]; Mooney and Duval [1993]; Poi [2004]; and Stine [1990]). It provides a way to obtain such measures when no formula is otherwise available or when available formulas make inappropriate assumptions. Cameron and Trivedi (2009, chap. 13) discuss many bootstrapping topics and demonstrate how to do them in Stata.

To illustrate bootstrapping, suppose that you have a dataset containing  $N$  observations and an estimator that, when applied to the data, produces certain statistics. You draw, with replacement,  $N$  observations from the  $N$ -observation dataset. In this random drawing, some of the original observations will appear once, some more than once, and some not at all. Using the resampled dataset, you apply the estimator and collect the statistics. This process is repeated many times; each time, a new random sample is drawn and the statistics are recalculated.

This process builds a dataset of replicated statistics. From these data, you can calculate the standard error by using the standard formula for the sample standard deviation

$$\widehat{\text{se}} = \left\{ \frac{1}{k-1} \sum (\hat{\theta}_i - \bar{\theta})^2 \right\}^{1/2}$$

where  $\hat{\theta}_i$  is the statistic calculated using the  $i$ th bootstrap sample and  $k$  is the number of replications. This formula gives an estimate of the standard error of the statistic, according to [Hall and Wilson \(1991\)](#). Although the average,  $\bar{\theta}$ , of the bootstrapped estimates is used in calculating the standard deviation, it is not used as the estimated value of the statistic itself. Instead, the original observed value of the statistic,  $\hat{\theta}$ , is used, meaning the value of the statistic computed using the original  $N$  observations.

You might think that  $\bar{\theta}$  is a better estimate of the parameter than  $\hat{\theta}$ , but it is not. If the statistic is biased, bootstrapping exaggerates the bias. In fact, the bias can be estimated as  $\bar{\theta} - \hat{\theta}$  ([Efron 1982](#), 33). Knowing this, you might be tempted to subtract this estimate of bias from  $\hat{\theta}$  to produce an unbiased statistic. The bootstrap bias estimate has an indeterminate amount of random error, so this unbiased estimator may have greater mean squared error than the biased estimator ([Mooney and Duval 1993](#); [Hinkley 1978](#)). Thus  $\hat{\theta}$  is the best point estimate of the statistic.

The logic behind the bootstrap is that all measures of precision come from a statistic's sampling distribution. When the statistic is estimated on a sample of size  $N$  from some population, the sampling distribution tells you the relative frequencies of the values of the statistic. The sampling distribution, in turn, is determined by the distribution of the population and the formula used to estimate the statistic.

Sometimes the sampling distribution can be derived analytically. For instance, if the underlying population is distributed normally and you calculate means, the sampling distribution for the mean is also normal but has a smaller variance than that of the population. In other cases, deriving the sampling distribution is difficult, as when means are calculated from nonnormal populations. Sometimes, as in the case of means, it is not too difficult to derive the sampling distribution as the sample size goes to infinity ( $N \rightarrow \infty$ ). However, such asymptotic distributions may not perform well when applied to finite samples.

If you knew the population distribution, you could obtain the sampling distribution by simulation: you could draw random samples of size  $N$ , calculate the statistic, and make a tally. Bootstrapping does precisely this, but it uses the observed distribution of the sample in place of the true population distribution. Thus the bootstrap procedure hinges on the assumption that the observed distribution is a good estimate of the underlying population distribution. In return, the bootstrap produces an estimate, called the bootstrap distribution, of the sampling distribution. From this, you can estimate the standard error of the statistic, produce confidence intervals, etc.

The accuracy with which the bootstrap distribution estimates the sampling distribution depends on the number of observations in the original sample and the number of replications in the bootstrap. A crudely estimated sampling distribution is adequate if you are only going to extract, say, a standard error. A better estimate is needed if you want to use the 2.5th and 97.5th percentiles of the distribution to produce a 95% confidence interval. To extract many features simultaneously about the distribution, an even better estimate is needed. Generally, replications on the order of 1,000 produce very good estimates, but only 50–200 replications are needed for estimates of standard errors. See [Poi \(2004\)](#) for a method to choose the number of bootstrap replications.



## Regression coefficients

### ► Example 1

Let's say that we wish to compute bootstrap estimates for the standard errors of the coefficients from the following regression:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
```

```
. regress mpg weight gear foreign
```

| Source   | SS         | df | MS         | Number of obs = 74     |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | 1629.67805 | 3  | 543.226016 | F( 3, 70) = 46.73      |  |  |
| Residual | 813.781411 | 70 | 11.6254487 | Prob > F = 0.0000      |  |  |
|          |            |    |            | R-squared = 0.6670     |  |  |
|          |            |    |            | Adj R-squared = 0.6527 |  |  |
| Total    | 2443.45946 | 73 | 33.4720474 | Root MSE = 3.4096      |  |  |

| mpg        | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|------------|-----------|-----------|-------|-------|----------------------|-----------|
| weight     | -.006139  | .0007949  | -7.72 | 0.000 | -.0077245            | -.0045536 |
| gear_ratio | 1.457113  | 1.541286  | 0.95  | 0.348 | -1.616884            | 4.53111   |
| foreign    | -2.221682 | 1.234961  | -1.80 | 0.076 | -4.684735            | .2413715  |
| _cons      | 36.10135  | 6.285984  | 5.74  | 0.000 | 23.56435             | 48.63835  |

To run the bootstrap, we simply prefix the above regression command with the `bootstrap` command (specifying its options before the colon separator). We must set the random-number seed before calling `bootstrap`.

```
. bootstrap, reps(100) seed(1): regress mpg weight gear foreign
(running regress on estimation sample)
```

```
Bootstrap replications (100)
```

```
-----| 1 -----| 2 -----| 3 -----| 4 -----| 5
.....
..... 50
..... 100
```

|                   |               |   |        |
|-------------------|---------------|---|--------|
| Linear regression | Number of obs | = | 74     |
|                   | Replications  | = | 100    |
|                   | Wald chi2(3)  | = | 111.96 |
|                   | Prob > chi2   | = | 0.0000 |
|                   | R-squared     | = | 0.6670 |
|                   | Adj R-squared | = | 0.6527 |
|                   | Root MSE      | = | 3.4096 |

| mpg        | Observed<br>Coef. | Bootstrap<br>Std. Err. | z     | P> z  | Normal-based<br>[95% Conf. Interval] |           |
|------------|-------------------|------------------------|-------|-------|--------------------------------------|-----------|
| weight     | -.006139          | .0006498               | -9.45 | 0.000 | -.0074127                            | -.0048654 |
| gear_ratio | 1.457113          | 1.297786               | 1.12  | 0.262 | -1.086501                            | 4.000727  |
| foreign    | -2.221682         | 1.162728               | -1.91 | 0.056 | -4.500587                            | .0572236  |
| _cons      | 36.10135          | 4.71779                | 7.65  | 0.000 | 26.85465                             | 45.34805  |

The displayed confidence interval is based on the assumption that the sampling (and hence bootstrap) distribution is approximately normal (see [Methods and formulas](#) below). Because this confidence interval is based on the standard error, it is a reasonable estimate if normality is approximately true, even for a few replications. Other types of confidence intervals are available after `bootstrap`; see [\[R\] bootstrap postestimation](#).

We could instead supply names to our expressions when we run `bootstrap`. For example,

```
. bootstrap diff=(_b[weight]-_b[gear]): regress mpg weight gear foreign
```

would bootstrap a statistic, named `diff`, equal to the difference between the coefficients on `weight` and `gear_ratio`.



## Expressions

### ► Example 2

When we use `bootstrap`, the list of statistics can contain complex expressions, as long as each expression is enclosed in parentheses. For example, to bootstrap the range of a variable `x`, we could type

```
. bootstrap range=(r(max)-r(min)), reps(1000): summarize x
```

Of course, we could also bootstrap the minimum and maximum and later compute the range.

```
. bootstrap max=r(max) min=r(min), reps(1000) saving(mybs): summarize x
. use mybs, clear
. (bootstrap: summarize)
. generate range = max - min
. bstat range, stat(19.5637501)
```

The difference between the maximum and minimum of `x` in the sample is 19.5637501.

The `stat()` option to `bstat` specifies the observed value of the statistic (`range`) to be summarized. This option is useful when, as shown above, the statistic of ultimate interest is not specified directly to `bootstrap` but instead is calculated by other means.

Here the observed values of `r(max)` and `r(min)` are saved as characteristics of the dataset created by `bootstrap` and are thus available for retrieval by `bstat`; see [R] [bstat](#). The observed range, however, is unknown to `bstat`, so it must be specified.



## Combining bootstrap datasets

You can combine two datasets from separate runs of `bootstrap` by using `append` (see [D] [append](#)) and then get the bootstrap statistics for the combined datasets by running `bstat`. The runs must have been performed independently (having different starting random-number seeds), and the original dataset, command, and bootstrap statistics must have been all the same.

## A note about macros

In the previous example, we executed the command

```
. bootstrap max=r(max) min=r(min), reps(1000) saving(mybs): summarize x
```

We did not enclose  $r(\max)$  and  $r(\min)$  in single quotes, as we would in most other contexts, because it would not produce what was intended:

```
. bootstrap 'r(max)' 'r(min)', reps(1000) saving(mybs): summarize x
```

To understand why, note that `'r(max)'`, like any reference to a local macro, will evaluate to a literal string containing the contents of  $r(\max)$  *before* `bootstrap` is even executed. Typing the command above would appear to Stata as if we had typed

```
. bootstrap 14.5441234 33.4393293, reps(1000) saving(mybs): summarize x
```

Even worse, the current contents of  $r(\min)$  and  $r(\max)$  could be empty, producing an even more confusing result. To avoid this outcome, refer to statistics by name (e.g.,  $r(\max)$ ) and not by value (e.g., `'r(max)'`).

## Achieved significance level

### ► Example 3

Suppose that we wish to estimate the *achieved significance level* (ASL) of a test statistic by using the bootstrap. ASL is another name for  $p$ -value. An example is

$$\text{ASL} = \Pr(\hat{\theta}^* \geq \hat{\theta} | H_0)$$

for an upper-tailed, alternative hypothesis, where  $H_0$  denotes the null hypothesis,  $\hat{\theta}$  is the observed value of the test statistic, and  $\hat{\theta}^*$  is the random variable corresponding to the test statistic, assuming that  $H_0$  is true.

Here we will compare the mean miles per gallon (mpg) between foreign and domestic cars by using the two-sample  $t$  test with unequal variances. The following results indicate the  $p$ -value to be 0.0034 for the two-sided test using Satterthwaite's approximation. Thus assuming that mean mpg is the same for foreign and domestic cars, we would expect to observe a  $t$  statistic more extreme (in absolute value) than 3.1797 in about 0.3% of all possible samples of the type that we observed. Thus we have evidence to reject the null hypothesis that the means are equal.

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
```

```
. ttest mpg, by(foreign) unequal
```

Two-sample t test with unequal variances

| Group    | Obs | Mean      | Std. Err. | Std. Dev. | [95% Conf. Interval] |           |
|----------|-----|-----------|-----------|-----------|----------------------|-----------|
| Domestic | 52  | 19.82692  | .657777   | 4.743297  | 18.50638             | 21.14747  |
| Foreign  | 22  | 24.77273  | 1.40951   | 6.611187  | 21.84149             | 27.70396  |
| combined | 74  | 21.2973   | .6725511  | 5.785503  | 19.9569              | 22.63769  |
| diff     |     | -4.945804 | 1.555438  |           | -8.120053            | -1.771556 |

```
diff = mean(Domestic) - mean(Foreign) t = -3.1797
Ho: diff = 0 Satterthwaite's degrees of freedom = 30.5463
Ha: diff < 0 Ha: diff != 0 Ha: diff > 0
Pr(T < t) = 0.0017 Pr(|T| > |t|) = 0.0034 Pr(T > t) = 0.9983
```

We also place the value of the test statistic in a scalar for later use.

```
. scalar tobs = r(t)
```

Efron and Tibshirani (1993, 224) describe an alternative to Satterthwaite’s approximation that estimates the ASL by bootstrapping the statistic from the test of equal means. Their idea is to recenter the two samples to the combined sample mean so that the data now conform to the null hypothesis but that the variances within the samples remain unchanged.

```
. summarize mpg, meanonly
. scalar omean = r(mean)
. summarize mpg if foreign==0, meanonly
. replace mpg = mpg - r(mean) + scalar(omean) if foreign==0
mpg was int now float
(52 real changes made)
. summarize mpg if foreign==1, meanonly
. replace mpg = mpg - r(mean) + scalar(omean) if foreign==1
(22 real changes made)
. sort foreign
. by foreign: summarize mpg
```

-> foreign = Domestic

| Variable | Obs | Mean    | Std. Dev. | Min      | Max      |
|----------|-----|---------|-----------|----------|----------|
| mpg      | 52  | 21.2973 | 4.743297  | 13.47037 | 35.47038 |

-> foreign = Foreign

| Variable | Obs | Mean    | Std. Dev. | Min      | Max      |
|----------|-----|---------|-----------|----------|----------|
| mpg      | 22  | 21.2973 | 6.611187  | 10.52457 | 37.52457 |

Each sample (foreign and domestic) is a stratum, so the bootstrapped samples must have the same number of foreign and domestic cars as the original dataset. This requirement is facilitated by the `strata()` option to `bootstrap`. By typing the following, we bootstrap the test statistic using the modified dataset and save the values in `bsauto2.dta`:

```
. keep mpg foreign
. set seed 1
. bootstrap t=r(t), rep(1000) strata(foreign) saving(bsauto2) nodots: ttest mpg,
> by(foreign) unequal
```

Warning: Because **ttest** is not an estimation command or does not set **e(sample)**, **bootstrap** has no way to determine which observations are used in calculating the statistics and so assumes that all observations are used. This means that no observations will be excluded from the resampling because of missing values or other reasons.

If the assumption is not true, press Break, save the data, and drop the observations that are to be excluded. Be sure that the dataset in memory contains only the relevant data.

Bootstrap results

```
Number of strata = 2 Number of obs = 74
Replications = 1000
command: ttest mpg, by(foreign) unequal
t: r(t)
```

|   | Observed<br>Coef. | Bootstrap<br>Std. Err. | z    | P> z  | Normal-based<br>[95% Conf. Interval] |          |
|---|-------------------|------------------------|------|-------|--------------------------------------|----------|
| t | 1.75e-07          | 1.036437               | 0.00 | 1.000 | -2.031379                            | 2.031379 |

We can use the data in `bsauto2.dta` to estimate ASL via the fraction of bootstrap test statistics that are more extreme than 3.1797.

```
. use bsauto2, clear
(bootstrap: ttest)
. generate indicator = abs(t)>=abs(scalar(tobs))
. summarize indicator, meanonly
. display "ASLboot = " r(mean)
ASLboot = .005
```

The result is  $ASL_{boot} = 0.005$ . Assuming that the mean mpg is the same between foreign and domestic cars, we would expect to observe a  $t$  statistic more extreme (in absolute value) than 3.1797 in about 0.5% of all possible samples of the type we observed. This finding is still strong evidence to reject the hypothesis that the means are equal.

◀

## Bootstrapping a ratio

### ► Example 4

Suppose that we wish to produce a bootstrap estimate of the ratio of two means. Because `summarize` saves results for only one variable, we must call `summarize` twice to compute the means. Actually, we could use `collapse` to compute the means in one call, but calling `summarize` twice is much faster. Thus we will have to write a small program that will return the results we want.

We write the program below and save it to a file called `ratio.ado` (see [U] 17 [Ado-files](#)). Our program takes two variable names as input and saves them in the local macros `y` (first variable) and `x` (second variable). It then computes one statistic: the mean of ‘`y`’ divided by the mean of ‘`x`’. This value is returned as a scalar in `r(ratio)`. `ratio` also returns the ratio of the number of observations used to the mean for each variable.

```

program myratio, rclass
 version 11
 args y x
 confirm var `y'
 confirm var `x'
 tempname ymean yn
 summarize `y', meanonly
 scalar `ymean' = r(mean)
 return scalar n_`y' = r(N)
 summarize `x', meanonly
 return scalar n_`x' = r(N)
 return scalar ratio = `ymean'/r(mean)
end

```

Remember to test any newly written commands before using them with `bootstrap`.

```

. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)

```

```

. summarize price

```

| Variable | Obs | Mean     | Std. Dev. | Min  | Max   |
|----------|-----|----------|-----------|------|-------|
| price    | 74  | 6165.257 | 2949.496  | 3291 | 15906 |

```

. scalar mean1=r(mean)

```

```

. summarize weight

```

| Variable | Obs | Mean     | Std. Dev. | Min  | Max  |
|----------|-----|----------|-----------|------|------|
| weight   | 74  | 3019.459 | 777.1936  | 1760 | 4840 |

```

. scalar mean2=r(mean)

```

```

. di scalar(mean1)/scalar(mean2)
2.0418412

```

```

. myratio price weight

```

```

. return list

```

```

scalars:

```

```

 r(ratio) = 2.041841210168278
 r(n_weight) = 74
 r(n_price) = 74

```

The results of running `bootstrap` on our program are

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. set seed 1
. bootstrap ratio=r(ratio), reps(1000) nowarn nodots: myratio price weight

Bootstrap results Number of obs = 74
 Replications = 1000

command: myratio price weight
ratio: r(ratio)
```

|       | Observed<br>Coef. | Bootstrap<br>Std. Err. | z     | P> z  | Normal-based<br>[95% Conf. Interval] |          |
|-------|-------------------|------------------------|-------|-------|--------------------------------------|----------|
| ratio | 2.041841          | .0942932               | 21.65 | 0.000 | 1.85703                              | 2.226652 |

As mentioned previously, we should specify the `saving()` option if we wish to save the bootstrap dataset.

◀

## Warning messages and `e(sample)`

`bootstrap` is not meant to be used with weighted calculations. `bootstrap` determines the presence of weights by parsing the prefixed command with standard syntax. However, commands like `stcox` and `streg` require that weights be specified in `stset`, and some user commands may allow weights to be specified by using an option instead of the standard syntax. Both cases pose a problem for `bootstrap` because it cannot determine the presence of weights under these circumstances. In these cases, we can only assume that you know what you are doing.

`bootstrap` does not know which variables of the dataset in memory matter to the calculation at hand. You can speed their execution by dropping unnecessary variables because, otherwise, they are included in each bootstrap sample.

You should thus drop observations with missing values. Leaving in missing values causes no problem in one sense because all Stata commands deal with missing values gracefully. It does, however, cause a statistical problem. Bootstrap sampling is defined as drawing, with replacement, samples of size  $N$  from a set of  $N$  observations. `bootstrap` determines  $N$  by counting the number of observations in memory, not counting the number of nonmissing values on the relevant variables. The result is that too many observations are resampled; the resulting bootstrap samples, because they are drawn from a population with missing values, are of unequal sizes.

If the number of missing values relative to the sample size is small, this will make little difference. If you have many missing values, however, you should first drop the observations that contain them.

## ► Example 5

To illustrate, we use the previous example but replace some of the values of `price` with missing values. The number of values of `price` used to compute the mean for each bootstrap is not constant. This is the purpose of the `Warning` message.

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. replace price = . if inlist(_n,1,3,5,7)
(4 real changes made, 4 to missing)
. set seed 1
```

```
. bootstrap ratio=r(ratio) np=r(n_price) nw=r(n_weight), reps(100) nodots:
> myratio price weight

Warning: Because myratio is not an estimation command or does not set
e(sample), bootstrap has no way to determine which observations are
used in calculating the statistics and so assumes that all
observations are used. This means that no observations will be
excluded from the resampling because of missing values or other
reasons.

If the assumption is not true, press Break, save the data, and drop
the observations that are to be excluded. Be sure that the dataset
in memory contains only the relevant data.

Bootstrap results Number of obs = 74
 Replications = 100

command: myratio price weight
ratio: r(ratio)
np: r(n_price)
nw: r(n_weight)
```

|       | Observed<br>Coef. | Bootstrap<br>Std. Err. | z     | P> z  | Normal-based<br>[95% Conf. Interval] |          |
|-------|-------------------|------------------------|-------|-------|--------------------------------------|----------|
| ratio | 2.063051          | .0893669               | 23.09 | 0.000 | 1.887896                             | 2.238207 |
| np    | 70                | 1.872178               | 37.39 | 0.000 | 66.3306                              | 73.6694  |
| nw    | 74                | .                      | .     | .     | .                                    | .        |



## Bootstrapping statistics from data with a complex structure

Here we describe how to bootstrap statistics from data with a complex structure, e.g., longitudinal or panel data, or matched data. `bootstrap`, however, is not designed to work with complex survey data. It is important to include all necessary information about the structure of the data in the `bootstrap` syntax to obtain correct bootstrap estimates for standard errors and confidence intervals.

`bootstrap` offers several options identifying the specifics of the data. These options are `strata()`, `cluster()`, `idcluster()`, and `group()`. The usage of `strata()` was described in example 3 above. Below we demonstrate several examples that require specifying the other three options.

### ➤ Example 6

Suppose that the `auto` data in [example 1](#) above are clustered by `rep78`. We want to obtain bootstrap estimates for the standard errors of the difference between the coefficients on `weight` and `gear_ratio`, taking into account clustering.

We supply the `cluster(rep78)` option to `bootstrap` to request resampling from clusters rather than from observations in the dataset.



```

. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)
. keep if rep78 < .
(5 observations deleted)
. bootstrap diff=(_b[weight]-_b[gear]), seed(1) cluster(rep78): regress mpg weight
> gear foreign
(running regress on estimation sample)
Bootstrap replications (50)
|-----| 1 |-----| 2 |-----| 3 |-----| 4 |-----| 5
..... 50
Linear regression Number of obs = 69
 Replications = 50

 command: regress mpg weight gear foreign
 diff: _b[weight]-_b[gear]

 (Replications based on 5 clusters in rep78)

```

|      | Observed<br>Coef. | Bootstrap<br>Std. Err. | z     | P> z  | Normal-based<br>[95% Conf. Interval] |          |
|------|-------------------|------------------------|-------|-------|--------------------------------------|----------|
| diff | -1.910396         | 1.876778               | -1.02 | 0.309 | -5.588812                            | 1.768021 |

We drop missing values in `rep78` before issuing the command because `bootstrap` does not allow missing values in `cluster()`. See the section above about using `bootstrap` when variables contain missing values.

We can also obtain these same results by using the following syntax:

```

. bootstrap diff=(_b[weight]-_b[gear]), seed(1): regress mpg weight gear foreign,
> vce(cluster rep78)

```

When only clustered information is provided to the command, `bootstrap` can pick up the `vce(cluster clustvar)` option from the main command and use it to resample from clusters.



## ➤ Example 7

Suppose now that we have matched data and want to use `bootstrap` to obtain estimates of the standard errors of the exponentiated difference between two coefficients (or, equivalently, the ratio of two odds ratios) estimated by `clogit`. Consider the example of matched case–control data on birthweight of infants described in [example 2](#) of [\[R\] clogit](#).

The infants are paired by being matched on mother's age. All groups, defined by the `paired` variable, have 1:2 matching. `clogit` requires that the matching information, `paired`, be supplied to the `group()` (or, equivalently, `strata()`) option to be used in computing the parameter estimates. Because the data are matched, we need to resample from groups rather than from the whole dataset. However, simply supplying the grouping variable `paired` in `cluster()` is not enough with `bootstrap`, as it is with clustered data.

```
. use http://www.stata-press.com/data/r11/lowbirth2, clear
(Applied Logistic Regression, Hosmer & Lemeshow)
. bootstrap ratio=exp(_b[smoke]-_b[ptd]), seed(1) cluster(pairid): clogit low lwt
> smoke ptd ht ui i.race, group(pairid)
(running clogit on estimation sample)
Bootstrap replications (50)
_____ 1 _____ 2 _____ 3 _____ 4 _____ 5
..... 50
Bootstrap results
Number of obs = 112
Replications = 50

command: clogit low lwt smoke ptd ht ui i.race, group(pairid)
ratio: exp(_b[smoke]-_b[ptd])

(Replications based on 56 clusters in pairid)
```

|       | Observed<br>Coef. | Bootstrap<br>Std. Err. | z    | P> z  | Normal-based<br>[95% Conf. Interval] |          |
|-------|-------------------|------------------------|------|-------|--------------------------------------|----------|
| ratio | .6654095          | 17.71791               | 0.04 | 0.970 | -34.06106                            | 35.39187 |

For the syntax above, imagine that the first pair was sampled twice during a replication. Then the bootstrap sample has four subjects with pairid equal to one, which clearly violates the original 1:2 matching design. As a result, the estimates of the coefficients obtained from this bootstrap sample will be incorrect.

Therefore, in addition to resampling from groups, we need to ensure that resampled groups are uniquely identified in each of the bootstrap samples. The `idcluster(newcluster)` option is designed for this. It requests that at each replication bootstrap create the new variable, *newcluster*, containing unique identifiers for all resampled groups. Thus, to make sure that the correct matching is preserved during each replication, we need to specify the grouping variable in `cluster()`, supply a variable name to `idcluster()`, and use this variable as the grouping variable with `clogit`, as we demonstrate below.

```
. bootstrap ratio=exp(_b[smoke]-_b[ptd]), seed(1) cluster(pairid)
> idcluster(newpairid): clogit low lwt smoke ptd ht ui i.race, group(newpairid)
(running clogit on estimation sample)
Bootstrap replications (50)
_____ 1 _____ 2 _____ 3 _____ 4 _____ 5
..... 50
Bootstrap results
Number of obs = 112
Replications = 50

command: clogit low lwt smoke ptd ht ui i.race, group(newpairid)
ratio: exp(_b[smoke]-_b[ptd])

(Replications based on 56 clusters in pairid)
```

|       | Observed<br>Coef. | Bootstrap<br>Std. Err. | z    | P> z  | Normal-based<br>[95% Conf. Interval] |          |
|-------|-------------------|------------------------|------|-------|--------------------------------------|----------|
| ratio | .6654095          | 7.919441               | 0.08 | 0.933 | -14.85641                            | 16.18723 |

Note the difference between the estimates of the bootstrap standard error for the two specifications of the bootstrap syntax.

## □ Technical note

Similarly, when you have panel (longitudinal) data, all resampled panels must be unique in each of the bootstrap samples to obtain correct bootstrap estimates of statistics. Therefore, both `cluster(panelvar)` and `idcluster(newpanelvar)` must be specified with `bootstrap`, and `i(newpanelvar)` must be used with the main command. Moreover, you must clear the current `xtset` settings by typing `xtset, clear` before calling `bootstrap`.

□

## ▷ Example 8

Continuing with our birthweight data, suppose that we have more information about doctors supervising women's pregnancies. We believe that the data on the pairs of infants from the same doctor may be correlated and want to adjust standard errors for possible correlation among the pairs. `clogit` offers the `vce(cluster clustvar)` option to do this.

Let's add a cluster variable to our dataset. One thing to keep in mind is that to use `vce(cluster clustvar)`, groups in `group()` must be nested within clusters.

```
. use http://www.stata-press.com/data/r11/lowbirth2, clear
(Appplied Logistic Regression, Hosmer & Lemeshow)
. set seed 12345
. by pairid, sort: egen byte doctor = total(int(2*runiform()+1)*(_n == 1))
. clogit low lwt smoke ptd ht ui i.race, group(pairid) vce(cluster doctor)

Iteration 0: log pseudolikelihood = -26.768693
Iteration 1: log pseudolikelihood = -25.810476
Iteration 2: log pseudolikelihood = -25.794296
Iteration 3: log pseudolikelihood = -25.794271
Iteration 4: log pseudolikelihood = -25.794271

Conditional (fixed-effects) logistic regression Number of obs = 112
 Wald chi2(1) = .
 Prob > chi2 = .
Log pseudolikelihood = -25.794271 Pseudo R2 = 0.3355

 (Std. Err. adjusted for 2 clusters in doctor)
```

| low   | Coef.     | Robust<br>Std. Err. | z      | P> z  | [95% Conf. Interval] |          |
|-------|-----------|---------------------|--------|-------|----------------------|----------|
| lwt   | -.0183757 | .0217802            | -0.84  | 0.399 | -.0610641            | .0243128 |
| smoke | 1.400656  | .0085545            | 163.73 | 0.000 | 1.38389              | 1.417423 |
| ptd   | 1.808009  | .938173             | 1.93   | 0.054 | -.0307765            | 3.646794 |
| ht    | 2.361152  | 1.587013            | 1.49   | 0.137 | -.7493362            | 5.47164  |
| ui    | 1.401929  | .8568119            | 1.64   | 0.102 | -.2773913            | 3.08125  |
| race  |           |                     |        |       |                      |          |
| 2     | .5713643  | .0672593            | 8.49   | 0.000 | .4395385             | .7031902 |
| 3     | -.0253148 | .9149785            | -0.03  | 0.978 | -1.81864             | 1.76801  |

To obtain correct bootstrap standard errors of the exponentiated difference between the two coefficients in this example, we need to make sure that both resampled clusters and groups within resampled clusters are unique in each of the bootstrap samples. To achieve this, `bootstrap` needs the information about clusters in `cluster()`, the variable name of the new identifier for clusters in `idcluster()`, and the information about groups in `group()`. We demonstrate the corresponding syntax of `bootstrap` below.



## Saved results

`bootstrap` saves the following in `e()`:

### Scalars

|                            |                                                                           |
|----------------------------|---------------------------------------------------------------------------|
| <code>e(N)</code>          | sample size                                                               |
| <code>e(N_reps)</code>     | number of complete replications                                           |
| <code>e(N_misreps)</code>  | number of incomplete replications                                         |
| <code>e(N_strata)</code>   | number of strata                                                          |
| <code>e(N_clust)</code>    | number of clusters                                                        |
| <code>e(k_eq)</code>       | number of equations                                                       |
| <code>e(k_exp)</code>      | number of standard expressions                                            |
| <code>e(k_eeexp)</code>    | number of extended expressions (i.e., <code>_b</code> )                   |
| <code>e(k_extra)</code>    | number of extra equations beyond the original ones from <code>e(b)</code> |
| <code>e(level)</code>      | confidence level for bootstrap CIs                                        |
| <code>e(bs_version)</code> | version for <code>bootstrap</code> results                                |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                                                 |

### Macros

|                            |                                                           |
|----------------------------|-----------------------------------------------------------|
| <code>e(cmdname)</code>    | command name from <i>command</i>                          |
| <code>e(cmd)</code>        | same as <code>e(cmdname)</code> or <code>bootstrap</code> |
| <code>e(command)</code>    | <i>command</i>                                            |
| <code>e(cmdline)</code>    | command as typed                                          |
| <code>e(prefix)</code>     | <code>bootstrap</code>                                    |
| <code>e(title)</code>      | title in estimation output                                |
| <code>e(strata)</code>     | strata variables                                          |
| <code>e(cluster)</code>    | cluster variables                                         |
| <code>e(seed)</code>       | initial random-number seed                                |
| <code>e(size)</code>       | from the <code>size(#)</code> option                      |
| <code>e(exp#)</code>       | expression for the <i>#</i> th statistic                  |
| <code>e(mse)</code>        | mse, if specified                                         |
| <code>e(vce)</code>        | <code>bootstrap</code>                                    |
| <code>e(vcetype)</code>    | title used to label Std. Err.                             |
| <code>e(properties)</code> | <code>b V</code>                                          |

### Matrices

|                               |                                      |
|-------------------------------|--------------------------------------|
| <code>e(b)</code>             | observed statistics                  |
| <code>e(b_bs)</code>          | bootstrap estimates                  |
| <code>e(reps)</code>          | number of nonmissing results         |
| <code>e(bias)</code>          | estimated biases                     |
| <code>e(se)</code>            | estimated standard errors            |
| <code>e(z0)</code>            | median biases                        |
| <code>e(accel)</code>         | estimated accelerations              |
| <code>e(ci_normal)</code>     | normal-approximation CIs             |
| <code>e(ci_percentile)</code> | percentile CIs                       |
| <code>e(ci_bc)</code>         | bias-corrected CIs                   |
| <code>e(ci_bca)</code>        | bias-corrected and accelerated CIs   |
| <code>e(V)</code>             | bootstrap variance–covariance matrix |
| <code>e(V_modelbased)</code>  | model-based variance                 |

When *exp\_list* is `_b`, `bootstrap` will also carry forward most of the results already in `e()` from *command*.

## Methods and formulas

`bootstrap` is implemented as an ado-file.

Let  $\hat{\theta}$  be the observed value of the statistic, that is, the value of the statistic calculated with the original dataset. Let  $i = 1, 2, \dots, k$  denote the bootstrap samples, and let  $\hat{\theta}_i$  be the value of the statistic from the  $i$ th bootstrap sample.

When the `mse` option is specified, the standard error is estimated as

$$\widehat{\text{se}}_{\text{MSE}} = \left\{ \frac{1}{k} \sum_{i=1}^k (\hat{\theta}_i - \hat{\theta})^2 \right\}^{1/2}$$

Otherwise, the standard error is estimated as

$$\widehat{\text{se}} = \left\{ \frac{1}{k-1} \sum_{i=1}^k (\hat{\theta}_i - \bar{\theta})^2 \right\}^{1/2}$$

where

$$\bar{\theta} = \frac{1}{k} \sum_{i=1}^k \hat{\theta}_i$$

The variance–covariance matrix is similarly computed. The bias is estimated as

$$\widehat{\text{bias}} = \bar{\theta} - \hat{\theta}$$

Confidence intervals with nominal coverage rates  $1 - \alpha$  are calculated according to the following formulas. The normal-approximation method yields the confidence intervals

$$[\hat{\theta} - z_{1-\alpha/2} \widehat{\text{se}}, \hat{\theta} + z_{1-\alpha/2} \widehat{\text{se}}]$$

where  $z_{1-\alpha/2}$  is the  $(1 - \alpha/2)$ th quantile of the standard normal distribution. If the `mse` option is specified, `bootstrap` will report the normal confidence interval using  $\widehat{\text{se}}_{\text{MSE}}$  instead of  $\widehat{\text{se}}$ . `estat bootstrap` only uses  $\widehat{\text{se}}$  in the normal confidence interval.

The percentile method yields the confidence intervals

$$[\theta_{\alpha/2}^*, \theta_{1-\alpha/2}^*]$$

where  $\theta_p^*$  is the  $p$ th quantile (the 100pth percentile) of the bootstrap distribution  $(\hat{\theta}_1, \dots, \hat{\theta}_k)$ .

Let

$$z_0 = \Phi^{-1} \{ \#(\hat{\theta}_i \leq \hat{\theta}) / k \}$$

where  $\#(\hat{\theta}_i \leq \hat{\theta})$  is the number of elements of the bootstrap distribution that are less than or equal to the observed statistic and  $\Phi$  is the standard cumulative normal.  $z_0$  is known as the median bias of  $\hat{\theta}$ . Let

$$a = \frac{\sum_{i=1}^n (\bar{\theta}_{(\cdot)} - \hat{\theta}_{(i)})^3}{6 \{ \sum_{i=1}^n (\bar{\theta}_{(\cdot)} - \hat{\theta}_{(i)})^2 \}^{3/2}}$$

where  $\hat{\theta}_{(i)}$  are the leave-one-out (jackknife) estimates of  $\hat{\theta}$  and  $\bar{\theta}_{(\cdot)}$  is their mean. This expression is known as the jackknife estimate of acceleration for  $\hat{\theta}$ . Let

$$p_1 = \Phi \left\{ z_0 + \frac{z_0 - z_{1-\alpha/2}}{1 - a(z_0 - z_{1-\alpha/2})} \right\}$$

$$p_2 = \Phi \left\{ z_0 + \frac{z_0 + z_{1-\alpha/2}}{1 - a(z_0 + z_{1-\alpha/2})} \right\}$$

where  $z_{1-\alpha/2}$  is the  $(1-\alpha/2)$ th quantile of the normal distribution. The bias-corrected and accelerated ( $BC_a$ ) method yields confidence intervals

$$[\theta_{p_1}^*, \theta_{p_2}^*]$$

where  $\theta_p^*$  is the  $p$ th quantile of the bootstrap distribution as defined previously. The bias-corrected (but not accelerated) method is a special case of  $BC_a$  with  $a = 0$ .

## References

- Cameron, A. C., and P. K. Trivedi. 2009. *Microeconometrics Using Stata*. College Station, TX: Stata Press.
- Davison, A. C., and D. V. Hinkley. 1997. *Bootstrap Methods and Their Application*. Cambridge: Cambridge University Press.
- Efron, B. 1979. Bootstrap methods: Another look at the jackknife. *Annals of Statistics* 7: 1–26.
- . 1982. *The Jackknife, the Bootstrap and Other Resampling Plans*. Philadelphia: Society for Industrial and Applied Mathematics.
- Efron, B., and C. Stein. 1981. The jackknife estimate of variance. *Annals of Statistics* 9: 586–596.
- Efron, B., and R. J. Tibshirani. 1986. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical Science* 1: 54–77.
- . 1993. *An Introduction to the Bootstrap*. New York: Chapman & Hall/CRC.
- Gleason, J. R. 1997. `ip18: A command for randomly resampling a dataset`. *Stata Technical Bulletin* 37: 17–22. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 77–83. College Station, TX: Stata Press.
- . 1999. `ip18.1: Update to resample`. *Stata Technical Bulletin* 52: 9–10. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, p. 119. College Station, TX: Stata Press.
- Gould, W. W. 1994. `ssi6.2: Faster and easier bootstrap estimation`. *Stata Technical Bulletin* 21: 24–33. Reprinted in *Stata Technical Bulletin Reprints*, vol. 4, pp. 211–223. College Station, TX: Stata Press.
- Guan, W. 2003. `From the help desk: Bootstrapped standard errors`. *Stata Journal* 3: 71–80.
- Hall, P., and S. R. Wilson. 1991. Two guidelines for bootstrap hypothesis testing. *Biometrics* 47: 757–762.
- Hamilton, L. C. 1991. `ssi2: Bootstrap programming`. *Stata Technical Bulletin* 4: 18–27. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 208–220. College Station, TX: Stata Press.
- . 1992. *Regression with Graphics: A Second Course in Applied Statistics*. Belmont, CA: Duxbury.
- . 2009. *Statistics with Stata (Updated for Version 10)*. Belmont, CA: Brooks/Cole.
- Hinkley, D. V. 1978. Improving the jackknife with special reference to correlation estimation. *Biometrika* 65: 13–22.
- Holmes, S., C. Morris, and R. J. Tibshirani. 2003. Bradley Efron: A conversation with good friends. *Statistical Science* 18: 268–281.
- Mooney, C. Z., and R. D. Duval. 1993. *Bootstrapping: A Nonparametric Approach to Statistical Inference*. Newbury Park, CA: Sage.
- Poi, B. P. 2004. `From the help desk: Some bootstrapping techniques`. *Stata Journal* 4: 312–328.
- Stine, R. 1990. An introduction to bootstrap methods: Examples and ideas. In *Modern Methods of Data Analysis*, ed. J. Fox and J. S. Long, 353–373. Newbury Park, CA: Sage.

## Also see

[R] [bootstrap postestimation](#) — Postestimation tools for bootstrap

[R] [jackknife](#) — Jackknife estimation

[R] [permute](#) — Monte Carlo permutation tests

[R] [simulate](#) — Monte Carlo simulations

[U] [13.5 Accessing coefficients and standard errors](#)

[U] [13.6 Accessing results from Stata commands](#)

[U] [20 Estimation and postestimation commands](#)



Description

The following postestimation command is of special interest after `bootstrap`:

| command                      | description                                   |
|------------------------------|-----------------------------------------------|
| <code>estat bootstrap</code> | percentile-based and bias-corrected CI tables |

For information about `estat bootstrap`, see below.

The following standard postestimation commands are also available:

| command                  | description                                                                                         |
|--------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>       | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estimates</code>   | cataloging estimation results                                                                       |
| * <code>hausman</code>   | Hausman’s specification test                                                                        |
| * <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| * <code>margins</code>   | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| * <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| * <code>predict</code>   | predictions, residuals, influence statistics, and other diagnostic measures                         |
| * <code>predictnl</code> | point estimates, standard errors, testing, and inference for generalized predictions                |
| * <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| * <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

\*This postestimation command is allowed if it may be used after *command*.

See the corresponding entries in the *Stata Base Reference Manual* for details.

Special-interest postestimation command

`estat bootstrap` displays a table of confidence intervals for each statistic from a bootstrap analysis.

Syntax for predict

The syntax of `predict` (and even if `predict` is allowed) following `bootstrap` depends upon the *command* used with `bootstrap`. If `predict` is not allowed, neither is `predictnl`.

## Syntax for estat bootstrap

estat bootstrap [ , *options* ]

| <i>options</i>    | description                                   |
|-------------------|-----------------------------------------------|
| bc                | bias-corrected CIs; the default               |
| bca               | bias-corrected and accelerated ( $BC_a$ ) CIs |
| <u>normal</u>     | normal-based CIs                              |
| <u>percentile</u> | percentile CIs                                |
| all               | all available CIs                             |
| <u>noheader</u>   | suppress table header                         |
| <u>nolegend</u>   | suppress table legend                         |
| <u>verbose</u>    | display the full table legend                 |

bc, bca, normal, and percentile may be used together.

## Menu

Statistics > Postestimation > Reports and statistics

## Options for estat bootstrap

- bc is the default and displays bias-corrected confidence intervals.
- bca displays bias-corrected and accelerated confidence intervals. This option assumes that you also specified the bca option on the bootstrap prefix command.
- normal displays normal approximation confidence intervals.
- percentile displays percentile confidence intervals.
- all displays all available confidence intervals.
- noheader suppresses display of the table header. This option implies nolegend.
- nolegend suppresses display of the table legend, which identifies the rows of the table with the expressions they represent.
- verbose requests that the full table legend be displayed.

## Remarks

### ► Example 1

The estat bootstrap postestimation command produces a table containing the observed value of the statistic, an estimate of its bias, the bootstrap standard error, and up to four different confidence intervals.

If we were interested merely in getting bootstrap standard errors for the model coefficients, we could use the bootstrap prefix with our estimation command. If we were interested in performing a thorough bootstrap analysis of the model coefficients, we could use the estat bootstrap postestimation command after fitting the model with the bootstrap prefix.

Using [example 1](#) from [\[R\] bootstrap](#), we need many more replications for the confidence interval types other than the normal based, so let’s rerun the estimation command. We will reset the random-number seed—in case we wish to reproduce the results—increase the number of replications, and save the bootstrap distribution as a dataset called `bsauto.dta`.

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. set seed 1
. bootstrap _b, reps(1000) saving(bsauto) bca: regress mpg weight gear foreign
(output omitted)
. estat bootstrap, all
```

Linear regression

Number of obs = 74

Replications = 1000

| mpg        | Observed<br>Coef. | Bias      | Bootstrap<br>Std. Err. | [95% Conf. Interval] |            |       |
|------------|-------------------|-----------|------------------------|----------------------|------------|-------|
| weight     | -.00613903        | .0000567  | .000628                | -0.0073699           | -0.0049082 | (N)   |
|            |                   |           |                        | -0.0073044           | -0.0048548 | (P)   |
|            |                   |           |                        | -0.0074355           | -0.004928  | (BC)  |
|            |                   |           |                        | -0.0075282           | -0.0050258 | (BCa) |
| gear_ratio | 1.4571134         | .1051696  | 1.4554785              | -1.395572            | 4.309799   | (N)   |
|            |                   |           |                        | -1.262111            | 4.585372   | (P)   |
|            |                   |           |                        | -1.523927            | 4.174376   | (BC)  |
|            |                   |           |                        | -1.492223            | 4.231356   | (BCa) |
| foreign    | -2.2216815        | -.0196361 | 1.2023286              | -4.578202            | .1348393   | (N)   |
|            |                   |           |                        | -4.442199            | .2677989   | (P)   |
|            |                   |           |                        | -4.155504            | .6170642   | (BC)  |
|            |                   |           |                        | -4.216531            | .5743973   | (BCa) |
| _cons      | 36.101353         | -.502281  | 5.4089441              | 25.50002             | 46.70269   | (N)   |
|            |                   |           |                        | 24.48569             | 46.07086   | (P)   |
|            |                   |           |                        | 25.59799             | 46.63227   | (BC)  |
|            |                   |           |                        | 25.85658             | 47.02108   | (BCa) |

- (N) normal confidence interval
- (P) percentile confidence interval
- (BC) bias-corrected confidence interval
- (BCa) bias-corrected and accelerated confidence interval

The estimated standard errors here differ from our previous estimates using only 100 replications by, respectively, 8%, 3%, 11%, and 6%; see [example 1](#) of [\[R\] bootstrap](#). So much for our advice that 50–200 replications are good enough to estimate standard errors. Well, the more replications the better—that advice you should believe.

Which of the methods to compute confidence intervals should we use? If the statistic is unbiased, the percentile (P) and bias-corrected (BC) methods should give similar results. The bias-corrected confidence interval will be the same as the percentile confidence interval when the observed value of the statistic is equal to the median of the bootstrap distribution. Thus, for unbiased statistics, the two methods should give similar results as the number of replications becomes large. For biased statistics, the bias-corrected method should yield confidence intervals with better coverage probability (closer to the nominal value of 95% or whatever was specified) than the percentile method. For statistics with variances that vary as a function of the parameter of interest, the bias-corrected and accelerated method (BC<sub>a</sub>) will typically have better coverage probability than the others.

When the bootstrap distribution is approximately normal, all these methods should give similar confidence intervals as the number of replications becomes large. If we examine the normality of these bootstrap distributions using, say, the `pnorm` command (see [\[R\] diagnostic plots](#)), we see that they closely follow a normal distribution. Thus here, the normal approximation would also be a valid

choice. The chief advantage of the normal-approximation method is that it (supposedly) requires fewer replications than the other methods. Of course, it should be used only when the bootstrap distribution exhibits normality.

We can load `bsauto.dta` containing the bootstrap distributions for these coefficients:

```
. use bsauto
(bootstrap: regress)
. describe *
```

| variable name              | storage<br>type | display<br>format | value<br>label | variable label              |
|----------------------------|-----------------|-------------------|----------------|-----------------------------|
| <code>_b_weight</code>     | float           | %9.0g             |                | <code>_b[weight]</code>     |
| <code>_b_gear_ratio</code> | float           | %9.0g             |                | <code>_b[gear_ratio]</code> |
| <code>_b_foreign</code>    | float           | %9.0g             |                | <code>_b[foreign]</code>    |
| <code>_b_cons</code>       | float           | %9.0g             |                | <code>_b[_cons]</code>      |

We can now run other commands, such as `pnorm`, on the bootstrap distributions. As with all standard estimation commands, we can use the `bootstrap` command to replay its output table. The default variable names assigned to the statistics in `exp_list` are `_bs_1`, `_bs_2`, ..., and each variable is labeled with the associated expression. The naming convention for the extended expressions `_b` and `_se` is to prepend `_b_` and `_se_`, respectively, onto the name of each element of the coefficient vector. Here the first coefficient is `_b[weight]`, so `bootstrap` named it `_b_weight`.



## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

- [R] [bootstrap](#) — Bootstrap sampling and estimation
- [U] [20 Estimation and postestimation commands](#)

Syntax

```
boxcox depvar [indepvars] [if] [in] [weight] [, options]
```

| <i>options</i>                       | description                                                 |
|--------------------------------------|-------------------------------------------------------------|
| Model                                |                                                             |
| <code>noconstant</code>              | suppress constant term                                      |
| <code>model(lhsonly)</code>          | left-hand-side Box–Cox model; the default                   |
| <code>model(rhsonly)</code>          | right-hand-side Box–Cox model                               |
| <code>model(lambda)</code>           | both sides Box–Cox model with same parameter                |
| <code>model(theta)</code>            | both sides Box–Cox model with different parameters          |
| <code>notrans(<i>varlist</i>)</code> | nontransformed independent variables                        |
| Reporting                            |                                                             |
| <code>level(#)</code>                | set confidence level; default is <code>level(95)</code>     |
| <code>lrtest</code>                  | perform likelihood-ratio test                               |
| Maximization                         |                                                             |
| <code>nolog</code>                   | suppress full-model iteration log                           |
| <code>nologlr</code>                 | suppress restricted-model <code>lrtest</code> iteration log |
| <code>maximize_options</code>        | control the maximization process; seldom used               |

*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.  
bootstrap, by, jackknife, rolling, statsby, and xi are allowed; see [U] 11.1.10 Prefix commands.  
Weights are not allowed with the bootstrap prefix.  
fweights and iweights are allowed; see [U] 11.1.6 weight.  
See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Linear models and related > Box-Cox regression

Description

boxcox finds the maximum likelihood estimates of the parameters of the Box–Cox transform, the coefficients on the independent variables, and the standard deviation of the normally distributed errors for a model in which *depvar* is regressed on *indepvars*. You can fit the following models:

| Option            | Estimates                                                                                                                                                                   |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lhsonly           | $y_j^{(\theta)} = \beta_1 x_{1j} + \beta_2 x_{2j} + \cdots + \beta_k x_{kj} + \epsilon_j$                                                                                   |
| rhsonly           | $y_j = \beta_1 x_{1j}^{(\lambda)} + \beta_2 x_{2j}^{(\lambda)} + \cdots + \beta_k x_{kj}^{(\lambda)} + \epsilon_j$                                                          |
| rhsonly notrans() | $y_j = \beta_1 x_{1j}^{(\lambda)} + \beta_2 x_{2j}^{(\lambda)} + \cdots + \beta_k x_{kj}^{(\lambda)} + \gamma_1 z_{1j} + \cdots + \gamma_l z_{lj} + \epsilon_j$             |
| lambda            | $y_j^{(\lambda)} = \beta_1 x_{1j}^{(\lambda)} + \beta_2 x_{2j}^{(\lambda)} + \cdots + \beta_k x_{kj}^{(\lambda)} + \epsilon_j$                                              |
| lambda notrans()  | $y_j^{(\lambda)} = \beta_1 x_{1j}^{(\lambda)} + \beta_2 x_{2j}^{(\lambda)} + \cdots + \beta_k x_{kj}^{(\lambda)} + \gamma_1 z_{1j} + \cdots + \gamma_l z_{lj} + \epsilon_j$ |
| theta             | $y_j^{(\theta)} = \beta_1 x_{1j}^{(\lambda)} + \beta_2 x_{2j}^{(\lambda)} + \cdots + \beta_k x_{kj}^{(\lambda)} + \epsilon_j$                                               |
| theta notrans()   | $y_j^{(\theta)} = \beta_1 x_{1j}^{(\lambda)} + \beta_2 x_{2j}^{(\lambda)} + \cdots + \beta_k x_{kj}^{(\lambda)} + \gamma_1 z_{1j} + \cdots + \gamma_l z_{lj} + \epsilon_j$  |

Any variable to be transformed must be strictly positive.

## Options

### Model

- `noconstant`; see [\[R\] estimation options](#).
- `model(lhsonly | rhsonly | lambda | theta)` specifies which of the four models to fit.
- `model(lhsonly)` applies the Box–Cox transform to *depvar* only. `model(lhsonly)` is the default.
  - `model(rhsonly)` applies the transform to the *indepvars* only.
  - `model(lambda)` applies the transform to both *depvar* and *indepvars*, and they are transformed by the same parameter.
  - `model(theta)` applies the transform to both *depvar* and *indepvars*, but this time, each side is transformed by a separate parameter.
- `notrans(varlist)` specifies that the variables in *varlist* be included as nontransformed independent variables.

### Reporting

- `level(#)`; see [\[R\] estimation options](#).
- `lrttest` specifies that a likelihood-ratio test of significance be performed and reported for each independent variable.

### Maximization

- `nolog` suppresses the iteration log when fitting the full model.
- `nologlr` suppresses the iteration log when fitting the restricted models required by the `lrttest` option.
- maximize\_options*: `iterate(#)` and `from(init_specs)`; see [\[R\] maximize](#).

| Model   | Initial value specification                                           |
|---------|-----------------------------------------------------------------------|
| lhsonly | <code>from(<math>\theta_0</math>, copy)</code>                        |
| rhsonly | <code>from(<math>\lambda_0</math>, copy)</code>                       |
| lambda  | <code>from(<math>\lambda_0</math>, copy)</code>                       |
| theta   | <code>from(<math>\lambda_0</math> <math>\theta_0</math>, copy)</code> |

## Remarks

Remarks are presented under the following headings:

[Introduction](#)  
[Theta model](#)  
[Lambda model](#)  
[Left-hand-side-only model](#)  
[Right-hand-side-only model](#)

## Introduction

The Box–Cox transform

$$y^{(\lambda)} = \frac{y^\lambda - 1}{\lambda}$$

has been widely used in applied data analysis. [Box and Cox \(1964\)](#) developed the transformation and argued that the transformation could make the residuals more closely normal and less heteroskedastic. [Cook and Weisberg \(1982\)](#) discuss the transform in this light. Because the transform embeds several popular functional forms, it has received some attention as a method for testing functional forms, in particular,

$$y^{(\lambda)} = \begin{cases} y - 1 & \text{if } \lambda = 1 \\ \ln(y) & \text{if } \lambda = 0 \\ 1 - 1/y & \text{if } \lambda = -1 \end{cases}$$

[Davidson and MacKinnon \(1993\)](#) discuss this use of the transform. [Atkinson \(1985\)](#) also gives a good general treatment.

## Theta model

`boxcox` obtains the maximum likelihood estimates of the parameters for four different models. The most general of the models, the `theta` model, is

$$y_j^{(\theta)} = \beta_0 + \beta_1 x_{1j}^{(\lambda)} + \beta_2 x_{2j}^{(\lambda)} + \cdots + \beta_k x_{kj}^{(\lambda)} + \gamma_1 z_{1j} + \gamma_2 z_{2j} + \cdots + \gamma_l z_{lj} + \epsilon_j$$

where  $\epsilon \sim N(0, \sigma^2)$ . Here the dependent variable,  $y$ , is subject to a Box–Cox transform with parameter  $\theta$ . Each of the *indepvars*,  $x_1, x_2, \dots, x_k$ , is transformed by a Box–Cox transform with parameter  $\lambda$ . The  $z_1, z_2, \dots, z_l$  specified in the `notrans()` option are independent variables that are not transformed.

[Box and Cox \(1964\)](#) argued that this transformation would leave behind residuals that more closely follow a normal distribution than those produced by a simple linear regression model. Bear in mind that the normality of  $\epsilon$  is assumed and that `boxcox` obtains maximum likelihood estimates of the  $k + l + 4$  parameters under this assumption. `boxcox` does not choose  $\lambda$  and  $\theta$  so that the residuals are approximately normally distributed. If you are interested in this type of transformation to normality, see the official Stata commands `lnskew0` and `bcskew0` in [\[R\] lnskew0](#). However, those commands work on a more restrictive model in which none of the independent variables is transformed.

➤ Example 1

Consider an example using the auto data.

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. boxcox mpg weight price, notrans(foreign) model(theta) lrtest
Fitting comparison model
Iteration 0: log likelihood = -234.39434
Iteration 1: log likelihood = -228.26891
Iteration 2: log likelihood = -228.26777
Iteration 3: log likelihood = -228.26777
Fitting full model
Iteration 0: log likelihood = -194.13727
(output omitted)
Fitting comparison models for LR tests
Iteration 0: log likelihood = -179.58214
Iteration 1: log likelihood = -177.59036
Iteration 2: log likelihood = -177.58739
Iteration 3: log likelihood = -177.58739
Iteration 0: log likelihood = -203.92855
Iteration 1: log likelihood = -201.30202
Iteration 2: log likelihood = -201.18359
Iteration 3: log likelihood = -201.18233
Iteration 4: log likelihood = -201.18233
Iteration 0: log likelihood = -178.83799
Iteration 1: log likelihood = -175.98405
Iteration 2: log likelihood = -175.97931
Iteration 3: log likelihood = -175.97931

 Number of obs = 74
 LR chi2(4) = 105.19
 Prob > chi2 = 0.000

Log likelihood = -175.67343
```

| mpg     | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|---------|-----------|-----------|-------|-------|----------------------|-----------|
| /lambda | .760169   | .6289991  | 1.21  | 0.227 | -.4726466            | 1.992985  |
| /theta  | -.7189315 | .3244439  | -2.22 | 0.027 | -1.35483             | -.0830331 |

Estimates of scale-variant parameters

|         | Coef.     | chi2(df) | P>chi2(df) | df of chi2 |
|---------|-----------|----------|------------|------------|
| Notrans |           |          |            |            |
| foreign | -.0114338 | 3.828    | 0.050      | 1          |
| _cons   | 1.377399  |          |            |            |
| Trans   |           |          |            |            |
| weight  | -.000239  | 51.018   | 0.000      | 1          |
| price   | -6.18e-06 | 0.612    | 0.434      | 1          |
| /sigma  | .0143509  |          |            |            |

| Test<br>H0:       | Restricted<br>log likelihood | chi2  | Prob > chi2 |
|-------------------|------------------------------|-------|-------------|
| theta=lambda = -1 | -181.64479                   | 11.94 | 0.001       |
| theta=lambda = 0  | -178.2406                    | 5.13  | 0.023       |
| theta=lambda = 1  | -194.13727                   | 36.93 | 0.000       |



The output is composed of the iteration logs and three distinct tables. The first table contains a standard header for a maximum likelihood estimator and a standard output table for the Box–Cox transform parameters. The second table contains the estimates of the scale-variant parameters. The third table contains the output from likelihood-ratio tests on three standard functional form specifications.

If we were to interpret this output, the right-hand-side transformation would not significantly add to the regression, whereas the left-hand-side transformation would make the 5% but not the 1% cutoff. `price` is certainly not significant, and `foreign` lies right on the 5% cutoff. `weight` is clearly significant. The output also shows that the linear and multiplicative inverse specifications are both strongly rejected. A natural log specification can be rejected at the 5% level but not at the 1% level.

◀

## □ Technical note

Spitzer (1984) showed that the Wald tests of the joint significance of the coefficients of the right-hand-side variables, either transformed or untransformed, are not invariant to changes in the scale of the transformed dependent variable. Davidson and MacKinnon (1993) also discuss this point. This problem demonstrates that Wald statistics can be manipulated in nonlinear models. Lafontaine and White (1986) analyze this problem numerically, and Phillips and Park (1988) analyze it by using Edgeworth expansions. See Drukker (2000b) for a more detailed discussion of this issue. Because the parameter estimates and their Wald tests are not scale invariant, no Wald tests or confidence intervals are reported for these parameters. However, when the `lrtest` option is specified, likelihood-ratio tests are performed and reported. Schlesselman (1971) showed that, if a constant is included in the model, the parameter estimates of the Box–Cox transforms are scale invariant. For this reason, we strongly recommend that you not use the `noconstant` option.

The `lrtest` option does not perform a likelihood-ratio test on the constant, so no value for this statistic is reported. Unless the data are properly scaled, the restricted model does not often converge. For this reason, no likelihood-ratio test on the constant is performed by the `lrtest` option. However, if you have a special interest in performing this test, you can do so by fitting the constrained model separately. If problems with convergence are encountered, rescaling the data by their means may help.

□

## Lambda model

A less general model than the one above is called the `lambda` model. It specifies that the same parameter be used in both the left-hand-side and right-hand-side transformations. Specifically,

$$y_j^{(\lambda)} = \beta_0 + \beta_1 x_{1j}^{(\lambda)} + \beta_2 x_{2j}^{(\lambda)} + \cdots + \beta_k x_{kj}^{(\lambda)} + \gamma_1 z_{1j} + \gamma_2 z_{2j} + \cdots + \gamma_l z_{lj} + \epsilon_j$$

where  $\epsilon \sim N(0, \sigma^2)$ . Here the `depvar` variable,  $y$ , and each of the `indepvars`,  $x_1, x_2, \dots, x_k$ , is transformed by a Box–Cox transform with the common parameter  $\lambda$ . Again the  $z_1, z_2, \dots, z_l$  are independent variables that are not transformed.

### Left-hand-side-only model

Even more restrictive than a common transformation parameter is transforming the dependent variable only. Because the dependent variable is on the left-hand side of the equation, this model is known as the `lhsonly` model. Here you are estimating the parameters of the model

$$y_j^{(\theta)} = \beta_0 + \beta_1 x_{1j} + \beta_2 x_{2j} + \cdots + \beta_k x_{kj} + \epsilon_j$$

where  $\epsilon \sim N(0, \sigma^2)$ . Here only the `depvar`, `y`, is transformed by a Box–Cox transform with the parameter  $\theta$ .

#### Example 2

We again hypothesize `mpg` to be a function of `weight`, `price`, and `foreign` in a Box–Cox model in which only `mpg` is subject to the transform:

```
. boxcox mpg weight price foreign, model(lhs) lrtest nolog nologlr
Fitting comparison model
Fitting full model
Fitting comparison models for LR tests

 Number of obs = 74
 LR chi2(3) = 105.04
 Prob > chi2 = 0.000

Log likelihood = -175.74705
```

| mpg    | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|--------|-----------|-----------|-------|-------|----------------------|-----------|
| /theta | -.7826999 | .281954   | -2.78 | 0.006 | -1.33532             | -.2300802 |

Estimates of scale-variant parameters

|         | Coef.     | chi2(df) | P>chi2(df) | df of chi2 |
|---------|-----------|----------|------------|------------|
| Notrans |           |          |            |            |
| weight  | -.0000294 | 58.056   | 0.000      | 1          |
| price   | -4.66e-07 | 0.469    | 0.493      | 1          |
| foreign | -.0097564 | 4.644    | 0.031      | 1          |
| _cons   | 1.249845  |          |            |            |
| /sigma  | .0118454  |          |            |            |

| Test       | Restricted     | LR statistic | P-value     |
|------------|----------------|--------------|-------------|
| H0:        | log likelihood | chi2         | Prob > chi2 |
| theta = -1 | -176.04312     | 0.59         | 0.442       |
| theta = 0  | -179.54104     | 7.59         | 0.006       |
| theta = 1  | -194.13727     | 36.78        | 0.000       |

This model rejects both linear and log specifications of `mpg` but fails to reject the hypothesis that `1/mpg` is linear in the independent variables. These findings are in line with what an engineer would have expected. In engineering terms, gallons per mile represents actual energy consumption, and energy consumption should be approximately linear in weight.

## Right-hand-side-only model

The fourth model leaves the *depvar* alone and transforms a subset of the *indepvars* using the parameter  $\lambda$ . This is the *rhsonly* model. In this model, the *depvar*,  $y$ , is given by

$$y_j = \beta_0 + \beta_1 x_{1j}^{(\lambda)} + \beta_2 x_{2j}^{(\lambda)} + \cdots + \beta_k x_{kj}^{(\lambda)} + \gamma_1 z_{1j} + \gamma_2 z_{2j} + \cdots + \gamma_l z_{lj} + \epsilon_j$$

where  $\epsilon \sim N(0, \sigma^2)$ . Here each of the *indepvars*,  $x_1, x_2, \dots, x_k$ , is transformed by a Box–Cox transform with the parameter  $\lambda$ . Again the  $z_1, z_2, \dots, z_l$  are independent variables that are not transformed.

### ► Example 3

Here is an example with the *rhsonly* model. *price* and *foreign* are not included in the list of covariates. (You are invited to use the *auto* data and check that they fare no better here than above.)

```
. boxcox mpg weight, model(rhs) lrtest nolog nologlr
Fitting full model
Fitting comparison models for LR tests
Comparison model for LR test on weight is a linear regression.
Lambda is not identified in the restricted model.
```

|  |               |   |       |
|--|---------------|---|-------|
|  | Number of obs | = | 74    |
|  | LR chi2(2)    | = | 82.90 |
|  | Prob > chi2   | = | 0.000 |

Log likelihood = -192.94368

| mpg     | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|---------|-----------|-----------|-------|-------|----------------------|----------|
| /lambda | -.4460916 | .6551108  | -0.68 | 0.496 | -1.730085            | .8379019 |

Estimates of scale-variant parameters

|         | Coef.     | chi2(df) | P>chi2(df) | df of chi2 |
|---------|-----------|----------|------------|------------|
| Notrans |           |          |            |            |
| _cons   | 1359.092  |          |            |            |
| Trans   |           |          |            |            |
| weight  | -614.3874 | 82.901   | 0.000      | 1          |
| /sigma  | 3.281854  |          |            |            |

| Test<br>H0: | Restricted<br>log likelihood | LR statistic<br>chi2 | P-value<br>Prob > chi2 |
|-------------|------------------------------|----------------------|------------------------|
| lambda = -1 | -193.2893                    | 0.69                 | 0.406                  |
| lambda = 0  | -193.17892                   | 0.47                 | 0.493                  |
| lambda = 1  | -195.38869                   | 4.89                 | 0.027                  |

The interpretation of the output is similar to that in all the cases above, with one caveat. As requested, a likelihood-ratio test was performed on the lone independent variable. However, when it is dropped to form the constrained model, the comparison model is not a right-hand-side-only Box–Cox model but rather a simple linear regression on a constant model. When *weight* is dropped, there are no longer any transformed variables. Hence,  $\lambda$  is not identified, and it must also be dropped. This process leaves a linear regression on a constant as the “comparison model”. It also implies that the test statistic has 2 degrees of freedom instead of 1. At the top of the output, a more concise warning informs you of this point.

A similar identification issue can also arise in the `lambda` and `theta` models when only one independent variable is specified. In these cases, warnings also appear on the output.



## Saved results

`boxcox` saves the following in `e()`:

### Scalars

|                         |                                                 |                          |                                                  |
|-------------------------|-------------------------------------------------|--------------------------|--------------------------------------------------|
| <code>e(N)</code>       | number of observations                          | <code>e(l1_tm1)</code>   | log likelihood of model $\lambda=\theta=-1$      |
| <code>e(l1)</code>      | log likelihood                                  | <code>e(chi2_tm1)</code> | LR of $\lambda=\theta=-1$ vs. full model         |
| <code>e(chi2)</code>    | LR statistic of full vs. comparison             | <code>e(p_tm1)</code>    | $p$ -value of $\lambda=\theta=-1$ vs. full model |
| <code>e(df_m)</code>    | full model degrees of freedom                   | <code>e(l1_t0)</code>    | log likelihood of model $\lambda=\theta=0$       |
| <code>e(l10)</code>     | log likelihood of the restricted model          | <code>e(chi2_t0)</code>  | LR of $\lambda=\theta=0$ vs. full model          |
| <code>e(df_r)</code>    | restricted model degrees of freedom             | <code>e(p_t0)</code>     | $p$ -value of $\lambda=\theta=0$ vs. full model  |
| <code>e(l1_t1)</code>   | log likelihood of model $\lambda=\theta=1$      | <code>e(rank)</code>     | rank of <code>e(V)</code>                        |
| <code>e(chi2_t1)</code> | LR of $\lambda=\theta=1$ vs. full model         | <code>e(ic)</code>       | number of iterations                             |
| <code>e(p_t1)</code>    | $p$ -value of $\lambda=\theta=1$ vs. full model | <code>e(rc)</code>       | return code                                      |

### Macros

|                         |                                                                                         |                              |                                                |
|-------------------------|-----------------------------------------------------------------------------------------|------------------------------|------------------------------------------------|
| <code>e(cmd)</code>     | <code>boxcox</code>                                                                     | <code>e(chi2type)</code>     | LR; type of model $\chi^2$ test                |
| <code>e(cmdline)</code> | command as typed                                                                        | <code>e(lrtest)</code>       | <code>lrtest</code> , if requested             |
| <code>e(depvar)</code>  | name of dependent variable                                                              | <code>e(properties)</code>   | <code>b V</code>                               |
| <code>e(model)</code>   | <code>lhonly</code> , <code>rhonly</code> , <code>lambda</code> , or <code>theta</code> | <code>e(predict)</code>      | program used to implement <code>predict</code> |
| <code>e(wtype)</code>   | weight type                                                                             |                              |                                                |
| <code>e(wexp)</code>    | weight expression                                                                       | <code>e(marginsnotok)</code> | predictions disallowed by <code>margins</code> |
| <code>e(ntrans)</code>  | <code>yes</code> if nontransformed <i>indepvars</i>                                     |                              |                                                |

### Matrices

|                    |                                                               |                       |                                                    |
|--------------------|---------------------------------------------------------------|-----------------------|----------------------------------------------------|
| <code>e(b)</code>  | coefficient vector                                            | <code>e(df)</code>    | degrees of freedom of LR tests on <i>indepvars</i> |
| <code>e(V)</code>  | variance–covariance matrix of the estimators (see note below) | <code>e(chi2m)</code> | LR statistics for tests on <i>indepvars</i>        |
| <code>e(pm)</code> | $p$ -values for LR tests on <i>indepvars</i>                  |                       |                                                    |

### Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

`e(V)` contains all zeros, except for the elements that correspond to the parameters of the Box–Cox transform.

## Methods and formulas

`boxcox` is implemented as an ado-file.

In the internal computations,

$$y^{(\lambda)} = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \text{if } |\lambda| > 10^{-10} \\ \ln(y) & \text{otherwise} \end{cases}$$

The unconcentrated log likelihood for the `theta` model is

$$\ln L = \left( \frac{-N}{2} \right) \{ \ln(2\pi) + \ln(\sigma^2) \} + (\theta - 1) \sum_{i=1}^N \ln(y_i) - \left( \frac{1}{2\sigma^2} \right) \text{SSR}$$

where

$$\text{SSR} = \sum_{i=1}^N (y_i^{(\theta)} - \beta_0 + \beta_1 x_{i1}^{(\lambda)} + \beta_2 x_{i2}^{(\lambda)} + \cdots + \beta_k x_{ik}^{(\lambda)} + \gamma_1 z_{i1} + \gamma_2 z_{i2} + \cdots + \gamma_l z_{il})^2$$

Writing the SSR in matrix form,

$$\text{SSR} = (\mathbf{Y}^{(\theta)} - \mathbf{X}^{(\lambda)} \mathbf{b}' - \mathbf{Z} \mathbf{g}')' (\mathbf{Y}^{(\theta)} - \mathbf{X}^{(\lambda)} \mathbf{b}' - \mathbf{Z} \mathbf{g}')$$

where  $\mathbf{Y}^{(\theta)}$  is an  $N \times 1$  vector of elementwise transformed data,  $\mathbf{X}^{(\lambda)}$  is an  $N \times k$  matrix of elementwise transformed data,  $\mathbf{Z}$  is an  $N \times l$  matrix of untransformed data,  $\mathbf{b}$  is a  $1 \times k$  vector of coefficients, and  $\mathbf{g}$  is a  $1 \times l$  vector of coefficients. Letting

$$\mathbf{W}_\lambda = \begin{pmatrix} \mathbf{X}^{(\lambda)} & \mathbf{Z} \end{pmatrix}$$

be the horizontal concatenation of  $\mathbf{X}^{(\lambda)}$  and  $\mathbf{Z}$  and

$$\mathbf{d}' = \begin{pmatrix} \mathbf{b}' \\ \mathbf{g}' \end{pmatrix}$$

be the vertical concatenation of the coefficients yields

$$\text{SSR} = (\mathbf{Y}^{(\theta)} - \mathbf{W}_\lambda \mathbf{d}')' (\mathbf{Y}^{(\theta)} - \mathbf{W}_\lambda \mathbf{d}')$$

For given values of  $\lambda$  and  $\theta$ , the solutions for  $\mathbf{d}'$  and  $\sigma^2$  are

$$\hat{\mathbf{d}}' = (\mathbf{W}_\lambda' \mathbf{W}_\lambda)^{-1} \mathbf{W}_\lambda' \mathbf{Y}^{(\theta)}$$

and

$$\hat{\sigma}^2 = \frac{1}{N} \left( \mathbf{Y}^{(\theta)} - \mathbf{W}_\lambda \hat{\mathbf{d}}' \right)' \left( \mathbf{Y}^{(\theta)} - \mathbf{W}_\lambda \hat{\mathbf{d}}' \right)$$

Substituting these solutions into the log-likelihood function yields the concentrated log-likelihood function

$$\ln L_c = \left( -\frac{N}{2} \right) \{ \ln(2\pi) + 1 + \ln(\hat{\sigma}^2) \} + (\theta - 1) \sum_{i=1}^N \ln(y_i)$$

Similar calculations yield the concentrated log-likelihood function for the `lambda` model,

$$\ln L_c = \left(-\frac{N}{2}\right) \{ \ln(2\pi) + 1 + \ln(\hat{\sigma}^2) \} + (\lambda - 1) \sum_{i=1}^N \ln(y_i)$$

the `lhsonly` model,

$$\ln L_c = \left(-\frac{N}{2}\right) \{ \ln(2\pi) + 1 + \ln(\hat{\sigma}^2) \} + (\theta - 1) \sum_{i=1}^N \ln(y_i)$$

and the `rhsonly` model,

$$\ln L_c = \left(-\frac{N}{2}\right) \{ \ln(2\pi) + 1 + \ln(\hat{\sigma}^2) \}$$

where  $\hat{\sigma}^2$  is specific to each model and is defined analogously to that in the `theta` model.

## References

- Atkinson, A. C. 1985. *Plots, Transformations, and Regression: An Introduction to Graphical Methods of Diagnostic Regression Analysis*. Oxford: Oxford University Press.
- Box, G. E. P., and D. R. Cox. 1964. An analysis of transformations. *Journal of the Royal Statistical Society, Series B* 26: 211–252.
- Carroll, R. J., and D. Ruppert. 1988. *Transformation and Weighting in Regression*. New York: Chapman & Hall.
- Cook, R. D., and S. Weisberg. 1982. *Residuals and Influence in Regression*. New York: Chapman & Hall/CRC.
- Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.
- Drukker, D. M. 2000a. [sg130: Box–Cox regression models](#). *Stata Technical Bulletin* 54: 27–36. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 307–319. College Station, TX: Stata Press.
- . 2000b. [sg131: On the manipulability of Wald tests in Box–Cox regression models](#). *Stata Technical Bulletin* 54: 36–42. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 319–327. College Station, TX: Stata Press.
- Lafontaine, F., and K. J. White. 1986. Obtaining any Wald statistic you want. *Economics Letters* 21: 35–40.
- Phillips, P. C. B., and J. Y. Park. 1988. On the formulation of Wald tests of nonlinear restrictions. *Econometrica* 56: 1065–1083.
- Schlesselman, J. J. 1971. Power families: A note on the Box and Cox transformation. *Journal of the Royal Statistical Society, Series B* 33: 307–311.
- Spitzer, J. J. 1984. Variance estimates in models with the Box–Cox transformation: Implications for estimation and hypothesis testing. *Review of Economics and Statistics* 66: 645–652.

## Also see

- [R] [boxcox postestimation](#) — Postestimation tools for `boxcox`
- [R] [regress](#) — Linear regression
- [R] [lnskew0](#) — Find zero-skewness log or Box–Cox transform
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `boxcox`:

| command                | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>     | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estimates</code> | cataloging estimation results                                                                       |
| * <code>lincom</code>  | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| * <code>nlcom</code>   | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>   | predictions, residuals, influence statistics, and other diagnostic measures                         |
| * <code>test</code>    | Wald tests of simple and composite linear hypotheses                                                |
| * <code>testnl</code>  | Wald tests of nonlinear hypotheses                                                                  |

\*Inference is valid only for hypotheses concerning  $\lambda$  and  $\theta$ .

See the corresponding entries in the *Base Reference Manual* for details.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic]
```

| statistic              | description                                |
|------------------------|--------------------------------------------|
| Main                   |                                            |
| <code>xbt</code>       | transformed linear prediction; the default |
| <code>yhat</code>      | predicted value of $y$                     |
| <code>residuals</code> | residuals                                  |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

Menu

Statistics > Postestimation > Predictions, residuals, etc.

Options for predict

Main

`xbt`, the default, calculates the “linear” prediction. For all the models except `model(1hsonly)`, all the *indepvars* except those specified in the `notrans()` option are transformed.

`yhat` calculates the predicted value of  $y$ .

`residuals` calculates the residuals after the predicted value of  $y$  has been subtracted from the actual value.

## Remarks

`boxcox` estimates variances only for the  $\lambda$  and  $\theta$  parameters (see the [technical note](#) in [\[R\] boxcox](#)), so the extent to which postestimation commands can be used following `boxcox` is limited. Formulas used in `lincom`, `nlcom`, `test`, and `testnl` are dependent on the estimated variances. Therefore, the use of these commands is limited and generally applicable only to inferences on the  $\lambda$  and  $\theta$  coefficients.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[\[R\] boxcox](#) — Box–Cox regression models

[\[R\] lnskew0](#) — Find zero-skewness log or Box–Cox transform

[\[U\] 20 Estimation and postestimation commands](#)



# Title

brier — Brier score decomposition

## Syntax

```
brier outcomevar forecastvar [if] [in] [, group(#)]
```

*by* is allowed; see [\[D\]](#) **by**.

## Menu

Statistics > Epidemiology and related > Other > Brier score decomposition

## Description

**brier** computes the Yates, Sanders, and Murphy decompositions of the Brier Mean Probability Score. *outcomevar* contains 0/1 values reflecting the actual outcome of the experiment, and *forecastvar* contains the corresponding probabilities as predicted by, say, logit, probit, or a human forecaster.

## Option

Main

*group*(#) specifies the number of groups that will be used to compute the decomposition. *group*(10) is the default.

## Remarks

You have a binary (0/1) response and a formula that predicts the corresponding probabilities of having observed a positive outcome (1). If the probabilities were obtained from logistic regression, there are many methods that assess goodness of fit (see, for instance, `estat gof` in [\[R\]](#) **logistic**). However, the probabilities might be computed from a published formula or from a model fit on another sample, both completely unrelated to the data at hand, or perhaps the forecasts are not from a formula at all. In any case, you now have a *test dataset* consisting of the forecast probabilities and observed outcomes. Your test dataset might, for instance, record predictions made by a meteorologist on the probability of rain along with a variable recording whether it actually rained.

The Brier score is an aggregate measure of disagreement between the observed outcome and a prediction—the average squared error difference. The Brier score decomposition is a partition of the Brier score into components that suggest reasons for discrepancy. These reasons fall roughly into three groups: 1) lack of overall calibration between the average predicted probability and the actual probability of the event in your data, 2) misfit of the data in groups defined within your sample, and 3) inability to match actual 0 and 1 responses.

Problem 1 refers to simply overstating or understating the probabilities.

Problem 2 refers to what is standardly called a goodness-of-fit test: the data are grouped, and the predictions for the group are compared with the outcomes.

Problem 3 refers to an individual-level measure of fit. Imagine that the grouped outcomes are predicted on average correctly but that within the group, the outcomes are poorly predicted.

Using logit or probit analysis to fit your data will guarantee that there is no lack of fit due to problem 1, and a good model fitter will be able to avoid problem 2. Problem 3 is inherent in any prediction exercise.

➤ Example 1

We have data on the outcomes of 20 basketball games (win) and the probability of victory predicted by a local pundit (for).

```
. use http://www.stata-press.com/data/r11/bball
. summarize win for
```

| Variable | Obs | Mean  | Std. Dev. | Min | Max |
|----------|-----|-------|-----------|-----|-----|
| win      | 20  | .65   | .4893605  | 0   | 1   |
| for      | 20  | .4785 | .2147526  | .15 | .9  |

```
. brier win for, group(5)
Mean probability of outcome 0.6500
 of forecast 0.4785
Correlation 0.5907
ROC area 0.8791 p = 0.0030
Brier score 0.1828
Spiegelhalter's z-statistic -0.6339 p = 0.7369
Sanders-modified Brier score 0.1861
Sanders resolution 0.1400
Outcome index variance 0.2275
Murphy resolution 0.0875
Reliability-in-the-small 0.0461
Forecast variance 0.0438
Excess forecast variance 0.0285
Minimum forecast variance 0.0153
Reliability-in-the-large 0.0294
2*Forecast-Outcome-Covar 0.1179
```

The mean probabilities of forecast and outcome are simply the mean of the predicted probabilities and the actual outcomes (wins/losses). The correlation is the product-moment correlation between them.

The Brier score measures the total difference between the event (winning) and the forecast probability of that event as an average squared difference. As a benchmark, a perfect forecaster would have a Brier score of 0, a perfect misforecaster (predicts probability of win is 1 when loses and 0 when wins) would have a Brier score of 1, and a fence-sitter (forecasts every game as 50/50) would have a Brier score of 0.25. Our pundit is doing reasonably well.

Spiegelhalter’s *z* statistic is a standard normal test statistic for testing whether an individual Brier score is extreme. The ROC area is the area under the receiver operating curve, and the associated test is a test of whether it is greater than 0.5. The more accurate the forecast probabilities, the larger the ROC area.

The Sanders-modified Brier score measures the difference between a grouped forecast measure and the event, where the data are grouped by sorting the sample on the forecast and dividing it into approximately equally sized groups. The difference between the modified and the unmodified score is typically minimal. For this and the other statistics that require grouping—the Sanders and Murphy resolutions and reliability-in-the-small—to be well-defined, group boundaries are chosen so as not to allocate observations with the same forecast probability to different groups. This task is done by grouping on the forecast using `xtile, n(#)`, with `#` being the number of groups; see [D] [pctile](#).

The Sanders resolution measures error that arises from statistical considerations in evaluating the forecast for a group. A group with all positive or all negative outcomes would have a Sanders resolution of 0; it would most certainly be feasible to predict exactly what happened to each member of the group. If the group had 40% positive responses, on the other hand, a forecast that assigned  $p = 0.4$  to each member of the group would be a good one, and yet, there would be “errors” in the squared difference sense. The “error” would be  $(1 - 0.4)^2$  or  $(0 - 0.4)^2$  for each member. The Sanders resolution is the average across groups of such “expected” errors. The 0.1400 value in our data from an overall Brier score of 0.1828 or 0.1861 suggests that a substantial portion of the “error” in our data is inherent.

Outcome index variance is just the variance of the outcome variable. This is the expected value of the Brier score if all the forecast probabilities were merely the average observed outcome. Remember that a fence-sitter has an expected Brier score of 0.25; a smarter fence sitter (who would guess  $p = 0.65$  for these data) would have a Brier score of 0.2275.

The Murphy resolution measures the variation in the average outcomes across groups. If all groups have the same frequency of positive outcomes, little information in any forecast is possible, and the Murphy resolution is 0. If groups differ markedly, the Murphy resolution is as large as 0.25. The 0.0875 means that there is some variation but not a lot, and 0.0875 is probably higher than in most real cases. If you had groups in your data that varied between 40% and 60% positive outcomes, the Murphy resolution would be 0.01; between 30% and 70%, it would be 0.04.

Reliability-in-the-small measures the error that comes from the average forecast within group not measuring the average outcome within group—a classical goodness-of-fit measure, with 0 meaning a perfect fit and 1 meaning a complete lack of fit. The calculated value of 0.0461 shows some amount of lack of fit. Remember, the number is squared, and we are saying that probabilities could be just more than  $\sqrt{0.0461} = 0.215$  or 21.5% off.

Forecast variance measures the amount of discrimination being attempted—that is, the variation in the forecasted probabilities. A small number indicates a fence-sitter making constant predictions. If the forecasts were from a logistic regression model, forecast variance would tend to increase with the amount of information available. Our pundit shows considerable forecast variance of 0.0438 (standard deviation  $\sqrt{0.0438} = 0.2093$ ), which is in line with the reliability-in-the-small, suggesting that the forecaster is attempting as much variation as is available in these data.

Excess forecast variance is the amount of actual forecast variance over a theoretical minimum. The theoretical minimum—called the minimum forecast variance—corresponds to forecasts of  $p_0$  for observations ultimately observed to be negative responses and  $p_1$  for observations ultimately observed to be positive outcomes. Moreover,  $p_0$  and  $p_1$  are set to the average forecasts made for the ultimate negative and positive outcomes. These predictions would be just as good as the predictions the forecaster did make, and any variation in the actual forecast probabilities above this is useless. If this number is large, above 1%–2%, then the forecaster may be attempting more than is possible. The 0.0285 in our data suggests this possibility.

Reliability-in-the-large measures the discrepancy between the mean forecast and the observed fraction of positive outcomes. This discrepancy will be 0 for forecasts made by most statistical models—at least when measured on the same sample used for estimation—because they, by design, reproduce sample means. For our human pundit, the 0.0294 says that there is a  $\sqrt{0.0294}$ , or 17-percentage-point, difference. (This difference can also be found by calculating the difference in the averages of the observed outcomes and forecast probabilities:  $0.65 - 0.4785 = 0.17$ .) That difference, however, is not significant, as we would see if we typed `ttest win=for`; see [R] [ttest](#). If these data were larger and the bias persisted, this difference would be a critical shortcoming of the forecast.

Twice the forecast-outcome covariance is a measure of how accurately the forecast corresponds to the outcome. The concept is similar to that of  $R$ -squared in linear regression.



## Saved results

`brier` saves the following in `r()`:

|                          |                               |                          |                                       |
|--------------------------|-------------------------------|--------------------------|---------------------------------------|
| Scalars                  |                               |                          |                                       |
| <code>r(p_roc)</code>    | significance of ROC area      | <code>r(murphy)</code>   | Murphy resolution                     |
| <code>r(roc_area)</code> | ROC area                      | <code>r(relinssm)</code> | reliability in the small              |
| <code>r(z)</code>        | Spiegelhalter's $z$ statistic | <code>r(Var_f)</code>    | forecast variance                     |
| <code>r(p)</code>        | significance of $z$ statistic | <code>r(Var_fex)</code>  | excess forecast variance              |
| <code>r(brier)</code>    | Brier score                   | <code>r(Var_fmin)</code> | minimum forecast variance             |
| <code>r(brier_s)</code>  | Sanders-modified Brier score  | <code>r(relinla)</code>  | reliability in the large              |
| <code>r(sanders)</code>  | Sanders resolution            | <code>r(cov_2f)</code>   | $2\times$ forecast-outcome-covariance |
| <code>r(oiv)</code>      | outcome index variance        |                          |                                       |

## Methods and formulas

`brier` is implemented as an ado-file.

See [Wilks \(2006, 284–287, 289–292, 298–299\)](#) or [Schmidt and Griffith \(2005\)](#) for a discussion of the Brier score.

Let  $d_j, j = 1, \dots, N$ , be the observed outcomes with  $d_j = 0$  or  $d_j = 1$ , and let  $f_j$  be the corresponding forecasted probabilities that  $d_j$  is 1,  $0 \leq f_j \leq 1$ . Assume that the data are ordered so that  $f_{j+1} \geq f_j$  (`brier` sorts the data to obtain this order). Divide the data into  $K$  nearly equally sized groups, with group 1 containing observations 1 through  $j_2 - 1$ , group 2 containing observations  $j_2$  through  $j_3 - 1$ , and so on.

Define

$$\bar{f}_0 = \text{average } f_j \text{ among } d_j = 0$$

$$\bar{f}_1 = \text{average } f_j \text{ among } d_j = 1$$

$$\bar{f} = \text{average } f_j$$

$$\bar{d} = \text{average } d_j$$

$$\tilde{f}_k = \text{average } f_j \text{ in group } k$$

$$\tilde{d}_k = \text{average } d_j \text{ in group } k$$

$$\tilde{n}_k = \text{number of observations in group } k$$

The Brier score is  $\sum_j (d_j - f_j)^2 / N$ .

The Sanders-modified Brier score is  $\sum_j (d_j - \tilde{f}_{k(j)})^2 / N$ .

Let  $p_j$  denote the true but unknown probability that  $d_j = 1$ . Under the null hypothesis that  $p_j = f_j$  for all  $j$ , [Spiegelhalter \(1986\)](#) determined that the expectation and variance of the Brier score is given by the following:

$$E(\text{Brier}) = \frac{1}{N} \sum_{j=1}^N f_j(1 - f_j)$$

$$\text{Var}(\text{Brier}) = \frac{1}{N^2} \sum_{j=1}^N f_j(1 - f_j)(1 - 2f_j)^2$$

Denoting the observed value of the Brier score by  $O(\text{Brier})$ , Spiegelhalter's  $z$  statistic is given by

$$Z = \frac{O(\text{Brier}) - E(\text{Brier})}{\sqrt{\text{Var}(\text{Brier})}}$$

The corresponding  $p$ -value is given by the upper-tail probability of  $Z$  under the standard normal distribution.

The area under the ROC curve is estimated by applying the trapezoidal rule to the empirical ROC curve. This area is Wilcoxon's test statistic, so the corresponding  $p$ -value is just that of a one-sided Wilcoxon test of the null hypothesis that the distribution of predictions is constant across the two outcomes.

The Sanders resolution is  $\sum_k \tilde{n}_k \{\tilde{d}_k(1 - \tilde{d}_k)\}/N$ .

The outcome index variance is  $\bar{d}(1 - \bar{d})$ .

The Murphy resolution is  $\sum_k \tilde{n}_k (\tilde{d}_k - \bar{d})^2/N$ .

Reliability-in-the-small is  $\sum_k \tilde{n}_k (\tilde{d}_k - \tilde{f}_k)^2/N$ .

The forecast variance is  $\sum_j (f_j - \bar{f})^2/N$ .

The minimum forecast variance is  $\{\sum_{j \in F} (f_j - \bar{f}_0)^2 + \sum_{j \in S} (f_j - \bar{f}_1)^2\}/N$ , where  $F$  is the set of observations for which  $d_j = 0$  and  $S$  is the complement.

The excess forecast variance is the difference between the forecast variance and the minimum forecast variance.

Reliability-in-the-large is  $(\bar{f} - \bar{d})^2$ .

Twice the outcome covariance is  $2(\bar{f}_1 - \bar{f}_0)\bar{d}(1 - \bar{d})$ .

Glenn Wilson Brier (1913–1998) was an American meteorological statistician who, after obtaining degrees in physics and statistics, was for many years head of meteorological statistics at the U.S. Weather Bureau, Washington, DC. In the latter part of his career, he was associated with Colorado State University. Brier worked especially on verification and evaluation of predictions and forecasts, statistical decision making, the statistical theory of turbulence, the analysis of weather modification experiments, and the application of permutation techniques.

## Acknowledgment

We thank Richard Goldstein for his contributions to this improved version of `brier`.

## References

- Brier, G. W. 1950. Verification of forecasts expressed in terms of probability. *Monthly Weather Review* 78: 1–3.
- Goldstein, R. 1996. [sg55: Extensions to the brier command](#). *Stata Technical Bulletin* 32: 21–22. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 133–134. College Station, TX: Stata Press.
- Hadorn, D. C., E. B. Keeler, W. H. Rogers, and R. H. Brook. 1993. *Assessing the Performance of Mortality Prediction Models*. Santa Monica, CA: Rand.
- Holloway, L., and P. Mielke. 1998. Glenn Wilson Brier 1913–1998. *Bulletin of the American Meteorological Society* 79: 1438–1439.
- Jolliffe, I. T., and D. B. Stephenson, ed. 2003. *Forecast Verification: A Practitioner’s Guide in Atmospheric Science*. Chichester, UK: Wiley.
- Murphy, A. H. 1973. A new vector partition of the probability score. *Journal of Applied Meteorology* 12: 595–600.
- . 1997. Forecast verification. In *Economic Value of Weather and Climate Forecasts*, ed. R. W. Katz and A. H. Murphy, 19–74. Cambridge: Cambridge University Press.
- Redelmeier, D. A., D. A. Bloch, and D. H. Hickam. 1991. Assessing predictive accuracy: How to compare Brier scores. *Journal of Clinical Epidemiology* 44: 1141–1146.
- Rogers, W. H. 1992. [sbe9: Brier score decomposition](#). *Stata Technical Bulletin* 10: 20–22. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 92–94. College Station, TX: Stata Press.
- Sanders, F. 1963. On subjective probability forecasting. *Journal of Applied Meteorology* 2: 191–201.
- Schmidt, C. H., and J. L. Griffith. 2005. Multivariate classification rules: Calibration and discrimination. In Vol. 2 of *Encyclopedia of Biostatistics*, ed. P. Armitage and T. Colton, 3492–3494. Chichester, UK: Wiley.
- Spiegelhalter, D. J. 1986. Probabilistic prediction in patient management and clinical trials. *Statistics in Medicine* 5: 421–433.
- Von Storch, H., and F. W. Zwiers. 1999. *Statistical Analysis in Climate Research*. Cambridge: Cambridge University Press.
- Wilks, D. S. 2006. *Statistical Methods in the Atmospheric Sciences*. 2nd ed. Burlington, MA: Academic Press.
- Yates, J. F. 1982. External correspondence: Decompositions of the mean probability score. *Organizational Behavior and Human Performance* 30: 132–156.

## Also see

- [R] [logistic](#) — Logistic regression, reporting odds ratios
- [R] [logit](#) — Logistic regression, reporting coefficients
- [R] [probit](#) — Probit regression

## Syntax

```
bsample [exp] [if] [in] [, options]
```

where *exp* is a standard Stata expression; see [\[U\] 13 Functions and expressions](#).

| <i>options</i>                 | description                                   |
|--------------------------------|-----------------------------------------------|
| <code>strata(varlist)</code>   | variables identifying strata                  |
| <code>cluster(varlist)</code>  | variables identifying resampling clusters     |
| <code>idcluster(newvar)</code> | create new cluster ID variable                |
| <code>weight(varname)</code>   | replace <i>varname</i> with frequency weights |

## Menu

Statistics > Resampling > Draw bootstrap sample

## Description

**bsample** draws random samples with replacement from the data in memory.

*exp* specifies the size of the sample, which must be less than or equal to the number of sampling units in the data. The observed number of units is the default when *exp* is not specified.

For a simple random sample (SRS) of the observations, *exp* must be less than or equal to `_N` (the number of observations in the data; see [\[U\] 13.4 System variables \(\\_variables\)](#)).

For stratified SRS, *exp* must be less than or equal to `_N` within the strata identified by the `strata()` option.

For clustered sampling, *exp* must be less than or equal to  $N_c$  (the number of clusters identified by the `cluster()` option).

For stratified sampling of clusters, *exp* must be less than or equal to  $N_c$  within the strata identified by the `strata()` option.

Observations that do not meet the optional `if` and `in` criteria are dropped (not sampled).

## Options

`strata(varlist)` specifies the variables identifying strata. If `strata()` is specified, bootstrap samples are selected within each stratum.

`cluster(varlist)` specifies the variables identifying resampling clusters. If `cluster()` is specified, the sample drawn during each replication is a bootstrap sample of clusters.

`idcluster(newvar)` creates a new variable containing a unique identifier for each resampled cluster.

`weight(varname)` specifies a variable in which the sampling frequencies will be placed. *varname* must be an existing variable, which will be replaced. After `bsample`, *varname* can be used as an `fweight` in any Stata command that accepts `fweights`, which can speed up resampling for commands like `regress` and `summarize`. This option cannot be combined with `idcluster()`.

By default, `bsample` replaces the data in memory with the sampled observations; however, specifying the `weight()` option causes only the specified *varname* to be changed.

## Remarks

Below is a series of examples illustrating how `bsample` is used with various sampling schemes.

### ➤ Example 1: Simple random sampling

We have data on the characteristics of hospital patients and wish to draw a simple random sample of 200 patients. We type

```
. use http://www.stata-press.com/data/r11/bsample1
. bsample 200
. count
200
```



### ➤ Example 2: Stratified samples with equal sizes

Among the variables in our dataset is `female`, an indicator for the female patients. To get a stratified simple random sample of 200 female patients and 200 male patients, we type

```
. use http://www.stata-press.com/data/r11/bsample1, clear
. bsample 200, strata(female)
. tab female
```

| female | Freq. | Percent | Cum.   |
|--------|-------|---------|--------|
| male   | 200   | 50.00   | 50.00  |
| female | 200   | 50.00   | 100.00 |
| Total  | 400   | 100.00  |        |



### ➤ Example 3: Stratified samples with unequal sizes

To sample 300 females and 200 males, we must generate a variable that is 300 for females and 200 for males and then use this variable in *exp* when we call `bsample`.

```
. use http://www.stata-press.com/data/r11/bsample1, clear
. gen nsamp = cond(female,300,200)
. bsample nsamp, strata(female)
. tab female
```

| female | Freq. | Percent | Cum.   |
|--------|-------|---------|--------|
| male   | 200   | 40.00   | 40.00  |
| female | 300   | 60.00   | 100.00 |
| Total  | 500   | 100.00  |        |





### ► Example 4: Samples satisfying a condition

For a simple random sample of 200 female patients, we type

```
. use http://www.stata-press.com/data/r11/bsample1, clear
. bsample 200 if female
. tab female
```

| female | Freq. | Percent | Cum.   |
|--------|-------|---------|--------|
| female | 200   | 100.00  | 100.00 |
| Total  | 200   | 100.00  |        |

◀

### ► Example 5: Generating frequency weights

To identify the sampled observations using frequency weights instead of dropping unsampled observations, we use the `weight()` option (we will need to supply it an existing variable name) and type

```
. use http://www.stata-press.com/data/r11/bsample1, clear
. set seed 1234
. gen fw = .
(5810 missing values generated)
. bsample 200 if female, weight(fw)
. tabulate fw female
```

| fw    | female |        | Total |
|-------|--------|--------|-------|
|       | male   | female |       |
| 0     | 2,392  | 3,221  | 5,613 |
| 1     | 0      | 194    | 194   |
| 2     | 0      | 3      | 3     |
| Total | 2,392  | 3,418  | 5,810 |

Note that  $(194 \times 1) + (3 \times 2) = 200$ .

◀

### ► Example 6: Oversampling observations

`bsample` requires the expression in *exp* to evaluate to a number that is less than or equal to the number of observations. To sample twice as many male and female patients as there are already in memory, we must expand the data before using `bsample`. For example,

```
. use http://www.stata-press.com/data/r11/bsample1, clear
. set seed 1234
. expand 2
(5810 observations created)
. bsample, strata(female)
```

```
. tab female
```

| female | Freq.  | Percent | Cum.   |
|--------|--------|---------|--------|
| male   | 4,784  | 41.17   | 41.17  |
| female | 6,836  | 58.83   | 100.00 |
| Total  | 11,620 | 100.00  |        |

➤ Example 7: Stratified oversampling with unequal sizes

To sample twice as many female patients as male patients, we must expand the records for the female patients because there are less than twice as many of them as there are male patients, but first put the number of observed male patients in a local macro. After expanding the female records, we generate a variable that contains the number of observations to sample within the two groups.

```
. use http://www.stata-press.com/data/r11/bsample1, clear
. set seed 1234
. count if !female
2392
. local nmale = r(N)
. expand 2 if female
(3418 observations created)
. gen nsamp = cond(female,2*`nmale`,`nmale`)
. bsample nsamp, strata(female)
. tab female
```

| female | Freq. | Percent | Cum.   |
|--------|-------|---------|--------|
| male   | 2,392 | 33.33   | 33.33  |
| female | 4,784 | 66.67   | 100.00 |
| Total  | 7,176 | 100.00  |        |

➤ Example 8: Oversampling of clusters

For clustered data, sampling more clusters than are present in the original dataset requires more than just expanding the data. To illustrate, suppose we wanted a bootstrap sample of eight clusters from a dataset consisting of five clusters of observations.

```
. use http://www.stata-press.com/data/r11/bsample2, clear
. tabstat x, stat(n mean) by(group)
Summary for variables: x
by categories of: group
```

| group | N  | mean      |
|-------|----|-----------|
| A     | 15 | -.3073028 |
| B     | 10 | -.00984   |
| C     | 11 | .0810985  |
| D     | 11 | -.1989179 |
| E     | 29 | -.095203  |
| Total | 76 | -.1153269 |

`bsample` will complain if we simply expand the dataset.

```
. use http://www.stata-press.com/data/r11/bsample2, clear
. expand 3
(152 observations created)
. bsample 8, cluster(group)
resample size must not be greater than number of clusters
r(498);
```

Expanding the data will only partly solve the problem. We also need a new variable that uniquely identifies the copied clusters. We use the `expandcl` command to accomplish both these tasks; see [D] [expandcl](#).

```
. use http://www.stata-press.com/data/r11/bsample2, clear
. set seed 1234
. expandcl 2, generate(expgroup) cluster(group)
(76 observations created)
. tabstat x, stat(n mean) by(expgroup)
```

Summary for variables: x  
by categories of: expgroup

| expgroup | N   | mean      |
|----------|-----|-----------|
| 1        | 15  | -.3073028 |
| 2        | 15  | -.3073028 |
| 3        | 10  | -.00984   |
| 4        | 10  | -.00984   |
| 5        | 11  | .0810985  |
| 6        | 11  | .0810985  |
| 7        | 11  | -.1989179 |
| 8        | 11  | -.1989179 |
| 9        | 29  | -.095203  |
| 10       | 29  | -.095203  |
| Total    | 152 | -.1153269 |

```
. gen fw = .
(152 missing values generated)
. bsample 8, cluster(expgroup) weight(fw)
. tabulate fw group
```

| fw    | group |    |    |    |    | Total |
|-------|-------|----|----|----|----|-------|
|       | A     | B  | C  | D  | E  |       |
| 0     | 15    | 10 | 0  | 0  | 29 | 54    |
| 1     | 15    | 10 | 22 | 22 | 0  | 69    |
| 2     | 0     | 0  | 0  | 0  | 29 | 29    |
| Total | 30    | 20 | 22 | 22 | 58 | 152   |

The results from `tabulate` on the generated frequency weight variable versus the original cluster ID (`group`) show us that the bootstrap sample contains one copy of cluster A, one copy of cluster B, two copies of cluster C, two copies of cluster D, and two copies of cluster E ( $1 + 1 + 2 + 2 + 2 = 8$ ).

➤ Example 9: Stratified oversampling of clusters

Suppose that we have a dataset containing two strata with five clusters in each stratum, but the cluster identifiers are not unique between the strata. To get a stratified bootstrap sample with eight clusters in each stratum, we first use `expandcl` to expand the data and get a new cluster ID variable. We use `cluster(strid group)` in the call to `expandcl`; this action will uniquely identify the  $2 \times 5 = 10$  clusters across the strata.

```
. use http://www.stata-press.com/data/r11/bsample2, clear
. set seed 1234
. tab group strid
```

| group | strid |    | Total |
|-------|-------|----|-------|
|       | 1     | 2  |       |
| A     | 7     | 8  | 15    |
| B     | 5     | 5  | 10    |
| C     | 5     | 6  | 11    |
| D     | 5     | 6  | 11    |
| E     | 14    | 15 | 29    |
| Total | 36    | 40 | 76    |

```
. expandcl 2, generate(expgroup) cluster(strid group)
(76 observations created)
```

Now we can use `bsample` with the expanded data, stratum ID variable, and new cluster ID variable.

```
. gen fw = .
(152 missing values generated)
. bsample 8, cluster(expgroup) str(strid) weight(fw)
. by strid, sort: tabulate fw group
```

-> strid = 1

| fw    | group |    |    |    |    | Total |
|-------|-------|----|----|----|----|-------|
|       | A     | B  | C  | D  | E  |       |
| 0     | 0     | 5  | 0  | 5  | 14 | 24    |
| 1     | 14    | 5  | 10 | 5  | 0  | 34    |
| 2     | 0     | 0  | 0  | 0  | 14 | 14    |
| Total | 14    | 10 | 10 | 10 | 28 | 72    |

-> strid = 2

| fw    | group |    |    |    |    | Total |
|-------|-------|----|----|----|----|-------|
|       | A     | B  | C  | D  | E  |       |
| 0     | 8     | 10 | 0  | 6  | 0  | 24    |
| 1     | 8     | 0  | 6  | 6  | 15 | 35    |
| 2     | 0     | 0  | 6  | 0  | 15 | 21    |
| Total | 16    | 10 | 12 | 12 | 30 | 80    |

The results from `by strid: tabulate` on the generated frequency weight variable versus the original cluster ID (`group`) show us how many times each cluster was sampled for each stratum. For stratum 1, the bootstrap sample contains two copies of cluster A, one copy of cluster B, two copies of cluster C, one copy of cluster D, and two copies of cluster E ( $2 + 1 + 2 + 1 + 2 = 8$ ). For stratum 2, the bootstrap sample contains one copy of cluster A, zero copies of cluster B, three copies of cluster C, one copy of cluster D, and three copies of cluster E ( $1 + 0 + 3 + 1 + 3 = 8$ ).



## Methods and formulas

`bsample` is implemented as an ado-file.

## Also see

[R] [bootstrap](#) — Bootstrap sampling and estimation

[R] [bstat](#) — Report bootstrap results

[R] [simulate](#) — Monte Carlo simulations

[D] [sample](#) — Draw random sample

Syntax

*Bootstrap statistics from variables*

```
bstat [varlist] [if] [in] [, options]
```

*Bootstrap statistics from file*

```
bstat [namelist] [using filename] [if] [in] [, options]
```

| options                    | description                                               |
|----------------------------|-----------------------------------------------------------|
| Main                       |                                                           |
| <code>stat(vector)</code>  | observed values for each statistic                        |
| <code>accel(vector)</code> | acceleration values for each statistic                    |
| <code>mse</code>           | use MSE formula for variance estimation                   |
| Reporting                  |                                                           |
| <code>level(#)</code>      | set confidence level; default is <code>level(95)</code>   |
| <code>n(#)</code>          | # of observations from which bootstrap samples were taken |
| <code>notable</code>       | suppress table of results                                 |
| <code>noheader</code>      | suppress table header                                     |
| <code>nolegend</code>      | suppress table legend                                     |
| <code>verbose</code>       | display the full table legend                             |
| <code>title(text)</code>   | use <i>text</i> as title for bootstrap results            |

See [\[U\] 20 Estimation and postestimation commands](#) for more capabilities of estimation commands.

Menu

Statistics > Resampling > Report bootstrap results

Description

`bstat` is a programmer’s command that computes and displays estimation results from bootstrap statistics.

For each variable in *varlist* (the default is all variables), then `bstat` computes a covariance matrix, estimates bias, and constructs several different confidence intervals (CIs). The following CIs are constructed by `bstat`:

1. Normal CIs (using the normal approximation)
2. Percentile CIs
3. Bias-corrected (BC) CIs
4. Bias-corrected and accelerated (BC<sub>a</sub>) CIs (optional)

`estat bootstrap` displays a table of one or more of the above confidence intervals; see [\[R\] bootstrap postestimation](#).

If there are bootstrap estimation results in `e()`, `bstat` replays them. If given the `using` modifier, `bstat` uses the data in *filename* to compute the bootstrap statistics while preserving the data currently in memory. Otherwise, `bstat` uses the data in memory to compute the bootstrap statistics.

The following options may be used to replay estimation results from `bstat`:

`level(#)` `notable` `noheader` `nolegend` `verbose` `title(text)`

For all other options and the qualifiers `using`, `if`, and `in`, `bstat` requires a bootstrap dataset.

## Options

### Main

`stat(vector)` specifies the observed value of each statistic (i.e., the value of the statistic using the original dataset).

`accel(vector)` specifies the acceleration of each statistic, which is used to construct  $BC_a$  CIs.

`mse` specifies that `bstat` compute the variance by using deviations of the replicates from the observed value of the statistics. By default, `bstat` computes the variance by using deviations from the average of the replicates.

### Reporting

`level(#)`; see [\[R\] estimation options](#).

`n(#)` specifies the number of observations from which bootstrap samples were taken. This value is used in no calculations but improves the table header when this information is not saved in the bootstrap dataset.

`notable` suppresses the display of the output table.

`noheader` suppresses the display of the table header. This option implies `nolegend`.

`nolegend` suppresses the display of the table legend.

`verbose` specifies that the full table legend be displayed. By default, coefficients and standard errors are not displayed.

`title(text)` specifies a title to be displayed above the table of bootstrap results; the default title is `Bootstrap results`.

## Remarks

Remarks are presented under the following headings:

[Bootstrap datasets](#)

[Creating a bootstrap dataset](#)

## Bootstrap datasets

Although `bstat` allows you to specify the observed value and acceleration of each bootstrap statistic via the `stat()` and `accel()` options, programmers may be interested in what `bstat` uses when these options are not supplied.

When working from a bootstrap dataset, `bstat` first checks the data characteristics (see [P] [char](#)) that it understands:

`_dta[bs_version]` identifies the version of the bootstrap dataset. This characteristic may be empty (not defined), 2, or 3; otherwise, `bstat` will quit and display an error message. This version tells `bstat` which other characteristics to look for in the bootstrap dataset.

`bstat` uses the following characteristics from version 3 bootstrap datasets:

```
_dta[N]
_dta[N_strata]
_dta[N_cluster]
_dta[command]
varname[observed]
varname[acceleration]
varname[expression]
```

`bstat` uses the following characteristics from version 2 bootstrap datasets:

```
_dta[N]
_dta[N_strata]
_dta[N_cluster]
varname[observed]
varname[acceleration]
```

An empty bootstrap dataset version implies that the dataset was created by the `bstrap` command in a version of Stata earlier than Stata 8. Here `bstat` expects `varname[bstrap]` to contain the observed value of the statistic identified by `varname` (`varname[observed]` in version 2). All other characteristics are ignored.

`_dta[N]` is the number of observations in the observed dataset. This characteristic may be overruled by specifying the `n()` option.

`_dta[N_strata]` is the number of strata in the observed dataset.

`_dta[N_cluster]` is the number of clusters in the observed dataset.

`_dta[command]` is the command used to compute the observed values of the statistics.

`varname[observed]` is the observed value of the statistic identified by `varname`. To specify a different value, use the `stat()` option.

`varname[acceleration]` is the estimate of acceleration for the statistic identified by `varname`. To specify a different value, use the `accel()` option.

`varname[expression]` is the expression or label that describes the statistic identified by `varname`.

## Creating a bootstrap dataset

Suppose that we are interested in obtaining bootstrap statistics by resampling the residuals from a regression (which is not possible with the `bootstrap` command). After loading some data, we run a regression, save some results relevant to the `bstat` command, and save the residuals in a new variable, `res`.



```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
```

```
. regress mpg weight length
```

| Source   | SS         | df | MS         |  |  |  |
|----------|------------|----|------------|--|--|--|
| Model    | 1616.08062 | 2  | 808.040312 |  |  |  |
| Residual | 827.378835 | 71 | 11.653223  |  |  |  |
| Total    | 2443.45946 | 73 | 33.4720474 |  |  |  |

|  |  |  |  |                 |        |  |
|--|--|--|--|-----------------|--------|--|
|  |  |  |  | Number of obs = | 74     |  |
|  |  |  |  | F( 2, 71) =     | 69.34  |  |
|  |  |  |  | Prob > F =      | 0.0000 |  |
|  |  |  |  | R-squared =     | 0.6614 |  |
|  |  |  |  | Adj R-squared = | 0.6519 |  |
|  |  |  |  | Root MSE =      | 3.4137 |  |

| mpg    | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|--------|-----------|-----------|-------|-------|----------------------|-----------|
| weight | -.0038515 | .001586   | -2.43 | 0.018 | -.0070138            | -.0006891 |
| length | -.0795935 | .0553577  | -1.44 | 0.155 | -.1899736            | .0307867  |
| _cons  | 47.88487  | 6.08787   | 7.87  | 0.000 | 35.746               | 60.02374  |

```
. matrix b = e(b)
. local n = e(N)
. predict res, residuals
```

We can resample the residual values in `res` by generating a random observation ID (`rid`), generate a new response variable (`y`), and run the original regression with the new response variables.

```
. set seed 54321
. gen rid = int(_N*runiform())+1
. matrix score double y = b
. replace y = y + res[rid]
(74 real changes made)
. regress y weight length
```

| Source   | SS         | df | MS         |  |  |  |
|----------|------------|----|------------|--|--|--|
| Model    | 1773.23548 | 2  | 886.617741 |  |  |  |
| Residual | 608.747732 | 71 | 8.57391172 |  |  |  |
| Total    | 2381.98321 | 73 | 32.629907  |  |  |  |

|  |  |  |  |                 |        |  |
|--|--|--|--|-----------------|--------|--|
|  |  |  |  | Number of obs = | 74     |  |
|  |  |  |  | F( 2, 71) =     | 103.41 |  |
|  |  |  |  | Prob > F =      | 0.0000 |  |
|  |  |  |  | R-squared =     | 0.7444 |  |
|  |  |  |  | Adj R-squared = | 0.7372 |  |
|  |  |  |  | Root MSE =      | 2.9281 |  |

| y      | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|--------|-----------|-----------|-------|-------|----------------------|-----------|
| weight | -.0059938 | .0013604  | -4.41 | 0.000 | -.0087064            | -.0032813 |
| length | -.0127875 | .0474837  | -0.27 | 0.788 | -.1074673            | .0818924  |
| _cons  | 42.23195  | 5.22194   | 8.09  | 0.000 | 31.8197              | 52.6442   |

Instead of programming this resampling inside a loop, it is much more convenient to write a short program and use the `simulate` command; see [\[R\] simulate](#). In the following, `mysim_r` requires the user to specify a coefficient vector and a residual variable. `mysim_r` then retrieves the list of predictor variables (removing `_cons` from the list), generates a new temporary response variable with the resampled residuals, and regresses the new response variable on the predictors.

```
program mysim_r
 version 11
 syntax name(name=bvector), res(varname)
 tempvar y rid
 local xvars : colnames 'bvector'
 local cons _cons
 local xvars : list xvars - cons
 matrix score double 'y' = 'bvector'
 gen long 'rid' = int(_N*runiform()) + 1
 replace 'y' = 'y' + 'res'[_rid]
 regress 'y' 'xvars'
end
```

We can now give `mysim_r` a test run, but we first set the random-number seed (to reproduce results).

```
. set seed 54321
. mysim_r b, res(res)
(74 real changes made)
```

| Source   | SS         | df | MS         |  | Number of obs = | 74     |
|----------|------------|----|------------|--|-----------------|--------|
| Model    | 1773.23548 | 2  | 886.617741 |  | F( 2, 71) =     | 103.41 |
| Residual | 608.747732 | 71 | 8.57391172 |  | Prob > F =      | 0.0000 |
|          |            |    |            |  | R-squared =     | 0.7444 |
|          |            |    |            |  | Adj R-squared = | 0.7372 |
| Total    | 2381.98321 | 73 | 32.629907  |  | Root MSE =      | 2.9281 |

| __000000 | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| weight   | -.0059938 | .0013604  | -4.41 | 0.000 | -.0087064            | -.0032813 |
| length   | -.0127875 | .0474837  | -0.27 | 0.788 | -.1074673            | .0818924  |
| _cons    | 42.23195  | 5.22194   | 8.09  | 0.000 | 31.8197              | 52.6442   |

Now that we have a program that will compute the results we want, we can use `simulate` to generate a bootstrap dataset and `bstat` to display the results.

```
. set seed 54321
. simulate, reps(200) nodots: mysim_r b, res(res)
 command: mysim_r b, res(res)

. bstat, stat(b) n('n')
```

Bootstrap results

|  |  |  |  |  |               |   |     |
|--|--|--|--|--|---------------|---|-----|
|  |  |  |  |  | Number of obs | = | 74  |
|  |  |  |  |  | Replications  | = | 200 |

|           | Observed<br>Coef. | Bootstrap<br>Std. Err. | z     | P> z  | Normal-based<br>[95% Conf. Interval] |           |
|-----------|-------------------|------------------------|-------|-------|--------------------------------------|-----------|
| _b_weight | -.0038515         | .0015715               | -2.45 | 0.014 | -.0069316                            | -.0007713 |
| _b_length | -.0795935         | .0552415               | -1.44 | 0.150 | -.1878649                            | .0286779  |
| _b_cons   | 47.88487          | 6.150069               | 7.79  | 0.000 | 35.83096                             | 59.93879  |

Finally, we see that `simulate` created some of the data characteristics recognized by `bstat`. All we need to do is correctly specify the version of the bootstrap dataset, and `bstat` will automatically use the relevant data characteristics.

```
. char list
_dta[seed]: X681014b5c43f462544a474abacbdd93d12a1
_dta[command]: mysim_r b, res(res)
_b_weight[is_eexp]: 1
_b_weight[colname]: weight
_b_weight[coleq]: -
_b_weight[expression]: _b[weight]
_b_length[is_eexp]: 1
_b_length[colname]: length
_b_length[coleq]: -
_b_length[expression]: _b[length]
_b_cons[is_eexp]: 1
_b_cons[colname]: _cons
_b_cons[coleq]: -
_b_cons[expression]: _b[_cons]

. char _dta[bs_version] 3

. bstat, stat(b) n('n')

Bootstrap results Number of obs = 74
 Replications = 200

command: mysim_r b, res(res)
```

|        | Observed<br>Coef. | Bootstrap<br>Std. Err. | z     | P> z  | Normal-based<br>[95% Conf. Interval] |           |
|--------|-------------------|------------------------|-------|-------|--------------------------------------|-----------|
| weight | -.0038515         | .0015715               | -2.45 | 0.014 | -.0069316                            | -.0007713 |
| length | -.0795935         | .0552415               | -1.44 | 0.150 | -.1878649                            | .0286779  |
| _cons  | 47.88487          | 6.150069               | 7.79  | 0.000 | 35.83096                             | 59.93879  |

See [Poi \(2004\)](#) for another example of residual resampling.

(Continued on next page)

## Saved results

`bstat` saves the following in `e()`:

### Scalars

|                            |                                                                           |
|----------------------------|---------------------------------------------------------------------------|
| <code>e(N)</code>          | sample size                                                               |
| <code>e(N_reps)</code>     | number of complete replications                                           |
| <code>e(N_misreps)</code>  | number of incomplete replications                                         |
| <code>e(N_strata)</code>   | number of strata                                                          |
| <code>e(N_clust)</code>    | number of clusters                                                        |
| <code>e(k_aux)</code>      | number of auxiliary parameters                                            |
| <code>e(k_eq)</code>       | number of equations                                                       |
| <code>e(k_exp)</code>      | number of standard expressions                                            |
| <code>e(k_eeexp)</code>    | number of extended expressions (i.e., <code>_b</code> )                   |
| <code>e(k_extra)</code>    | number of extra equations beyond the original ones from <code>e(b)</code> |
| <code>e(level)</code>      | confidence level for bootstrap CIs                                        |
| <code>e(bs_version)</code> | version for bootstrap results                                             |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                                                 |

### Macros

|                            |                                                |
|----------------------------|------------------------------------------------|
| <code>e(cmd)</code>        | <code>bstat</code>                             |
| <code>e(command)</code>    | from <code>_dta[command]</code>                |
| <code>e(cmdline)</code>    | command as typed                               |
| <code>e(title)</code>      | title in estimation output                     |
| <code>e(exp#)</code>       | expression for the <code>#</code> th statistic |
| <code>e(prefix)</code>     | <code>bootstrap</code>                         |
| <code>e(mse)</code>        | <code>mse</code> if specified                  |
| <code>e(vce)</code>        | <code>bootstrap</code>                         |
| <code>e(vcetype)</code>    | title used to label Std. Err.                  |
| <code>e(properties)</code> | <code>b V</code>                               |

### Matrices

|                               |                                      |
|-------------------------------|--------------------------------------|
| <code>e(b)</code>             | observed statistics                  |
| <code>e(b_bs)</code>          | bootstrap estimates                  |
| <code>e(reps)</code>          | number of nonmissing results         |
| <code>e(bias)</code>          | estimated biases                     |
| <code>e(se)</code>            | estimated standard errors            |
| <code>e(z0)</code>            | median biases                        |
| <code>e(accel)</code>         | estimated accelerations              |
| <code>e(ci_normal)</code>     | normal-approximation CIs             |
| <code>e(ci_percentile)</code> | percentile CIs                       |
| <code>e(ci_bc)</code>         | bias-corrected CIs                   |
| <code>e(ci_bca)</code>        | bias-corrected and accelerated CIs   |
| <code>e(V)</code>             | bootstrap variance–covariance matrix |

## Methods and formulas

`bstat` is implemented as an ado-file.

---

## Reference

Poi, B. P. 2004. [From the help desk: Some bootstrapping techniques](#). *Stata Journal* 4: 312–328.

## Also see

[R] [bootstrap](#) — Bootstrap sampling and estimation

[R] [bsample](#) — Sampling with replacement

# Title

centile — Report centile and confidence interval

## Syntax

`centile` [*varlist*] [*if*] [*in*] [, *options*]

| <i>options</i>                              | description                                                    |
|---------------------------------------------|----------------------------------------------------------------|
| Main                                        |                                                                |
| <code><u>centile</u>(<i>numlist</i>)</code> | report specified centiles; default is <code>centile(50)</code> |
| Options                                     |                                                                |
| <code><u>cci</u></code>                     | binomial exact; conservative confidence interval               |
| <code><u>normal</u></code>                  | normal, based on observed centiles                             |
| <code><u>meansd</u></code>                  | normal, based on mean and standard deviation                   |
| <code><u>level</u>(#)</code>                | set confidence level; default is <code>level(95)</code>        |

`by` is allowed; see [\[D\]](#) `by`.

## Menu

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Centiles with CIs

## Description

`centile` estimates specified centiles and calculates confidence intervals. If no *varlist* is specified, `centile` calculates centiles for all the variables in the dataset. If `centile()` is not specified, medians (`centile(50)`) are reported.

## Options

### Main

`centile(numlist)` specifies the centiles to be reported. The default is to display the 50th centile. Specifying `centile(5)` requests that the fifth centile be reported. Specifying `centile(5 50 95)` requests that the 5th, 50th, and 95th centiles be reported. Specifying `centile(10(10)90)` requests that the 10th, 20th, . . . , 90th centiles be reported; see [\[U\]](#) [11.1.8 numlist](#).

### Options

`cci` (conservative confidence interval) forces the confidence limits to fall exactly on sample values. Confidence intervals displayed with the `cci` option are slightly wider than those with the default (`nocci`) option.

`normal` causes the confidence interval to be calculated by using a formula for the standard error of a normal-distribution quantile given by [Kendall and Stuart \(1969, 237\)](#). The `normal` option is useful when you want empirical centiles—that is, centiles based on sample order statistics rather than on the mean and standard deviation—and are willing to assume normality.

`meansd` causes the centile and confidence interval to be calculated based on the sample mean and standard deviation, and it assumes normality.

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see [R] [level](#).

## Remarks

The  $q$ th centile of a continuous random variable,  $X$ , is defined as the value of  $C_q$ , which fulfills the condition  $\Pr(X \leq C_q) = q/100$ . The value of  $q$  must be in the range  $0 < q < 100$ , though  $q$  is not necessarily an integer. By default, `centile` estimates  $C_q$  for the variables in `varlist` and for the values of  $q$  given in `centile(numlist)`. It makes no assumptions about the distribution of  $X$ , and, if necessary, uses linear interpolation between neighboring sample values. Extreme centiles (for example, the 99th centile in samples smaller than 100) are fixed at the minimum or maximum sample value. An “exact” confidence interval for  $C_q$  is also given, using the binomial-based method described below in [Methods and formulas](#) and in [Conover \(1999, 143–148\)](#). Again linear interpolation is used to improve the accuracy of the estimated confidence limits, but extremes are fixed at the minimum or maximum sample value.

You can prevent `centile` from interpolating when calculating binomial-based confidence intervals by specifying `cci`. The resulting intervals are generally wider than with the default; that is, the coverage (confidence level) tends to be greater than the nominal value (given as usual by `level(#)`, by default 95%).

If the data are believed to be normally distributed (a common case), there are two alternative methods for estimating centiles. If `normal` is specified,  $C_q$  is calculated, as just described, but its confidence interval is based on a formula for the standard error (se) of a normal-distribution quantile given by [Kendall and Stuart \(1969, 237\)](#). If `meansd` is alternatively specified,  $C_q$  is estimated as  $\bar{x} + z_q \times s$ , where  $\bar{x}$  and  $s$  are the sample mean and standard deviation, and  $z_q$  is the  $q$ th centile of the standard normal distribution (e.g.,  $z_{95} = 1.645$ ). The confidence interval is derived from the se of the estimate of  $C_q$ .

## ► Example 1

Using `auto.dta`, we estimate the 5th, 50th, and 95th centiles of the price variable:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. format price %8.2fc
. centile price, centile(5 50 95)
```

| Variable | Obs | Percentile | Centile   | — Binom. Interp. —<br>[95% Conf. Interval] |           |
|----------|-----|------------|-----------|--------------------------------------------|-----------|
| price    | 74  | 5          | 3,727.75  | 3,291.23                                   | 3,914.16  |
|          |     | 50         | 5,006.50  | 4,593.57                                   | 5,717.90  |
|          |     | 95         | 13,498.00 | 11,061.53                                  | 15,865.30 |

`summarize` produces somewhat different results from `centile`; see [Methods and formulas](#).

```
. summarize price, detail
```

| Price       |        |          |             |          |
|-------------|--------|----------|-------------|----------|
| Percentiles |        | Smallest |             |          |
| 1%          | 3291   | 3291     |             |          |
| 5%          | 3748   | 3299     |             |          |
| 10%         | 3895   | 3667     | Obs         | 74       |
| 25%         | 4195   | 3748     | Sum of Wgt. | 74       |
| 50%         | 5006.5 |          | Mean        | 6165.257 |
|             |        | Largest  | Std. Dev.   | 2949.496 |
| 75%         | 6342   | 13466    |             |          |
| 90%         | 11385  | 13594    | Variance    | 8699526  |
| 95%         | 13466  | 14500    | Skewness    | 1.653434 |
| 99%         | 15906  | 15906    | Kurtosis    | 4.819188 |

The confidence limits produced by using the `cci` option are slightly wider than those produced without this option:

```
. centile price, c(5 50 95) cci
```

| Variable | Obs | Percentile | Centile   | — Binomial Exact —<br>[95% Conf. Interval] |           |
|----------|-----|------------|-----------|--------------------------------------------|-----------|
| price    | 74  | 5          | 3,727.75  | 3,291.00                                   | 3,955.00  |
|          |     | 50         | 5,006.50  | 4,589.00                                   | 5,719.00  |
|          |     | 95         | 13,498.00 | 10,372.00                                  | 15,906.00 |

If we are willing to assume that `price` is normally distributed, we could include either the `normal` or the `meansd` option:

```
. centile price, c(5 50 95) normal
```

| Variable | Obs | Percentile | Centile   | — Normal, based on observed centiles —<br>[95% Conf. Interval] |           |
|----------|-----|------------|-----------|----------------------------------------------------------------|-----------|
| price    | 74  | 5          | 3,727.75  | 3,211.19                                                       | 4,244.31  |
|          |     | 50         | 5,006.50  | 4,096.68                                                       | 5,916.32  |
|          |     | 95         | 13,498.00 | 5,426.81                                                       | 21,569.19 |

```
. centile price, c(5 50 95) meansd
```

| Variable | Obs | Percentile | Centile   | — Normal, based on mean and std. dev. —<br>[95% Conf. Interval] |           |
|----------|-----|------------|-----------|-----------------------------------------------------------------|-----------|
| price    | 74  | 5          | 1,313.77  | 278.93                                                          | 2,348.61  |
|          |     | 50         | 6,165.26  | 5,493.24                                                        | 6,837.27  |
|          |     | 95         | 11,016.75 | 9,981.90                                                        | 12,051.59 |

With the `normal` option, the centile estimates are, by definition, the same as before. The confidence intervals for the 5th and 50th centiles are similar to the previous ones, but the interval for the 95th centile is different. The results using the `meansd` option also differ from both previous sets of estimates.

We can use `sktest` (see [\[R\] sktest](#)) to check the correctness of the normality assumption:

```
. sktest price
```

| Skewness/Kurtosis tests for Normality |     |              |              |             |                 |
|---------------------------------------|-----|--------------|--------------|-------------|-----------------|
| Variable                              | Obs | Pr(Skewness) | Pr(Kurtosis) | adj chi2(2) | joint Prob>chi2 |
| price                                 | 74  | 0.0000       | 0.0127       | 21.77       | 0.0000          |



`sktest` reveals that `price` is definitely not normally distributed, so the normal assumption is not reasonable, and the `normal` and `meansd` options are not appropriate for these data. We should rely on the results from the default choice, which does not assume normality. If the data are normally distributed, however, the precision of the estimated centiles and their confidence intervals will be ordered (best) `meansd` > `normal` > [default] (worst). The `normal` option is useful when we really do want empirical centiles (that is, centiles based on sample order statistics rather than on the mean and standard deviation) but are willing to assume normality.

◀

## Saved results

`centile` saves the following in `r()`:

### Scalars

|                        |                              |                      |                                            |
|------------------------|------------------------------|----------------------|--------------------------------------------|
| <code>r(N)</code>      | number of observations       | <code>r(lb_#)</code> | #-requested centile lower confidence bound |
| <code>r(n_cent)</code> | number of centiles requested | <code>r(ub_#)</code> | #-requested centile upper confidence bound |
| <code>r(c_#)</code>    | value of # centile           |                      |                                            |

### Macros

`r(centiles)` centiles requested

## Methods and formulas

`centile` is implemented as an ado-file.

Methods and formulas are presented under the following headings:

*Default case*

*Normal case*

*meansd case*

## Default case

The calculation is based on the method of [Mood and Graybill \(1963, 408\)](#). Let  $x_1 \leq x_2 \leq \dots \leq x_n$  be a sample of size  $n$  arranged in ascending order. Denote the estimated  $q$ th centile of the  $x$ 's as  $c_q$ . We require that  $0 < q < 100$ . Let  $R = (n+1)q/100$  have integer part  $r$  and fractional part  $f$ ; that is,  $r = \text{int}(R)$  and  $f = R - r$ . (If  $R$  is itself an integer, then  $r = R$  and  $f = 0$ .) Note that  $0 \leq r \leq n$ . For convenience, define  $x_0 = x_1$  and  $x_{n+1} = x_n$ .  $C_q$  is estimated by

$$c_q = x_r + f \times (x_{r+1} - x_r)$$

that is,  $c_q$  is a weighted average of  $x_r$  and  $x_{r+1}$ . Loosely speaking, a (conservative)  $p\%$  confidence interval for  $C_q$  involves finding the observations ranked  $t$  and  $u$ , which correspond, respectively, to the  $\alpha = (100 - p)/200$  and  $1 - \alpha$  quantiles of a binomial distribution with parameters  $n$  and  $q/100$ , i.e.,  $B(n, q/100)$ . More precisely, define the  $i$ th value ( $i = 0, \dots, n$ ) of the cumulative binomial distribution function as  $F_i = \Pr(S \leq i)$ , where  $S$  has distribution  $B(n, q/100)$ . For convenience, let  $F_{-1} = 0$  and  $F_{n+1} = 1$ .  $t$  is found such that  $F_t \leq \alpha$  and  $F_{t+1} > \alpha$ , and  $u$  is found such that  $1 - F_u \leq \alpha$  and  $1 - F_{u-1} > \alpha$ .

With the `cci` option in force, the (conservative) confidence interval is  $(x_{t+1}, x_{u+1})$ , and its actual coverage probability is  $F_u - F_t$ .

The default case uses linear interpolation on the  $F_i$  as follows. Let

$$\begin{aligned} g &= (\alpha - F_t)/(F_{t+1} - F_t) \\ h &= \{\alpha - (1 - F_u)\}/\{(1 - F_{u+1}) - (1 - F_u)\} \\ &= (\alpha - 1 + F_u)/(F_u - F_{u+1}) \end{aligned}$$

The interpolated lower and upper confidence limits ( $c_{qL}, c_{qU}$ ) for  $C_q$  are

$$\begin{aligned} c_{qL} &= x_{t+1} + g \times (x_{t+2} - x_{t+1}) \\ c_{qU} &= x_{u+1} - h \times (x_{u+1} - x_u) \end{aligned}$$

Suppose that we want a 95% confidence interval for the median of a sample of size 13.  $n = 13$ ,  $q = 50$ ,  $p = 95$ ,  $\alpha = 0.025$ ,  $R = 14 \times 50/100 = 7$ , and  $f = 0$ . Therefore, the median is the 7th observation. Some example data,  $x_i$ , and the values of  $F_i$  are as follows:

| $i$ | $F_i$  | $1 - F_i$ | $x_i$ | $i$ | $F_i$  | $1 - F_i$ | $x_i$ |
|-----|--------|-----------|-------|-----|--------|-----------|-------|
| 0   | 0.0001 | 0.9999    | —     | 7   | 0.7095 | 0.2905    | 33    |
| 1   | 0.0017 | 0.9983    | 5     | 8   | 0.8666 | 0.1334    | 37    |
| 2   | 0.0112 | 0.9888    | 7     | 9   | 0.9539 | 0.0461    | 45    |
| 3   | 0.0461 | 0.9539    | 10    | 10  | 0.9888 | 0.0112    | 59    |
| 4   | 0.1334 | 0.8666    | 15    | 11  | 0.9983 | 0.0017    | 77    |
| 5   | 0.2905 | 0.7095    | 23    | 12  | 0.9999 | 0.0001    | 104   |
| 6   | 0.5000 | 0.5000    | 28    | 13  | 1.0000 | 0.0000    | 211   |

The median is  $x_7 = 33$ . Also,  $F_2 \leq 0.025$  and  $F_3 > 0.025$ , so  $t = 2$ ;  $1 - F_{10} \leq 0.025$  and  $1 - F_9 > 0.025$ , so  $u = 10$ . The conservative confidence interval is therefore

$$(c_{50L}, c_{50U}) = (x_{t+1}, x_{u+1}) = (x_3, x_{11}) = (10, 77)$$

with actual coverage  $F_{10} - F_2 = 0.9888 - 0.0112 = 0.9776$  (97.8% confidence). For the interpolation calculation, we have

$$\begin{aligned} g &= (0.025 - 0.0112)/(0.0461 - 0.0112) = 0.395 \\ h &= (0.025 - 1 + 0.9888)/(0.9888 - 0.9539) = 0.395 \end{aligned}$$

So,

$$\begin{aligned} c_{50L} &= x_3 + 0.395 \times (x_4 - x_3) = 10 + 0.395 \times 5 = 11.98 \\ c_{50U} &= x_{11} - 0.395 \times (x_{11} - x_{10}) = 77 - 0.395 \times 18 = 69.89 \end{aligned}$$

### Normal case

The value of  $c_q$  is as above. Its se is given by the formula

$$s_q = \sqrt{q(100 - q)} \Big/ \left\{ 100\sqrt{n}Z(c_q; \bar{x}, s) \right\}$$

where  $\bar{x}$  and  $s$  are the mean and standard deviation of the  $x_i$ , and

$$Z(Y; \mu, \sigma) = \left(1/\sqrt{2\pi\sigma^2}\right) e^{-(Y-\mu)^2/2\sigma^2}$$

is the density function of a normally distributed variable  $Y$  with mean  $\mu$  and standard deviation  $\sigma$ . The confidence interval for  $C_q$  is  $(c_q - z_{100(1-\alpha)}s_q, c_q + z_{100(1-\alpha)}s_q)$ .

## meansd case

The value of  $c_q$  is  $\bar{x} + z_q \times s$ . Its se is given by the formula

$$s_q^* = s \sqrt{1/n + z_q^2/(2n - 2)}$$

The confidence interval for  $C_q$  is  $(c_q - z_{100(1-\alpha)} \times s_q^*, c_q + z_{100(1-\alpha)} \times s_q^*)$ .

## Acknowledgment

`centile` was written by Patrick Royston, MRC Clinical Trials Unit, London.

## References

- Conover, W. J. 1999. *Practical Nonparametric Statistics*. 3rd ed. New York: Wiley.
- Kendall, M. G., and A. Stuart. 1969. *The Advanced Theory of Statistics, Vol. 1: Distribution Theory*. 3rd ed. London: Griffin.
- Mood, A. M., and F. A. Graybill. 1963. *Introduction to the Theory of Statistics*. 2nd ed. New York: McGraw-Hill.
- Newson, R. 2000. [snp16: Robust confidence intervals for median and other percentile differences between two groups](#). *Stata Technical Bulletin* 58: 30–35. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 324–331. College Station, TX: Stata Press.
- Royston, P. 1992. [sg7: Centile estimation command](#). *Stata Technical Bulletin* 8: 12–15. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 122–125. College Station, TX: Stata Press.
- Stuart, A., and J. K. Ord. 1994. *Kendall's Advanced Theory of Statistics: Distribution Theory, Vol I*. 6th ed. London: Arnold.

## Also see

- [\[R\] ci](#) — Confidence intervals for means, proportions, and counts
- [\[R\] summarize](#) — Summary statistics
- [\[D\] pctlile](#) — Create variable containing percentiles

Syntax

*Syntax for ci*

```
ci [varlist] [if] [in] [weight] [, options]
```

*Immediate command for variable distributed as normal*

```
cii #obs #mean #sd [, ciin_option]
```

*Immediate command for variable distributed as binomial*

```
cii #obs #succ [, ciib_options]
```

*Immediate command for variable distributed as Poisson*

```
cii #exposure #events , poisson [ciip_options]
```

| options                            | description                                                                         |
|------------------------------------|-------------------------------------------------------------------------------------|
| Main                               |                                                                                     |
| <u>b</u> inomial                   | binomial 0/1 variables; compute exact confidence intervals                          |
| <u>p</u> oisson                    | Poisson variables; compute exact confidence intervals                               |
| <u>e</u> xposure( <i>varname</i> ) | exposure variable; implies <u>p</u> oisson                                          |
| <u>e</u> xact                      | calculate exact confidence intervals; the default                                   |
| <u>w</u> ald                       | calculate Wald confidence intervals                                                 |
| <u>w</u> ilson                     | calculate Wilson confidence intervals                                               |
| <u>a</u> gresti                    | calculate Agresti–Coull confidence intervals                                        |
| <u>j</u> effreys                   | calculate Jeffreys confidence intervals                                             |
| <u>t</u> otal                      | add output for all groups combined (for use with <u>b</u> y only)                   |
| <u>s</u> eparator( <i>#</i> )      | draw separator line after every <i>#</i> variables; default is <u>s</u> eparator(5) |
| <u>l</u> evel( <i>#</i> )          | set confidence level; default is <u>l</u> evel(95)                                  |

by is allowed with ci; see [D] by.  
aweights and fweights are allowed, but aweights may not be specified with the binomial or poisson options; see [U] 11.1.6 weight.

| ciin_option               | description                                        |
|---------------------------|----------------------------------------------------|
| <u>l</u> evel( <i>#</i> ) | set confidence level; default is <u>l</u> evel(95) |

| <i>ciib_options</i>   | description                                             |
|-----------------------|---------------------------------------------------------|
| <code>level(#)</code> | set confidence level; default is <code>level(95)</code> |
| <code>exact</code>    | calculate exact confidence intervals; the default       |
| <code>wald</code>     | calculate Wald confidence intervals                     |
| <code>wilson</code>   | calculate Wilson confidence intervals                   |
| <code>agresti</code>  | calculate Agresti–Coull confidence intervals            |
| <code>jeffreys</code> | calculate Jeffreys confidence intervals                 |

| <i>ciip_options</i>   | description                                             |
|-----------------------|---------------------------------------------------------|
| <code>*poisson</code> | numbers are Poisson-distributed counts                  |
| <code>level(#)</code> | set confidence level; default is <code>level(95)</code> |

\*poisson is required.

## Menu

### ci

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Confidence intervals

### cii for variable distributed as normal

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Normal CI calculator

### cii for variable distributed as binomial

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Binomial CI calculator

### cii for variable distributed as Poisson

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Poisson CI calculator

## Description

`ci` computes standard errors and confidence intervals for each of the variables in *varlist*.

`cii` is the immediate form of `ci`; see [\[U\] 19 Immediate commands](#) for a general discussion of immediate commands.

In the binomial and Poisson variants of `cii`, the second number specified (`#succ` or `#events`) must be an integer or between 0 and 1. If the number is between 0 and 1, Stata interprets it as the fraction of successes or events and converts it to an integer number representing the number of successes or events. The computation then proceeds as if two integers had been specified.

## Options

### Main

`binomial` tells `ci` that the variables are 0/1 variables and that binomial confidence intervals will be calculated. (`cii` produces binomial confidence intervals when only two numbers are specified.)

`poisson` specifies that the variables (or numbers for `cii`) are Poisson-distributed counts; exact Poisson confidence intervals will be calculated.

`exposure(varname)` is used only with `poisson`. You do not need to specify `poisson` if you specify `exposure()`; `poisson` is assumed. `varname` contains the total exposure (typically a time or an area) during which the number of events recorded in `varlist` were observed.

`exact`, `wald`, `wilson`, `agresti`, and `jeffreys` specify that variables are 0/1 and specify how binomial confidence intervals are to be calculated.

`exact` is the default and specifies exact (also known in the literature as Clopper–Pearson [1934]) binomial confidence intervals.

`wald` specifies calculation of Wald confidence intervals.

`wilson` specifies calculation of Wilson confidence intervals.

`agresti` specifies calculation of Agresti–Coull confidence intervals.

`jeffreys` specifies calculation of Jeffreys confidence intervals.

See [Brown, Cai, and DasGupta \(2001\)](#) for a discussion and comparison of the different binomial confidence intervals.

`total` is for use with the `by` prefix. It requests that, in addition to output for each by-group, output be added for all groups combined.

`separator(#)` specifies how often separation lines should be inserted into the output. The default is `separator(5)`, meaning that a line is drawn after every five variables. `separator(10)` would draw the line after every 10 variables. `separator(0)` suppresses the separation line.

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see [\[R\] level](#).

## Remarks

Remarks are presented under the following headings:

- [Ordinary confidence intervals](#)
- [Binomial confidence intervals](#)
- [Poisson confidence intervals](#)
- [Immediate form](#)

## Ordinary confidence intervals

### ► Example 1

Without the `binomial` or `poisson` options, `ci` produces “ordinary” confidence intervals, meaning those that are correct if the variable is distributed normally, and *asymptotically* correct for all other distributions satisfying the conditions of the central limit theorem.

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. ci mpg price
```

| Variable | Obs | Mean     | Std. Err. | [95% Conf. Interval] |          |
|----------|-----|----------|-----------|----------------------|----------|
| mpg      | 74  | 21.2973  | .6725511  | 19.9569              | 22.63769 |
| price    | 74  | 6165.257 | 342.8719  | 5481.914             | 6848.6   |

The standard error of the mean of mpg is 0.67, and the 95% confidence interval is [19.96, 22.64]. We can obtain wider confidence intervals, 99%, by typing

```
. ci mpg price, level(99)
```

| Variable | Obs | Mean     | Std. Err. | [99% Conf. Interval] |          |
|----------|-----|----------|-----------|----------------------|----------|
| mpg      | 74  | 21.2973  | .6725511  | 19.51849             | 23.07611 |
| price    | 74  | 6165.257 | 342.8719  | 5258.405             | 7072.108 |



## Example 2

by() breaks out the confidence intervals according to by-group; total adds an overall summary. For instance,

```
. ci mpg, by(foreign) total
```

| -> foreign = Domestic |     |          |           |                      |          |
|-----------------------|-----|----------|-----------|----------------------|----------|
| Variable              | Obs | Mean     | Std. Err. | [95% Conf. Interval] |          |
| mpg                   | 52  | 19.82692 | .657777   | 18.50638             | 21.14747 |
| -> foreign = Foreign  |     |          |           |                      |          |
| Variable              | Obs | Mean     | Std. Err. | [95% Conf. Interval] |          |
| mpg                   | 22  | 24.77273 | 1.40951   | 21.84149             | 27.70396 |
| -> Total              |     |          |           |                      |          |
| Variable              | Obs | Mean     | Std. Err. | [95% Conf. Interval] |          |
| mpg                   | 74  | 21.2973  | .6725511  | 19.9569              | 22.63769 |



## Technical note

You can control the formatting of the numbers in the output by specifying a display format for the variable; see [U] [12.5 Formats: Controlling how data are displayed](#). For instance,

```
. format mpg %9.2f
```

```
. ci mpg
```

| Variable | Obs | Mean  | Std. Err. | [95% Conf. Interval] |       |
|----------|-----|-------|-----------|----------------------|-------|
| mpg      | 74  | 21.30 | 0.67      | 19.96                | 22.64 |



## Binomial confidence intervals

### Example 3

We have data on employees, including a variable marking whether the employee was promoted last year.

```
. use http://www.stata-press.com/data/r11/promo
```

```
. ci promoted, binomial
```

| Variable | Obs | Mean | Std. Err. | — Binomial Exact —   |          |
|----------|-----|------|-----------|----------------------|----------|
|          |     |      |           | [95% Conf. Interval] |          |
| promoted | 20  | .1   | .067082   | .0123485             | .3169827 |

The above interval is the default for binomial data, known equivalently as both the exact binomial and the Clopper–Pearson interval.

Nominally, the interpretation of a 95% confidence interval is that under repeated samples or experiments, 95% of the resultant intervals would contain the unknown parameter in question. However, for binomial data, the actual coverage probability, regardless of method, usually differs from that interpretation. This result occurs because of the discreteness of the binomial distribution, which produces only a finite set of outcomes, meaning that coverage probabilities are subject to discrete jumps and the exact nominal level cannot always be achieved. Therefore, the term *exact confidence interval* refers to its being derived from the binomial distribution, the distribution exactly generating the data, rather than resulting in exactly the nominal coverage.

For the Clopper–Pearson interval, the actual coverage probability is guaranteed to be greater than or equal to the nominal confidence interval, here 95%. Because of the way it is calculated—see [Methods and formulas](#)—it may also be interpreted as follows: If the true probability of being promoted were 0.012, the chances of observing a result as extreme or more extreme than the result observed ( $20 \times 0.1 = 2$  or more promotions) would be 2.5%. If the true probability of being promoted were 0.317, the chances of observing a result as extreme or more extreme than the result observed (two or fewer promotions) would be 2.5%.

◀

► Example 4

The Clopper–Pearson interval is desirable because it guarantees nominal coverage; however, by dropping this restriction, you may obtain accurate intervals that are not as conservative. In this vein, you might opt for the [Wilson \(1927\)](#) interval,

```
. ci promoted, binomial wilson
```

| Variable | Obs | Mean | Std. Err. | ——— Wilson ———       |          |
|----------|-----|------|-----------|----------------------|----------|
|          |     |      |           | [95% Conf. Interval] |          |
| promoted | 20  | .1   | .067082   | .0278665             | .3010336 |

the [Agresti–Coull \(1998\)](#) interval,

```
. ci promoted, binomial agresti
```

| Variable | Obs | Mean | Std. Err. | — Agresti–Coull —    |          |
|----------|-----|------|-----------|----------------------|----------|
|          |     |      |           | [95% Conf. Interval] |          |
| promoted | 20  | .1   | .067082   | .0156562             | .3132439 |

or the Bayesian-derived Jeffreys interval ([Brown, Cai, and DasGupta 2001](#)),

```
. ci promoted, binomial jeffreys
```

| Variable | Obs | Mean | Std. Err. | ——— Jeffreys ———     |          |
|----------|-----|------|-----------|----------------------|----------|
|          |     |      |           | [95% Conf. Interval] |          |
| promoted | 20  | .1   | .067082   | .0213725             | .2838533 |



Picking the best interval is a matter of balancing accuracy (coverage) against precision (average interval length) and depends on sample size and success probability. Brown, Cai, and DasGupta (2001) recommend the Wilson or Jeffreys interval for small sample sizes ( $\leq 40$ ) yet favor the Agresti–Coul interval for its simplicity, decent performance for sample sizes less than or equal to 40, and performance comparable to Wilson/Jeffreys for sample sizes greater than 40. They also deem the Clopper–Pearson interval to be “wastefully conservative and [...] not a good choice for practical use”, unless of course one requires, at a minimum, the nominal coverage level.

◀

Finally, the binomial Wald confidence interval is obtained by specifying the `binomial` and `wald` options. The Wald interval is the one taught in most introductory statistics courses and for the above is simply, for level  $1 - \alpha$ ,  $\text{Mean} \pm z_\alpha(\text{Std. Err.})$ , where  $z_\alpha$  is the  $1 - \alpha/2$  quantile of the standard normal. Because its overall poor performance makes it impractical, the Wald interval is available mainly for pedagogical purposes. The binomial Wald interval is also similar to the interval produced by treating binary data as normal data and using `ci` without the `binomial` option, with two exceptions. First, when `binomial` is specified, the calculation of the standard error uses denominator  $n$  rather than  $n - 1$ , used for normal data. Second, confidence intervals for normal data are based on the  $t$  distribution rather than the standard normal. Of course, both discrepancies vanish as sample size increases.

## □ Technical note

Let’s repeat example 3, but this time with data in which there are no promotions over the observed period:

```
. use http://www.stata-press.com/data/r11/promonone
. ci promoted, binomial
```

| Variable                                 | Obs | Mean | Std. Err. | — Binomial Exact —<br>[95% Conf. Interval] |           |
|------------------------------------------|-----|------|-----------|--------------------------------------------|-----------|
| promoted                                 | 20  | 0    | 0         | 0                                          | .1684335* |
| (*) one-sided, 97.5% confidence interval |     |      |           |                                            |           |

The confidence interval is  $[0, 0.168]$ , and this is the confidence interval that most books publish. It is not, however, a true 95% confidence interval because the lower tail has vanished. As Stata notes, it is a one-sided, 97.5% confidence interval. If you wanted to put 5% in the right tail, you could type `ci promoted, binomial level(90)`.

□

## □ Technical note

`ci` with the `binomial` option ignores any variables that do not take on the values 0 and 1 exclusively. For instance, with our automobile dataset,

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. ci mpg foreign, binomial
```

| Variable | Obs | Mean     | Std. Err. | — Binomial Exact —<br>[95% Conf. Interval] |          |
|----------|-----|----------|-----------|--------------------------------------------|----------|
| foreign  | 74  | .2972973 | .0531331  | .196584                                    | .4148353 |

We also requested the confidence interval for `mpg`, but Stata ignored us. It does that so you can type `ci, binomial` and obtain correct confidence intervals for all the variables that are 0/1 in your data.

□

Poisson confidence intervals

➤ Example 5

We have data on the number of bacterial colonies on a Petri dish. The dish has been divided into 36 small squares, and the number of colonies in each square has been counted. Each observation in our dataset represents a square on the dish. The variable `count` records the number of colonies in each square counted, which varies from 0 to 5.

```
. use http://www.stata-press.com/data/r11/petri
. ci count, poisson
```

| Variable | Exposure | Mean     | Std. Err. | — Poisson<br>[95% Conf. Interval] | Exact —  |
|----------|----------|----------|-----------|-----------------------------------|----------|
| count    | 36       | 2.333333 | .2545875  | 1.861158                          | 2.888825 |

`ci` reports that the average number of colonies per square is 2.33. If the expected number of colonies per square were as low as 1.86, the probability of observing 2.33 or more colonies per square would be 2.5%. If the expected number were as large as 2.89, the probability of observing  $36 \times 2.33 = 84$  or fewer colonies per square would be 2.5%.



□ Technical note

The number of “observations”—how finely the Petri dish is divided—makes no difference. The Poisson distribution is a function only of the count. In example 4, we observed a total of  $2.33 \times 36 = 84$  colonies and a confidence interval of  $[1.86 \times 36, 2.89 \times 36] = [67, 104]$ . We would obtain the same  $[67, 104]$  confidence interval if our dish were divided into, say, 49 squares, rather than 36.

For the counts, it is not even important that all the squares be of the same size. For *rates*, however, such differences do matter, but in an easy-to-calculate way. Rates are obtained from counts by dividing by exposure, which is typically a number multiplied by either time or an area. For our Petri dishes, we divide by an area to obtain a rate, but if our example were cast in terms of being infected by a disease, we might divide by person-years to obtain the rate. Rates are convenient because they are easier to compare: we might have 2.3 colonies per square inch or 0.0005 infections per person-year.

So, let’s assume that we wish to obtain the number of colonies per square inch, and, moreover, that not all the “squares” on our dish are of equal size. We have a variable called `area` that records the area of each “square”:

```
. ci count, exposure(area)
```

| Variable | Exposure | Mean | Std. Err. | — Poisson<br>[95% Conf. Interval] | Exact —  |
|----------|----------|------|-----------|-----------------------------------|----------|
| count    | 3        | 28   | 3.055051  | 22.3339                           | 34.66591 |

The rates are now in more familiar terms. In our sample, there are 28 colonies per square inch and the 95% confidence interval is  $[22.3, 34.7]$ . When we did not specify `exposure()`, `ci` assumed that each observation contributed 1 to exposure.



## □ Technical note

As with the binomial option, if there were no colonies on our dish, ci would calculate a one-sided confidence interval:

```
. use http://www.stata-press.com/data/r11/petrinone
. ci count, poisson
```

| Variable | Exposure | Mean | Std. Err. | — Poisson [95% Conf. Interval] | Exact Interval |
|----------|----------|------|-----------|--------------------------------|----------------|
| count    | 36       | 0    | 0         | 0                              | .1024689*      |

(\*) one-sided, 97.5% confidence interval



## Immediate form

### ▷ Example 6

We are reading a soon-to-be-published paper by a colleague. In it is a table showing the number of observations, mean, and standard deviation of 1980 median family income for the Northeast and West. We correctly think that the paper would be much improved if it included the confidence intervals. The paper claims that for 166 cities in the Northeast, the average of median family income is \$19,509 with a standard deviation of \$4,379:

For the Northeast:

```
. cii 166 19509 4379
```

| Variable | Obs | Mean  | Std. Err. | [95% Conf. Interval] |          |
|----------|-----|-------|-----------|----------------------|----------|
|          | 166 | 19509 | 339.8763  | 18837.93             | 20180.07 |

For the West:

```
. cii 256 22557 5003
```

| Variable | Obs | Mean  | Std. Err. | [95% Conf. Interval] |          |
|----------|-----|-------|-----------|----------------------|----------|
|          | 256 | 22557 | 312.6875  | 21941.22             | 23172.78 |



### ▷ Example 7

We flip a coin 10 times, and it comes up heads only once. We are shocked and decide to obtain a 99% confidence interval for this coin:

```
. cii 10 1, level(99)
```

| Variable | Obs | Mean | Std. Err. | — Binomial [99% Conf. Interval] | Exact Interval |
|----------|-----|------|-----------|---------------------------------|----------------|
|          | 10  | .1   | .0948683  | .0005011                        | .5442871       |



➤ Example 8

The number of reported traffic accidents in Santa Monica over a 24-hour period is 27. We need know nothing else:

```
. ci 1 27, poisson
```

| Variable | Exposure | Mean | Std. Err. | — Poisson<br>[95% Conf. Interval] | Exact<br>Interval |
|----------|----------|------|-----------|-----------------------------------|-------------------|
|          | 1        | 27   | 5.196152  | 17.79317                          | 39.28358          |



Saved results

ci and cii saves the following in `r()`:

|                      |                                    |                    |                                    |
|----------------------|------------------------------------|--------------------|------------------------------------|
| Scalars              |                                    |                    |                                    |
| <code>r(N)</code>    | number of observations or exposure | <code>r(lb)</code> | lower bound of confidence interval |
| <code>r(mean)</code> | mean                               | <code>r(ub)</code> | upper bound of confidence interval |
| <code>r(se)</code>   | estimate of standard error         |                    |                                    |

Methods and formulas

ci and cii are implemented as ado-files.  
Methods and formulas are presented under the following headings:

- Ordinary
- Binomial
- Poisson

Ordinary

Define  $n$ ,  $\bar{x}$ , and  $s^2$  as, respectively, the number of observations, (weighted) average, and (unbiased) estimated variance of the variable in question; see [\[R\] summarize](#).

The standard error of the mean,  $s_\mu$ , is defined as  $\sqrt{s^2/n}$ .

Let  $\alpha$  be  $1 - l/100$ , where  $l$  is the significance level specified by the user. Define  $t_\alpha$  as the two-sided  $t$  statistic corresponding to a significance level of  $\alpha$  with  $n - 1$  degrees of freedom;  $t_\alpha$  is obtained from Stata as `invttail(n-1,0.5*\alpha)`. The lower and upper confidence bounds are, respectively,  $\bar{x} - s_\mu t_\alpha$  and  $\bar{x} + s_\mu t_\alpha$ .

Binomial

Given  $k$  successes of  $n$  trials, the estimated probability is  $\hat{p} = k/n$  with standard error  $\sqrt{\hat{p}(1 - \hat{p})/n}$ . `ci` calculates the exact (Clopper–Pearson) confidence interval  $[p_1, p_2]$  such that

$$\Pr(K \geq k | p = p_1) = \alpha/2$$

and

$$\Pr(K \leq k | p = p_2) = \alpha/2$$

where  $K$  is distributed as  $\text{binomial}(n, p)$ . The endpoints may be obtained directly by using Stata's `invbinomial()` function. If  $k = 0$  or  $k = n$ , the calculation of the appropriate tail is skipped.

The Wald interval is  $\hat{p} \pm z_\alpha \sqrt{\hat{p}(1-\hat{p})/n}$ , where  $z_\alpha$  is the  $1 - \alpha/2$  quantile of the standard normal. The interval is obtained by inverting the acceptance region of the large-sample Wald test of  $H_0: p = p_0$  versus the two-sided alternative. That is, the confidence interval is the set of all  $p_0$  such that

$$\left| \frac{\hat{p} - p_0}{\sqrt{n^{-1}\hat{p}(1-\hat{p})}} \right| \leq z_\alpha$$

The Wilson interval is a variation on the Wald interval, using the null standard error  $\sqrt{n^{-1}p_0(1-p_0)}$  in place of the estimated standard error  $\sqrt{n^{-1}\hat{p}(1-\hat{p})}$  in the above expression. Inverting this acceptance region is more complicated yet results in the closed form

$$\frac{k + z_\alpha^2/2}{n + z_\alpha^2} \pm \frac{z_\alpha n^{1/2}}{n + z_\alpha^2/2} \left\{ \hat{p}(1-\hat{p}) + \frac{z_\alpha^2}{4n} \right\}^{1/2}$$

The Agresti–Coull interval is basically a Wald interval that borrows its center from the Wilson interval. Defining  $\tilde{k} = k + z_\alpha^2/2$ ,  $\tilde{n} = n + z_\alpha^2$ , and (hence)  $\tilde{p} = \tilde{k}/\tilde{n}$ , the Agresti–Coull interval is

$$\tilde{p} \pm z_\alpha \sqrt{\tilde{p}(1-\tilde{p})/\tilde{n}}$$

When  $\alpha = 0.05$ ,  $z_\alpha$  is near enough to 2 that  $\tilde{p}$  can be thought of as a typical estimate of proportion where two successes and two failures have been added to the sample (Agresti and Coull 1998). This typical estimate of proportion makes the Agresti–Coull interval an easy-to-present alternative for introductory statistics students.

The Jeffreys interval is a Bayesian interval and is based on the Jeffreys prior, which is the  $\text{Beta}(1/2, 1/2)$  distribution. Assigning this prior to  $p$  results in a posterior distribution for  $p$  that is Beta with parameters  $k + 1/2$  and  $n - k + 1/2$ . The Jeffreys interval is then taken to be the  $1 - \alpha$  central posterior probability interval, namely, the  $\alpha/2$  and  $1 - \alpha/2$  quantiles of the  $\text{Beta}(k + 1/2, n - k + 1/2)$  distribution. These quantiles may be obtained directly by using Stata's `invibeta()` function.

## Poisson

Given the total cases,  $k$ , the estimate of the expected count  $\lambda$  is  $k$ , and its standard error is  $\sqrt{k}$ . `ci` calculates the exact confidence interval  $[\lambda_1, \lambda_2]$  such that

$$\Pr(K \geq k | \lambda = \lambda_1) = \alpha/2$$

and

$$\Pr(K \leq k | \lambda = \lambda_2) = \alpha/2$$

where  $K$  is Poisson with mean  $\lambda$ . Solution is by Newton's method. If  $k = 0$ , the calculation of  $\lambda_1$  is skipped. All values are then reported as rates, which are the above numbers divided by the total exposure.

Harold Jeffreys (1891–1989) was born near Durham, England, and spent more than 75 years studying and working at the University of Cambridge, principally on theoretical and observational problems in geophysics, astronomy, mathematics, and statistics. He developed a systematic Bayesian approach to inference in his monograph *Theory of Probability*.

E. B. Wilson (1879–1964) majored in mathematics at Harvard and studied and taught at Yale and MIT before returning to Harvard in 1922. He worked in mathematics, physics, and statistics. His method for binomial intervals can be considered a precursor, for a particular problem, of Neyman’s concept of confidence intervals.

Jerzy Neyman (1894–1981) was born in Bendery, Moldavia. He studied and then taught at Kharkov University, moving from physics to mathematics. In 1921, Neyman moved to Poland, where he worked in statistics at Bydgoszcz and then Warsaw. Receiving a Rockefeller Fellowship to work with Karl Pearson at University College London, he befriended Egon Pearson, Karl’s son, and they worked together on the theory of hypothesis testing. Life in Poland became progressively more difficult, and Neyman returned to UCL to work there from 1934 to 1938. At this time, he published on the theory of confidence intervals. He then was offered a post in California at Berkeley, where he settled. Neyman established an outstanding statistics department and remained highly active in research, including applications in astronomy, meteorology, and medicine. He was one of the great statisticians of the 20th century.

## Acknowledgment

We thank Nicholas J. Cox of Durham University for his assistance with the `jeffreys` and `wilson` options.

## References

- Agresti, A., and B. A. Coull. 1998. Approximate is better than “exact” for interval estimation of binomial proportions. *American Statistician* 52: 119–126.
- Brown, L. D., T. T. Cai, and A. DasGupta. 2001. Interval estimation for a binomial proportion. *Statistical Science* 16: 101–133.
- Campbell, M. J., D. Machin, and S. J. Walters. 2007. *Medical Statistics: A Textbook for the Health Sciences*. 4th ed. Chichester, UK: Wiley.
- Clopper, C. J., and E. S. Pearson. 1934. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika* 26: 404–413.
- Cook, A. 1990. Sir Harold Jeffreys. 2 April 1891–18 March 1989. *Biographical Memoirs of Fellows of the Royal Society* 36: 303–333.
- Gleason, J. R. 1999. [sg119: Improved confidence intervals for binomial proportions](#). *Stata Technical Bulletin* 52: 16–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 208–211. College Station, TX: Stata Press.
- Jeffreys, H. 1946. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London, Series A* 186: 453–461.
- Lindley, D. V. 2001. Harold Jeffreys. In *Statisticians of the Centuries*, ed. C. C. Heyde and E. Seneta, 402–405. New York: Springer.
- Reid, C. 1982. *Neyman—from Life*. New York: Springer.
- Rothman, K. J., S. Greenland, and T. L. Lash. 2008. *Modern Epidemiology*. 3rd ed. Philadelphia: Lippincott Williams & Wilkins.

- Seed, P. T. 2001. [sg159: Confidence intervals for correlations](#). *Stata Technical Bulletin* 59: 27–28. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 267–269. College Station, TX: Stata Press.
- Stigler, S. M. 1997. Wilson, Edwin Bidwell. In *Leading Personalities in Statistical Sciences: From the Seventeenth Century to the Present*, ed. N. L. Johnson and S. Kotz, 344–346. New York: Wiley.
- Utts, J. M. 2005. *Seeing Through Statistics*. 3rd ed. Belmont, CA: Brooks/Cole.
- Wang, D. 2000. [sg154: Confidence intervals for the ratio of two binomial proportions by Koopman's method](#). *Stata Technical Bulletin* 58: 16–19. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 244–247. College Station, TX: Stata Press.
- Wilson, E. B. 1927. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association* 22: 209–212.

## Also see

- [R] [bitest](#) — Binomial probability test
- [R] [prtest](#) — One- and two-sample tests of proportions
- [R] [ttest](#) — Mean-comparison tests
- [R] [ameans](#) — Arithmetic, geometric, and harmonic means
- [R] [centile](#) — Report centile and confidence interval
- [R] [summarize](#) — Summary statistics
- [D] [pctile](#) — Create variable containing percentiles

Syntax

```
clogit depvar [indepvars] [if] [in] [weight] , group(varname) [options]
```

| <i>options</i>                        | description                                                                                                                             |
|---------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Model                                 |                                                                                                                                         |
| * <u>group</u> ( <i>varname</i> )     | matched group variable                                                                                                                  |
| <u>offset</u> ( <i>varname</i> )      | include <i>varname</i> in model with coefficient constrained to 1                                                                       |
| <u>constraints</u> ( <i>numlist</i> ) | apply specified linear constraints                                                                                                      |
| <u>collinear</u>                      | keep collinear variables                                                                                                                |
| SE/Robust                             |                                                                                                                                         |
| <u>vce</u> ( <i>vcetype</i> )         | <i>vcetype</i> may be <u>oim</u> , <u>robust</u> , <u>cluster</u> <i>clustvar</i> , <u>opg</u> , <u>bootstrap</u> , or <u>jackknife</u> |
| <u>nonest</u>                         | do not check that panels are nested within clusters                                                                                     |
| Reporting                             |                                                                                                                                         |
| <u>level</u> (#)                      | set confidence level; default is <u>level</u> (95)                                                                                      |
| or                                    | report odds ratios                                                                                                                      |
| <u>nocnsreport</u>                    | do not display constraints                                                                                                              |
| <u>display_options</u>                | control spacing and display of omitted variables and base and empty cells                                                               |
| Maximization                          |                                                                                                                                         |
| <u>maximize_options</u>               | control the maximization process; seldom used                                                                                           |
| <sup>†</sup> <u>coeflegend</u>        | display coefficients' legend instead of coefficient table                                                                               |

\*group(*varname*) is required.

<sup>†</sup>coeflegend does not appear in the dialog box.

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

bootstrap, by, fracpoly, jackknife, mfp, mi estimate, nestreg, rolling, statsby, stepwise, and svy are allowed; see [U] 11.1.10 Prefix commands.

vce(bootstrap) and vce(jackknife) are not allowed with the mi estimate prefix.

Weights are not allowed with the bootstrap prefix.

vce(), nonest, and weights are not allowed with the svy prefix.

fweights, iweights, and pweights are allowed (see [U] 11.1.6 weight), but they are interpreted to apply to groups as a whole, not to individual observations. See Use of weights below.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Categorical outcomes > Conditional logistic regression



## Description

`clogit` fits what biostatisticians and epidemiologists call conditional logistic regression for matched case–control groups (see, for example, [Hosmer Jr. and Lemeshow \[2000, chap. 7\]](#)) and what economists and other social scientists call fixed-effects logit for panel data (see, for example, [Chamberlain \[1980\]](#)). Computationally, these models are the same.

See [\[R\] `asclogit`](#) if you want to fit McFadden’s choice model ([McFadden 1974](#)). Also see [\[R\] `logistic`](#) for a list of related estimation commands.

## Options

### Model

`group(varname)` is required; it specifies an identifier variable (numeric or string) for the matched groups. `strata(varname)` is a synonym for `group()`.

`offset(varname)`, `constraints(numlist)`, `collinear`; see [\[R\] `estimation options`](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [\[R\] `vce\_option`](#).

`nonest`, available only with `vce(cluster clustvar)`, prevents checking that matched groups are nested within clusters. It is the user’s responsibility to verify that the standard errors are theoretically correct.

### Reporting

`level(#)`; see [\[R\] `estimation options`](#).

`or` reports the estimated coefficients transformed to odds ratios, i.e.,  $e^b$  rather than  $b$ . Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated. `or` may be specified at estimation or when replaying previously estimated results.

`nocnsreport`; see [\[R\] `estimation options`](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [\[R\] `estimation options`](#).

### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [\[R\] `maximize`](#). These options are seldom used.

Setting the optimization type to `technique(bhhh)` resets the default `vcetype` to `vce(opg)`.

The following option is available with `clogit` but is not shown in the dialog box:

`coeflegend`; see [\[R\] `estimation options`](#).

## Remarks

Remarks are presented under the following headings:

[Introduction](#)

[Matched case–control data](#)

[Use of weights](#)

[Fixed-effects logit](#)

## Introduction

`clogit` fits maximum likelihood models with a dichotomous dependent variable coded as 0/1 (more precisely, `clogit` interprets 0 and not 0 to indicate the dichotomy). Conditional logistic analysis differs from regular logistic regression in that the data are grouped and the likelihood is calculated relative to each group; i.e., a conditional likelihood is used. See [Methods and formulas](#) at the end of this entry.

Biostatisticians and epidemiologists fit these models when analyzing matched case–control studies with 1:1 matching, 1: $k_{2i}$  matching, or  $k_{1i}:k_{2i}$  matching, where  $i$  denotes the  $i$ th matched group for  $i = 1, 2, \dots, n$ , where  $n$  is the total number of groups. `clogit` fits a model appropriate for all these matching schemes or for any mix of the schemes because the matching  $k_{1i}:k_{2i}$  can vary from group to group. `clogit` always uses the true conditional likelihood, not an approximation. Biostatisticians and epidemiologists sometimes refer to the matched groups as “strata”, but we will stick to the more generic term “group”.

Economists and other social scientists fitting fixed-effects logit models have data that look exactly like the data biostatisticians and epidemiologists call  $k_{1i}:k_{2i}$  matched case–control data. In terms of how the data are arranged,  $k_{1i}:k_{2i}$  matching means that in the  $i$ th group, the dependent variable is 1 a total of  $k_{1i}$  times and 0 a total of  $k_{2i}$  times. There are a total of  $T_i = k_{1i} + k_{2i}$  observations for the  $i$ th group. This data arrangement is what economists and other social scientists call “panel data”, “longitudinal data”, or “cross-sectional time-series data”.

So no matter what terminology you use, the computation and the use of the `clogit` command is the same. The following example shows how your data should be arranged to use `clogit`.

### ► Example 1

Suppose that we have grouped data with the variable `id` containing a unique identifier for each group. Our outcome variable, `y`, contains 0s and 1s. If we were biostatisticians,  $y = 1$  would indicate a case,  $y = 0$  would be a control, and `id` would be an identifier variable that indicates the groups of matched case–control subjects.

If we were economists,  $y = 1$  might indicate that a person was unemployed at any time during a year and  $y = 0$ , that a person was employed all year, and `id` would be an identifier variable for persons.

If we list the first few observations of this dataset, it looks like

```
. use http://www.stata-press.com/data/r11/clogitid
. list y x1 x2 id in 1/11
```

|     | y | x1 | x2 | id   |
|-----|---|----|----|------|
| 1.  | 0 | 0  | 4  | 1014 |
| 2.  | 0 | 1  | 4  | 1014 |
| 3.  | 0 | 1  | 6  | 1014 |
| 4.  | 1 | 1  | 8  | 1014 |
| 5.  | 0 | 0  | 1  | 1017 |
| 6.  | 0 | 0  | 7  | 1017 |
| 7.  | 1 | 1  | 10 | 1017 |
| 8.  | 0 | 0  | 1  | 1019 |
| 9.  | 0 | 1  | 7  | 1019 |
| 10. | 1 | 1  | 7  | 1019 |
| 11. | 1 | 1  | 9  | 1019 |

Pretending that we are biostatisticians, we describe our data as follows. The first group ( $\text{id} = 1014$ ) consists of four matched persons: 1 case ( $y = 1$ ) and three controls ( $y = 0$ ), i.e., 1:3 matching. The second group has 1:2 matching, and the third 2:2.

Pretending that we are economists, we describe our data as follows. The first group consists of 4 observations (one per year) for person 1014. This person had a period of unemployment during 1 year of 4. The second person had a period of unemployment during 1 year of 3, and the third had a period of 2 years of 4.

Our independent variables are  $x1$  and  $x2$ . To fit the conditional (fixed-effects) logistic model, we type

```
. clogit y x1 x2, group(id)
note: multiple positive outcomes within groups encountered.

Iteration 0: log likelihood = -123.42828
Iteration 1: log likelihood = -123.41386
Iteration 2: log likelihood = -123.41386

Conditional (fixed-effects) logistic regression
Number of obs = 369
LR chi2(2) = 9.07
Prob > chi2 = 0.0107
Pseudo R2 = 0.0355

Log likelihood = -123.41386
```

| y  | Coef.    | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|----|----------|-----------|------|-------|----------------------|----------|
| x1 | .653363  | .2875215  | 2.27 | 0.023 | .0898312             | 1.216895 |
| x2 | .0659169 | .0449555  | 1.47 | 0.143 | -.0221943            | .1540281 |

◀

## □ Technical note

The message “note: multiple positive outcomes within groups encountered” at the top of the `clogit` output for the previous example merely informs us that we have  $k_{1i} : k_{2i}$  matching with  $k_{1i} > 1$  for at least one group. If your data should be 1:  $k_{2i}$  matched, this message tells you that there is an error in the data somewhere.

We can see the distribution of  $k_{1i}$  and  $T_i = k_{1i} + k_{2i}$  for the data of the previous example by using the following steps:

```
. by id, sort: gen k1 = sum(y)
. by id: replace k1 = . if _n < _N
(303 real changes made, 303 to missing)
. by id: gen T = sum(y < .)
. by id: replace T = . if _n < _N
(303 real changes made, 303 to missing)
. tab k1
```

| k1    | Freq. | Percent | Cum.   |
|-------|-------|---------|--------|
| 1     | 48    | 72.73   | 72.73  |
| 2     | 12    | 18.18   | 90.91  |
| 3     | 4     | 6.06    | 96.97  |
| 4     | 2     | 3.03    | 100.00 |
| Total | 66    | 100.00  |        |

```
. tab T
```

| T     | Freq. | Percent | Cum.   |
|-------|-------|---------|--------|
| 2     | 5     | 7.58    | 7.58   |
| 3     | 5     | 7.58    | 15.15  |
| 4     | 12    | 18.18   | 33.33  |
| 5     | 11    | 16.67   | 50.00  |
| 6     | 13    | 19.70   | 69.70  |
| 7     | 8     | 12.12   | 81.82  |
| 8     | 3     | 4.55    | 86.36  |
| 9     | 7     | 10.61   | 96.97  |
| 10    | 2     | 3.03    | 100.00 |
| Total | 66    | 100.00  |        |

We see that  $k_{1i}$  ranges from 1 to 4 and  $T_i$  ranges from 2 to 10 for these data.

Technical note

For  $k_{1i} : k_{2i}$  matching (and hence in the general case of fixed-effects logit), `clogit` uses a recursive algorithm to compute the likelihood, which means that there are no limits on the size of  $T_i$ . However, computation time is proportional to  $\sum T_i \min(k_{1i}, k_{2i})$ , so `clogit` will take roughly 10 times longer to fit a model with 10:10 matching than one with 1:10 matching. But `clogit` is fast, so computation time becomes an issue only when  $\min(k_{1i}, k_{2i})$  is around 100 or more. See [Methods and formulas](#) for details.

Matched case–control data

Here we give a more detailed example of matched case–control data.

## ► Example 2

Hosmer Jr. and Lemeshow (2000, 25) present data on matched pairs of infants, each pair having one with low birthweight and another with regular birthweight. The data are matched on age of the mother. Several possible maternal exposures are considered: race (three categories), smoking status, presence of hypertension, presence of uterine irritability, previous preterm delivery, and weight at the last menstrual period.

```
. use http://www.stata-press.com/data/r11/lowbirth2, clear
(Applied Logistic Regression, Hosmer & Lemeshow)

. describe

Contains data from http://www.stata-press.com/data/r11/lowbirth2.dta
 obs: 112 Applied Logistic Regression,
 Hosmer & Lemeshow
 26 Apr 2009 09:33
 vars: 9
 size: 1,904 (99.9% of memory free)
```

| variable name | storage<br>type | display<br>format | value<br>label | variable label                               |
|---------------|-----------------|-------------------|----------------|----------------------------------------------|
| pairid        | byte            | %8.0g             |                | Case-control pair ID                         |
| low           | byte            | %8.0g             |                | Baby has low birthweight                     |
| age           | byte            | %8.0g             |                | Age of mother                                |
| lwt           | int             | %8.0g             |                | Mother's last menstrual weight               |
| smoke         | byte            | %8.0g             |                | Mother smoked during pregnancy               |
| ptd           | byte            | %8.0g             |                | Mother had previous preterm baby             |
| ht            | byte            | %8.0g             |                | Mother has hypertension                      |
| ui            | byte            | %8.0g             |                | Uterine irritability                         |
| race          | float           | %9.0g             |                | race of mother: 1=white, 2=black,<br>3=other |

Sorted by:

We list the case–control indicator variable, `low`; the match identifier variable, `pairid`; and two of the covariates, `lwt` and `smoke`, for the first 10 observations.

```
. list low lwt smoke pairid in 1/10
```

|     | low | lwt | smoke | pairid |
|-----|-----|-----|-------|--------|
| 1.  | 0   | 135 | 0     | 1      |
| 2.  | 1   | 101 | 1     | 1      |
| 3.  | 0   | 98  | 0     | 2      |
| 4.  | 1   | 115 | 0     | 2      |
| 5.  | 0   | 95  | 0     | 3      |
| 6.  | 1   | 130 | 0     | 3      |
| 7.  | 0   | 103 | 0     | 4      |
| 8.  | 1   | 130 | 1     | 4      |
| 9.  | 0   | 122 | 1     | 5      |
| 10. | 1   | 110 | 1     | 5      |

We fit a conditional logistic model of low birthweight on mother's weight, race, smoking behavior, and history.

```
. clogit low lwt smoke ptd ht ui i.race, group(pairid) nolog
Conditional (fixed-effects) logistic regression Number of obs = 112
 LR chi2(7) = 26.04
 Prob > chi2 = 0.0005
Log likelihood = -25.794271 Pseudo R2 = 0.3355
```

| low   | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-------|-----------|-----------|-------|-------|----------------------|----------|
| lwt   | -.0183757 | .0100806  | -1.82 | 0.068 | -.0381333            | .0013819 |
| smoke | 1.400656  | .6278396  | 2.23  | 0.026 | .1701131             | 2.631199 |
| ptd   | 1.808009  | .7886502  | 2.29  | 0.022 | .2622828             | 3.353735 |
| ht    | 2.361152  | 1.086128  | 2.17  | 0.030 | .2323796             | 4.489924 |
| ui    | 1.401929  | .6961585  | 2.01  | 0.044 | .0374836             | 2.766375 |
| race  |           |           |       |       |                      |          |
| 2     | .5713643  | .689645   | 0.83  | 0.407 | -.7803149            | 1.923044 |
| 3     | -.0253148 | .6992044  | -0.04 | 0.971 | -1.39573             | 1.345101 |

We might prefer to see results presented as odds ratios. We could have specified the `or` option when we first fit the model, or we can now redisplay results and specify `or`:

```
. clogit, or
Conditional (fixed-effects) logistic regression Number of obs = 112
 LR chi2(7) = 26.04
 Prob > chi2 = 0.0005
Log likelihood = -25.794271 Pseudo R2 = 0.3355
```

| low   | Odds Ratio | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-------|------------|-----------|-------|-------|----------------------|----------|
| lwt   | .9817921   | .009897   | -1.82 | 0.068 | .9625847             | 1.001383 |
| smoke | 4.057862   | 2.547686  | 2.23  | 0.026 | 1.185439             | 13.89042 |
| ptd   | 6.098293   | 4.80942   | 2.29  | 0.022 | 1.299894             | 28.60938 |
| ht    | 10.60316   | 11.51639  | 2.17  | 0.030 | 1.261599             | 89.11467 |
| ui    | 4.06303    | 2.828513  | 2.01  | 0.044 | 1.038195             | 15.90088 |
| race  |            |           |       |       |                      |          |
| 2     | 1.770681   | 1.221141  | 0.83  | 0.407 | .4582617             | 6.84175  |
| 3     | .975003    | .6817263  | -0.04 | 0.971 | .2476522             | 3.838573 |

Smoking, previous preterm delivery, hypertension, uterine irritability, and possibly the mother’s weight all contribute to low birthweight. `2.race` (mother black) and `3.race` (mother other) are statistically insignificant when compared with the `1.race` (mother white) omitted group, although the `2.race` effect is large. We can test the joint statistical significance of `2.race` and `3.race` by using `test`:

```
. test 2.race 3.race
(1) [low]2.race = 0
(2) [low]3.race = 0

 chi2(2) = 0.88
Prob > chi2 = 0.6436
```

For a more complete description of `test`, see [\[R\] test](#). `test` presents results in coefficients rather than odds ratios. Jointly testing that the coefficients on `2.race` and `3.race` are 0 is equivalent to jointly testing that the odds ratios are 1.

Here one case was matched to one control, i.e., 1:1 matching. From `clogit`'s point of view, that was not important— $k_1$  cases could have been matched to  $k_2$  controls ( $k_1:k_2$  matching), and we would have fit the model in the same way. Furthermore, the matching can change from group to group, which we have denoted as  $k_{1i}:k_{2i}$  matching, where  $i$  denotes the group. `clogit` does not care. To fit the conditional logistic regression model, we specified the `group(varname)` option, `group(pairid)`. The case and control are stored in separate observations. `clogit` knew that they were linked (in the same group) because the related observations share the same value of `pairid`.

◀

## □ Technical note

`clogit` provides a way to extend McNemar's test to multiple controls per case (1: $k_{2i}$  matching) and to multiple controls matched with multiple cases ( $k_{1i}:k_{2i}$  matching).

In Stata, McNemar's test is calculated by the `mcc` command; see [ST] [epitab](#). The `mcc` command, however, requires that the matched case and control appear in one observation, so the data will need to be manipulated from 1 to 2 observations per stratum before using `clogit`. Alternatively, if you begin with `clogit`'s 2-observations-per-group organization, you will have to change it to 1 observation per group if you wish to use `mcc`. In either case, `reshape` provides an easy way to change the organization of the data. We will demonstrate its use below, but we direct you to [D] [reshape](#) for a more thorough discussion.

In the example above, we used `clogit` to analyze the relationship between low birthweight and various characteristics of the mother. Assume that we now want to assess the relationship between low birthweight and smoking, ignoring the mother's other characteristics. Using `clogit`, we obtain the following results:

```
. clogit low smoke, group(pairid) or
Iteration 0: log likelihood = -35.425931
Iteration 1: log likelihood = -35.419283
Iteration 2: log likelihood = -35.419282

Conditional (fixed-effects) logistic regression Number of obs = 112
 LR chi2(1) = 6.79
 Prob > chi2 = 0.0091
Log likelihood = -35.419282 Pseudo R2 = 0.0875
```

| low   | Odds Ratio | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|-------|------------|-----------|------|-------|----------------------|----------|
| smoke | 2.75       | 1.135369  | 2.45 | 0.014 | 1.224347             | 6.176763 |

Let's compare our estimated odds ratio and 95% confidence interval with that produced by `mcc`. We begin by reshaping the data:

```
. keep low smoke pairid
. reshape wide smoke, i(pairid) j(low 0 1)

Data long -> wide

Number of obs. 112 -> 56
Number of variables 3 -> 3
j variable (2 values) low -> (dropped)
xij variables:
 smoke -> smoke0 smoke1
```

We now have the variables `smoke0` (formed from `smoke` and `low = 0`), recording 1 if the control mother smoked and 0 otherwise; and `smoke1` (formed from `smoke` and `low = 1`), recording 1 if the case mother smoked and 0 otherwise. We can now use `mcc`:

```
. mcc smoke1 smoke0
```

| Cases     | Controls<br>Exposed | Unexposed | Total |
|-----------|---------------------|-----------|-------|
| Exposed   | 8                   | 22        | 30    |
| Unexposed | 8                   | 18        | 26    |
| Total     | 16                  | 40        | 56    |

McNemar's chi2(1) = 6.53      Prob > chi2 = 0.0106  
Exact McNemar significance probability = 0.0161

Proportion with factor

| Cases      | .5357143 |                      |                  |
|------------|----------|----------------------|------------------|
| Controls   | .2857143 | [95% Conf. Interval] |                  |
| difference | .25      | .0519726             | .4480274         |
| ratio      | 1.875    | 1.148685             | 3.060565         |
| rel. diff. | .35      | .1336258             | .5663742         |
| odds ratio | 2.75     | 1.179154             | 7.143667 (exact) |

Both methods estimated the same odds ratio, and the 95% confidence intervals are similar. `clogit` produced a confidence interval of [1.22, 6.18], whereas `mcc` produced a confidence interval of [1.18, 7.14].



### Use of weights

With `clogit`, weights apply to groups as a whole, not to individual observations. For example, if there is a group in your dataset with a frequency weight of 3, there are a total of three groups in your sample with the same values of the dependent and independent variables as this one group. Weights must have the same value for all observations belonging to the same group; otherwise, an error message will be displayed.

### ➤ Example 3

We use the example from the above discussion of the `mcc` command. Here we have a total of 56 matched case–control groups, each with one case matched to one control. We had 8 matched pairs in which both the case and the control are exposed, 22 pairs in which the case is exposed and the control is unexposed, 8 pairs in which the case is unexposed and the control is exposed, and 18 pairs in which they are both unexposed.

With weights, it is easy to enter these data into Stata and run `clogit`.



```

. clear
. input id case exposed weight
 id case exposed weight
1. 1 1 1 8
2. 1 0 1 8
3. 2 1 1 22
4. 2 0 0 22
5. 3 1 0 8
6. 3 0 1 8
7. 4 1 0 18
8. 4 0 0 18
9. end

. clogit case exposed [w=weight], group(id) or
(frequency weights assumed)

Iteration 0: log likelihood = -35.425931
Iteration 1: log likelihood = -35.419283
Iteration 2: log likelihood = -35.419282

Conditional (fixed-effects) logistic regression Number of obs = 112
 LR chi2(1) = 6.79
 Prob > chi2 = 0.0091
 Pseudo R2 = 0.0875

Log likelihood = -35.419282

```

| case    | Odds Ratio | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|---------|------------|-----------|------|-------|----------------------|----------|
| exposed | 2.75       | 1.135369  | 2.45 | 0.014 | 1.224347             | 6.176763 |

◀

## Fixed-effects logit

The fixed-effects logit model can be written as

$$\Pr(y_{it} = 1 \mid \mathbf{x}_{it}) = F(\alpha_i + \mathbf{x}_{it}\boldsymbol{\beta})$$

where  $F$  is the cumulative logistic distribution

$$F(z) = \frac{\exp(z)}{1 + \exp(z)}$$

$i = 1, 2, \dots, n$  denotes the independent units (called “groups” by clogit), and  $t = 1, 2, \dots, T_i$  denotes the observations for the  $i$ th unit (group).

Fitting this model by using a full maximum-likelihood approach leads to difficulties, however. When  $T_i$  is fixed, the maximum likelihood estimates for  $\alpha_i$  and  $\boldsymbol{\beta}$  are inconsistent ([Andersen 1970](#); [Chamberlain 1980](#)). This difficulty can be circumvented by looking at the probability of  $\mathbf{y}_i = (y_{i1}, \dots, y_{iT_i})$  conditional on  $\sum_{t=1}^{T_i} y_{it}$ . This conditional probability does not involve the  $\alpha_i$ , so they are never estimated when the resulting conditional likelihood is used. See [Hamerle and Ronning \(1995\)](#) for a succinct and lucid development. See [Methods and formulas](#) for the estimation equation.

➤ Example 4

We are studying unionization of women in the United States by using the union dataset; see [XT] xt. We fit the fixed-effects logit model:

```
. use http://www.stata-press.com/data/r11/union, clear
(NLS Women 14-24 in 1968)

. clogit union age grade not_smsa south black, group(idcode)
note: multiple positive outcomes within groups encountered.
note: 2744 groups (14165 obs) dropped because of all positive or
 all negative outcomes.
note: black omitted because of no within-group variance.

Iteration 0: log likelihood = -4521.3385
Iteration 1: log likelihood = -4516.1404
Iteration 2: log likelihood = -4516.1385
Iteration 3: log likelihood = -4516.1385

Conditional (fixed-effects) logistic regression Number of obs = 12035
 LR chi2(4) = 68.09
 Prob > chi2 = 0.0000
Log likelihood = -4516.1385 Pseudo R2 = 0.0075
```

| union    | Coef.    | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|----------|-----------|-------|-------|----------------------|-----------|
| age      | .0170301 | .004146   | 4.11  | 0.000 | .0089042             | .0251561  |
| grade    | .0853572 | .0418781  | 2.04  | 0.042 | .0032777             | .1674368  |
| not_smsa | .0083678 | .1127963  | 0.07  | 0.941 | -.2127088            | .2294445  |
| south    | -.748023 | .1251752  | -5.98 | 0.000 | -.9933619            | -.5026842 |

We received three messages at the top of the output. The first one, “multiple positive outcomes within groups encountered”, we expected. Our data do indeed have multiple positive outcomes (union = 1) in many groups. (Here a group consists of all the observations for a particular individual.)

The second message tells us that 2,744 groups were “dropped” by clogit. When either union = 0 or union = 1 for all observations for an individual, this individual’s contribution to the log-likelihood is zero. Although these are perfectly valid observations in every sense, they have no effect on the estimation, so they are not included in the total “Number of obs”. Hence, the reported “Number of obs” gives the effective sample size of the estimation. Here it is 12,035 observations—only 46% of the total 26,200.

We can easily check that there are indeed 2,744 groups with union either all 0 or all 1. We will generate a variable that contains the fraction of observations for each individual who has union = 1.

```
. by idcode, sort: generate fraction = sum(union)/sum(union < .)
. by idcode: replace fraction = . if _n < _N
(21766 real changes made, 21766 to missing)
. tabulate fraction
```

| fraction                | Freq. | Percent | Cum.   |
|-------------------------|-------|---------|--------|
| 0                       | 2,481 | 55.95   | 55.95  |
| .0833333                | 30    | 0.68    | 56.63  |
| .0909091                | 33    | 0.74    | 57.37  |
| .1                      | 53    | 1.20    | 58.57  |
| <i>(output omitted)</i> |       |         |        |
| .9                      | 10    | 0.23    | 93.59  |
| .9090909                | 11    | 0.25    | 93.84  |
| .9166667                | 10    | 0.23    | 94.07  |
| 1                       | 263   | 5.93    | 100.00 |
| Total                   | 4,434 | 100.00  |        |

Because  $2481 + 263 = 2744$ , we confirm what `clogit` did.

The third warning message from `clogit` said “black omitted because of no within-group variance”. Obviously, race stays constant for an individual across time. Any such variables are collinear with the  $\alpha_i$  (i.e., the fixed effects), and just as the  $\alpha_i$  drop out of the conditional likelihood, so do all variables that are unchanging within groups. Thus they cannot be estimated with the conditional fixed-effects model.

There are several other estimators implemented in Stata that we could use with these data:

```
cloglog ... , vce(cluster idcode)
logit ... , vce(cluster idcode)
probit ... , vce(cluster idcode)
scobit ... , vce(cluster idcode)
xtcloglog ...
xtgee ... , family(binomial) link(logit) corr(exchangeable)
xtlogit ...
xtprobit ...
```

See [\[R\] cloglog](#), [\[R\] logit](#), [\[R\] probit](#), [\[R\] scobit](#), [\[XT\] xtcloglog](#), [\[XT\] xtgee](#), [\[XT\] xtlogit](#), and [\[XT\] xtprobit](#) for details.

◀

*(Continued on next page)*

## Saved results

clogit saves the following in `e()`:

### Scalars

|                              |                                                                                 |
|------------------------------|---------------------------------------------------------------------------------|
| <code>e(N)</code>            | number of observations                                                          |
| <code>e(N_drop)</code>       | number of observations dropped because of all positive or all negative outcomes |
| <code>e(N_group_drop)</code> | number of groups dropped because of all positive or all negative outcomes       |
| <code>e(k)</code>            | number of parameters                                                            |
| <code>e(k_eq)</code>         | number of equations in <code>e(b)</code>                                        |
| <code>e(k_eq_model)</code>   | number of equations in model Wald test                                          |
| <code>e(k_dv)</code>         | number of dependent variables                                                   |
| <code>e(k_autoCns)</code>    | number of base, empty, and omitted constraints                                  |
| <code>e(df_m)</code>         | model degrees of freedom                                                        |
| <code>e(r2_p)</code>         | pseudo- <i>R</i> -squared                                                       |
| <code>e(ll)</code>           | log likelihood                                                                  |
| <code>e(ll_0)</code>         | log likelihood, constant-only model                                             |
| <code>e(N_clust)</code>      | number of clusters                                                              |
| <code>e(chi2)</code>         | $\chi^2$                                                                        |
| <code>e(p)</code>            | significance                                                                    |
| <code>e(rank)</code>         | rank of <code>e(V)</code>                                                       |
| <code>e(ic)</code>           | number of iterations                                                            |
| <code>e(rc)</code>           | return code                                                                     |
| <code>e(converged)</code>    | 1 if converged, 0 otherwise                                                     |

### Macros

|                                 |                                                                                                     |
|---------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>clogit</code>                                                                                 |
| <code>e(cmdline)</code>         | command as typed                                                                                    |
| <code>e(depvar)</code>          | name of dependent variable                                                                          |
| <code>e(group)</code>           | name of <code>group()</code> variable                                                               |
| <code>e(multiple)</code>        | <code>multiple</code> if multiple positive outcomes within group                                    |
| <code>e(wtype)</code>           | weight type                                                                                         |
| <code>e(wexp)</code>            | weight expression                                                                                   |
| <code>e(title)</code>           | title in estimation output                                                                          |
| <code>e(clustvar)</code>        | name of cluster variable                                                                            |
| <code>e(offset)</code>          | offset                                                                                              |
| <code>e(chi2type)</code>        | LR; type of model $\chi^2$ test                                                                     |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                                                      |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                                                       |
| <code>e(opt)</code>             | type of optimization                                                                                |
| <code>e(which)</code>           | <code>max</code> or <code>min</code> ; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>       | type of <code>ml</code> method                                                                      |
| <code>e(user)</code>            | name of likelihood-evaluator program                                                                |
| <code>e(technique)</code>       | maximization technique                                                                              |
| <code>e(singularHmethod)</code> | <code>m-marquardt</code> or <code>hybrid</code> ; method used when Hessian is singular              |
| <code>e(crittype)</code>        | optimization criterion                                                                              |
| <code>e(properties)</code>      | <code>b</code> <code>V</code>                                                                       |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                                                      |
| <code>e(marginsok)</code>       | predictions allowed by <code>margins</code>                                                         |
| <code>e(marginsnotok)</code>    | predictions disallowed by <code>margins</code>                                                      |
| <code>e(marginsprop)</code>     | signals to the <code>margins</code> command                                                         |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>                                      |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>                                      |

## Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |
| <code>e(gradient)</code>     | gradient vector                              |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`clogit` is implemented as an ado-file.

Breslow and Day (1980, 247–279), Collett (2003, 251–267), and Hosmer Jr. and Lemeshow (2000, 223–259) provide a biostatistical point of view on conditional logistic regression. Hamerle and Ronning (1995) give a succinct and lucid review of fixed-effects logit; Chamberlain (1980) is a standard reference for this model. Greene (2008, chap. 23) provides a straightforward textbook description of conditional logistic regression from an economist’s point of view, as well as a brief description of choice models.

Let  $i = 1, 2, \dots, n$  denote the groups and let  $t = 1, 2, \dots, T_i$  denote the observations for the  $i$ th group. Let  $y_{it}$  be the dependent variable taking on values 0 or 1. Let  $\mathbf{y}_i = (y_{i1}, \dots, y_{iT_i})$  be the outcomes for the  $i$ th group as a whole. Let  $\mathbf{x}_{it}$  be a row vector of covariates. Let

$$k_{1i} = \sum_{t=1}^{T_i} y_{it}$$

be the observed number of ones for the dependent variable in the  $i$ th group. Biostatisticians would say that there are  $k_{1i}$  cases matched to  $k_{2i} = T_i - k_{1i}$  controls in the  $i$ th group.

We consider the probability of a possible value of  $\mathbf{y}_i$  conditional on  $\sum_{t=1}^{T_i} y_{it} = k_{1i}$  (Hamerle and Ronning 1995, eq. 8.33; Hosmer Jr. and Lemeshow 2000, eq. 7.4),

$$\Pr(\mathbf{y}_i \mid \sum_{t=1}^{T_i} y_{it} = k_{1i}) = \frac{\exp(\sum_{t=1}^{T_i} y_{it} \mathbf{x}_{it} \boldsymbol{\beta})}{\sum_{\mathbf{d}_i \in S_i} \exp(\sum_{t=1}^{T_i} d_{it} \mathbf{x}_{it} \boldsymbol{\beta})}$$

where  $d_{it}$  is equal to 0 or 1 with  $\sum_{t=1}^{T_i} d_{it} = k_{1i}$ , and  $S_i$  is the set of all possible combinations of  $k_{1i}$  ones and  $k_{2i}$  zeros. Clearly, there are  $\binom{T_i}{k_{1i}}$  such combinations, but we need not count all these combinations to compute the denominator of the above equation. It can be computed recursively.

Denote the denominator by

$$f_i(T_i, k_{1i}) = \sum_{\mathbf{d}_i \in S_i} \exp\left(\sum_{t=1}^{T_i} d_{it} \mathbf{x}_{it} \boldsymbol{\beta}\right)$$

Consider, computationally, how  $f_i$  changes as we go from a total of 1 observation in the group to 2 observations to 3, etc. Doing this, we derive the recursive formula

$$f_i(T, k) = f_i(T - 1, k) + f_i(T - 1, k - 1) \exp(\mathbf{x}_{iT} \boldsymbol{\beta})$$

where we define  $f_i(T, k) = 0$  if  $T < k$  and  $f_i(T, 0) = 1$ .

The conditional log-likelihood is

$$\ln L = \sum_{i=1}^n \left\{ \sum_{t=1}^{T_i} y_{it} \mathbf{x}_{it} \boldsymbol{\beta} - \log f_i(T_i, k_{1i}) \right\}$$

The derivatives of the conditional log-likelihood can also be computed recursively by taking derivatives of the recursive formula for  $f_i$ .

Computation time is roughly proportional to

$$p^2 \sum_{i=1}^n T_i \min(k_{1i}, k_{2i})$$

where  $p$  is the number of independent variables in the model. If  $\min(k_{1i}, k_{2i})$  is small, computation time is not an issue. But if it is large—say, 100 or more—patience may be required.

If  $T_i$  is large for all groups, the bias of the unconditional fixed-effects estimator is not a concern, and we can confidently use `logit` with an indicator variable for each group (provided, of course, that the number of groups does not exceed `matsize`; see [R] [matsize](#)).

This command supports the clustered version of the Huber/White/sandwich estimator of the variance using `vce(robust)` and `vce(cluster clustvar)`. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#). Specifying `vce(robust)` is equivalent to specifying `vce(cluster groupvar)`, where `groupvar` is the variable for the matched groups.

`clogit` also supports estimation with survey data. For details on VCEs with survey data, see [SVY] [variance estimation](#).

## References

- Andersen, E. B. 1970. Asymptotic properties of conditional maximum likelihood estimators. *Journal of the Royal Statistical Society, Series B* 32: 283–301.
- Breslow, N. E., and N. E. Day. 1980. *Statistical Methods in Cancer Research: Vol. 1—The Analysis of Case-Control Studies*. Lyon: IARC.
- Chamberlain, G. 1980. Analysis of covariance with qualitative data. *Review of Economic Studies* 47: 225–238.
- Collett, D. 2003. *Modelling Binary Data*. 2nd ed. London: Chapman & Hall/CRC.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice-Hall.
- Hamerle, A., and G. Ronning. 1995. Panel analysis for qualitative variables. In *Handbook of Statistical Modeling for the Social and Behavioral Sciences*, ed. G. Arminger, C. C. Clogg, and M. E. Sobel, 401–451. New York: Plenum.
- Hole, A. R. 2007. Fitting mixed logit models by using maximum simulated likelihood. *Stata Journal* 7: 388–401.
- Hosmer Jr., D. W., and S. Lemeshow. 2000. *Applied Logistic Regression*. 2nd ed. New York: Wiley.
- Kleinbaum, D. G., and M. Klein. 2002. *Logistic Regression: A Self-Learning Text*. 2nd ed. New York: Springer.
- Long, J. S., and J. Freese. 2006. *Regression Models for Categorical Dependent Variables Using Stata*. 2nd ed. College Station, TX: Stata Press.
- McFadden, D. L. 1974. Conditional logit analysis of qualitative choice behavior. In *Frontiers in Econometrics*, ed. P. Zarembka, 105–142. New York: Academic Press.

## Also see

- [R] **clogit postestimation** — Postestimation tools for clogit
- [R] **asclogit** — Alternative-specific conditional logit (McFadden’s choice) model
- [R] **logistic** — Logistic regression, reporting odds ratios
- [R] **mlogit** — Multinomial (polytomous) logistic regression
- [R] **nlogit** — Nested logit regression
- [R] **ologit** — Ordered logistic regression
- [R] **scobit** — Skewed logistic regression
- [SVY] **svy estimation** — Estimation commands for survey data
- [XT] **xtgee** — Fit population-averaged panel-data models by using GEE
- [XT] **xtlogit** — Fixed-effects, random-effects, and population-averaged logit models
- [U] **20 Estimation and postestimation commands**

Description

The following postestimation commands are available for `clogit`:

| command                           | description                                                                                         |
|-----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>                | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>          | postestimation statistics for survey data                                                           |
| <code>estimates</code>            | cataloging estimation results                                                                       |
| <code>hausman</code>              | Hausman's specification test                                                                        |
| <code>lincom</code>               | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>             | link test for model specification                                                                   |
| <code>lrtest</code> <sup>1</sup>  | likelihood-ratio test                                                                               |
| <code>margins</code> <sup>2</sup> | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>                | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>              | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>            | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>                | seemingly unrelated estimation                                                                      |
| <code>test</code>                 | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>               | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

<sup>2</sup> The default prediction statistic `pc1` cannot be correctly handled by `margins`; however, `margins` can be used after `clogit` with options `predict(pu0)` and `predict(xb)`.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.



## Syntax for predict

```
predict [type] newvar [if] [in] [, statistic nooffset]
```

| statistic | description |
|-----------|-------------|
|-----------|-------------|

|                          |                                                                            |
|--------------------------|----------------------------------------------------------------------------|
| Main                     |                                                                            |
| <code>pc1</code>         | probability of a positive outcome; the default                             |
| <code>pu0</code>         | probability of a positive outcome, assuming fixed effect is zero           |
| <code>xb</code>          | linear prediction                                                          |
| <code>stdp</code>        | standard error of the linear prediction                                    |
| * <code>dbeta</code>     | Delta- $\beta$ influence statistic                                         |
| * <code>dx2</code>       | Delta- $\chi^2$ lack-of-fit statistic                                      |
| * <code>gdbeta</code>    | Delta- $\beta$ influence statistic for each group                          |
| * <code>gdx2</code>      | Delta- $\chi^2$ lack-of-fit statistic for each group                       |
| * <code>hat</code>       | Hosmer and Lemeshow leverage                                               |
| * <code>residuals</code> | Pearson residuals                                                          |
| * <code>rstandard</code> | standardized Pearson residuals                                             |
| <code>score</code>       | first derivative of the log likelihood with respect to $\mathbf{x}_j\beta$ |

Unstarred statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample. Starred statistics are calculated only for the estimation sample, even when `if e(sample)` is not specified.

Starred statistics are available for multiple controls per case-matching design only. They are not available if `vce(robust)`, `vce(cluster clustvar)`, or `pweights` were specified with `clogit`.

`dbeta`, `dx2`, `gdbeta`, `gdx2`, `hat`, and `rstandard` are not available if `constraints()` was specified with `clogit`.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

`pc1`, the default, calculates the probability of a positive outcome conditional on one positive outcome within group.

`pu0` calculates the probability of a positive outcome, assuming that the fixed effect is zero.

`xb` calculates the linear prediction.

`stdp` calculates the standard error of the linear prediction.

`dbeta` calculates the Delta- $\beta$  influence statistic, a standardized measure of the difference in the coefficient vector that is due to deletion of the observation.

`dx2` calculates the Delta- $\chi^2$  influence statistic, reflecting the decrease in the Pearson chi-squared that is due to deletion of the observation.

`gdbeta` calculates the approximation to the Pregibon stratum-specific Delta- $\beta$  influence statistic, a standardized measure of the difference in the coefficient vector that is due to deletion of the entire stratum.

`gdx2` calculates the approximation to the Pregibon stratum-specific Delta- $\chi^2$  influence statistic, reflecting the decrease in the Pearson chi-squared that is due to deletion of the entire stratum.

`hat` calculates the Hosmer and Lemeshow leverage or the diagonal element of the hat matrix.

`residuals` calculates the Pearson residuals.

`rstandard` calculates the standardized Pearson residuals.

`score` calculates the equation-level score,  $\partial \ln L / \partial (\mathbf{x}_{it} \boldsymbol{\beta})$ .

`nooffset` is relevant only if you specified `offset(varname)` for `clogit`. It modifies the calculations made by `predict` so that they ignore the offset variable; the linear prediction is treated as  $\mathbf{x}_j \mathbf{b}$  rather than as  $\mathbf{x}_j \mathbf{b} + \text{offset}_j$ . This option cannot be specified with `dbeta`, `dx2`, `gdbeta`, `gdx2`, `hat`, and `rstandard`.

## Remarks

`predict` may be used after `clogit` to obtain predicted values of the index  $\mathbf{x}_{it} \boldsymbol{\beta}$ . Predicted probabilities for conditional logistic regression must be interpreted carefully. Probabilities are estimated for each group as a whole, not for individual observations. Furthermore, the probabilities are conditional on the number of positive outcomes in the group (i.e., the number of cases and the number of controls), or it is assumed that the fixed effect is zero. `predict` may also be used to obtain influence and lack of fit statistics for an individual observation and for the whole group, to compute Pearson, standardized Pearson residuals, and leverage values.

`predict` may be used for both within-sample and out-of-sample predictions.

## ► Example 1

Suppose that we have 1: $k_{2i}$  matched data and that we have previously fit the following model:

```
. use http://www.stata-press.com/data/r11/clogitid
. clogit y x1 x2, group(id)
(output omitted)
```

To obtain the predicted values of the index, we could type `predict idx, xb` to create a new variable called `idx`. From `idx`, we could then calculate the predicted probabilities. Easier, however, would be to type

```
. predict phat
(option p1 assumed; probability of success given one success within group)
```

`phat` would then contain the predicted probabilities.

As noted previously, the predicted probabilities are really predicted probabilities for the group as a whole (i.e., they are the predicted probability of observing  $y_{it} = 1$  and  $y_{it'} = 0$  for all  $t' \neq t$ ). Thus, if we want to obtain the predicted probabilities for the estimation sample, it is important that, when we make the calculation, predictions be restricted to the same sample on which we estimated the data. We cannot predict the probabilities and then just keep the relevant ones because the entire sample determines each probability. Thus, assuming that we are not attempting to make out-of-sample predictions, we type

```
. predict phat2 if e(sample)
(option p1 assumed; probability of success given one success within group)
```

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

Recall that  $i = 1, \dots, n$  denote the groups and  $t = 1, \dots, T_i$  denote the observations for the  $i$ th group.

`predict` produces probabilities of a positive outcome within group conditional on there being one positive outcome (`pc1`),

$$\Pr \left( y_{it} = 1 \mid \sum_{t=1}^{T_i} y_{it} = 1 \right) = \frac{\exp(\mathbf{x}_{it}\boldsymbol{\beta})}{\sum_{t=1}^{T_i} \exp(\mathbf{x}_{it}\boldsymbol{\beta})}$$

or `predict` calculates the unconditional `pu0`:

$$\Pr(y_{it} = 1) = \frac{\exp(\mathbf{x}_{it}\boldsymbol{\beta})}{1 + \exp(\mathbf{x}_{it}\boldsymbol{\beta})}$$

Let  $N = \sum_{j=1}^n T_j$  denote the total number of observations,  $p$  denote the number of covariates, and  $\hat{\theta}_{it}$  denote the conditional predicted probabilities of a positive outcome (`pc1`).

For the multiple control per case ( $1 : k_{2i}$ ) matching, [Hosmer Jr. and Lemeshow \(2000, 248–251\)](#) propose the following diagnostics:

The Pearson residual is

$$r_{it} = \frac{(y_{it} - \hat{\theta}_{it})}{\sqrt{\hat{\theta}_{it}}}$$

The leverage (hat) value is defined as

$$h_{it} = \hat{\theta}_{it} \tilde{\mathbf{x}}_{it}^T (\tilde{\mathbf{X}}^T \mathbf{U} \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{x}}_{it}$$

where  $\tilde{\mathbf{x}}_{it} = \mathbf{x}_{it} - \sum_{j=1}^{T_i} \mathbf{x}_{ij} \hat{\theta}_{ij}$  is the  $1 \times p$  row vector of centered by a weighted stratum-specific mean covariate values,  $\mathbf{U}_N = \text{diag}\{\hat{\theta}_{it}\}$ , and the rows of  $\tilde{\mathbf{X}}_{N \times p}$  are composed of  $\tilde{\mathbf{x}}_{it}$  values.

The standardized Pearson residual is

$$r_{sit} = \frac{r_{it}}{\sqrt{1 - h_{it}}}$$

The lack of fit and influence diagnostics for an individual observation are (respectively) computed as

$$\Delta \chi_{it}^2 = r_{sit}^2$$

and

$$\Delta \hat{\beta}_{it} = \Delta \chi_{it}^2 \frac{h_{it}}{1 - h_{it}}$$

The lack of fit and influence diagnostics for the groups are the group-specific totals of the respective individual diagnostics shown above.

## Reference

Hosmer Jr., D. W., and S. Lemeshow. 2000. *Applied Logistic Regression*. 2nd ed. New York: Wiley.

## Also see

[R] [clogit](#) — Conditional (fixed-effects) logistic regression

[U] [20 Estimation and postestimation commands](#)

Syntax

```
cloglog depvar [indepvars] [if] [in] [weight] [, options]
```

| <i>options</i>                            | description                                                                                                                             |
|-------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Model                                     |                                                                                                                                         |
| <u>noconstant</u>                         | suppress constant term                                                                                                                  |
| <u>offset</u> ( <i>varname</i> )          | include <i>varname</i> in model with coefficient constrained to 1                                                                       |
| <u>asis</u>                               | retain perfect predictor variables                                                                                                      |
| <u>constraints</u> ( <i>constraints</i> ) | apply specified linear constraints                                                                                                      |
| <u>collinear</u>                          | keep collinear variables                                                                                                                |
| SE/Robust                                 |                                                                                                                                         |
| <u>vce</u> ( <i>vcetype</i> )             | <i>vcetype</i> may be <u>oim</u> , <u>robust</u> , <u>cluster</u> <i>clustvar</i> , <u>opg</u> , <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting                                 |                                                                                                                                         |
| <u>level</u> (#)                          | set confidence level; default is <u>level</u> (95)                                                                                      |
| <u>eform</u>                              | report exponentiated coefficients                                                                                                       |
| <u>nocnsreport</u>                        | do not display constraints                                                                                                              |
| <u>display_options</u>                    | control spacing and display of omitted variables and base and empty cells                                                               |
| Maximization                              |                                                                                                                                         |
| <u>maximize_options</u>                   | control the maximization process; seldom used                                                                                           |
| <sup>†</sup> <u>coeflegend</u>            | display coefficients' legend instead of coefficient table                                                                               |

<sup>†</sup>coeflegend does not appear in the dialog box.  
*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.  
*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.  
bootstrap, by, jackknife, mi estimate, nestreg, rolling, statsby, stepwise, and svy are allowed; see [U] 11.1.10 Prefix commands.  
vce(bootstrap) and vce(jackknife) are not allowed with the mi estimate prefix.  
Weights are not allowed with the bootstrap prefix.  
vce() and weights are not allowed with the svy prefix.  
fweights, iweights, and pweights are allowed; see [U] 11.1.6 weight.  
See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Binary outcomes > Complementary log-log regression

## Description

`cloglog` fits maximum-likelihood complementary log-log models.

See [\[R\] logistic](#) for a list of related estimation commands.

## Options

### Model

`noconstant`, `offset(varname)`; see [\[R\] estimation options](#).

`asis` forces retention of perfect predictor variables and their associated perfectly predicted observations and may produce instabilities in maximization; see [\[R\] probit](#).

`constraints(constraints)`, `collinear`; see [\[R\] estimation options](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [\[R\] vce\\_option](#).

### Reporting

`level(#)`; see [\[R\] estimation options](#).

`eform` displays the exponentiated coefficients and corresponding standard errors and confidence intervals.

`nocnsreport`; see [\[R\] estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [\[R\] estimation options](#).

### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [\[R\] maximize](#). These options are seldom used.

Setting the optimization type to `technique(bhhh)` resets the default `vcetype` to `vce(opg)`.

The following option is available with `cloglog` but is not shown in the dialog box:

`coeflegend`; see [\[R\] estimation options](#).

## Remarks

Remarks are presented under the following headings:

*Introduction to complementary log-log regression*  
*Robust standard errors*

## Introduction to complementary log-log regression

cloglog fits maximum likelihood models with dichotomous dependent variables coded as 0/1 (or, more precisely, coded as 0 and not 0).

### ► Example 1

We have data on the make, weight, and mileage rating of 22 foreign and 52 domestic automobiles. We wish to fit a model explaining whether a car is foreign based on its weight and mileage. Here is an overview of our data:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
```

```
. keep make mpg weight foreign
. describe
```

```
Contains data from http://www.stata-press.com/data/r11/auto.dta
 obs: 74 1978 Automobile Data
 vars: 4 13 Apr 2009 17:45
 size: 1,998 (99.9% of memory free) (_dta has notes)
```

| variable name | storage<br>type | display<br>format | value<br>label | variable label |
|---------------|-----------------|-------------------|----------------|----------------|
| make          | str18           | %-18s             |                | Make and Model |
| mpg           | int             | %8.0g             |                | Mileage (mpg)  |
| weight        | int             | %8.0gc            |                | Weight (lbs.)  |
| foreign       | byte            | %8.0g             | origin         | Car type       |

```
Sorted by: foreign
```

```
Note: dataset has changed since last saved
```

```
. inspect foreign
```

```
foreign: Car type
```

|   |   |          | Number of Observations |          |             |
|---|---|----------|------------------------|----------|-------------|
|   |   |          | Total                  | Integers | Nonintegers |
| # |   | Negative | -                      | -        | -           |
| # |   | Zero     | 52                     | 52       | -           |
| # |   | Positive | 22                     | 22       | -           |
| # |   |          |                        |          |             |
| # | # | Total    | 74                     | 74       | -           |
| # | # | Missing  | -                      |          |             |
|   |   |          | 74                     |          |             |

0 1  
(2 unique values)

foreign is labeled and all values are documented in the label.

The variable `foreign` takes on two unique values, 0 and 1. The value 0 denotes a domestic car, and 1 denotes a foreign car.

The model that we wish to fit is

$$\Pr(\text{foreign} = 1) = F(\beta_0 + \beta_1 \text{weight} + \beta_2 \text{mpg})$$

where  $F(z) = 1 - \exp\{-\exp(z)\}$ .

To fit this model, we type

```
. cloglog foreign weight mpg
Iteration 0: log likelihood = -34.054593
Iteration 1: log likelihood = -27.869915
Iteration 2: log likelihood = -27.742997
Iteration 3: log likelihood = -27.742769
Iteration 4: log likelihood = -27.742769

Complementary log-log regression Number of obs = 74
 Zero outcomes = 52
 Nonzero outcomes = 22
 LR chi2(2) = 34.58
 Prob > chi2 = 0.0000

Log likelihood = -27.742769
```

| foreign | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|---------|-----------|-----------|-------|-------|----------------------|-----------|
| weight  | -.0029153 | .0006974  | -4.18 | 0.000 | -.0042823            | -.0015483 |
| mpg     | -.1422911 | .076387   | -1.86 | 0.062 | -.2920069            | .0074247  |
| _cons   | 10.09694  | 3.351841  | 3.01  | 0.003 | 3.527448             | 16.66642  |

We find that heavier cars are less likely to be foreign and that cars yielding better gas mileage are also less likely to be foreign, at least when holding the weight of the car constant.

See [R] [maximize](#) for an explanation of the output.



❑ Technical note

Stata interprets a value of 0 as a negative outcome (failure) and treats all other values (except missing) as positive outcomes (successes). Thus, if your dependent variable takes on the values 0 and 1, 0 is interpreted as failure and 1 as success. If your dependent variable takes on the values 0, 1, and 2, 0 is still interpreted as failure, but both 1 and 2 are treated as successes.

If you prefer a more formal mathematical statement, when you type `cloglog y x`, Stata fits the model

$$\Pr(y_j \neq 0 \mid \mathbf{x}_j) = 1 - \exp\left\{-\exp(\mathbf{x}_j\boldsymbol{\beta})\right\}$$



Robust standard errors

If you specify the `vce(robust)` option, `cloglog` reports robust standard errors, as described in [U] [20.16 Obtaining robust variance estimates](#). For the model of `foreign` on `weight` and `mpg`, the robust calculation increases the standard error of the coefficient on `mpg` by 44%:



```
. cloglog foreign weight mpg, vce(robust)
```

Iteration 0: log pseudolikelihood = -34.054593  
Iteration 1: log pseudolikelihood = -27.869915  
Iteration 2: log pseudolikelihood = -27.742997  
Iteration 3: log pseudolikelihood = -27.742769  
Iteration 4: log pseudolikelihood = -27.742769

Complementary log-log regression

|                  |   |        |
|------------------|---|--------|
| Number of obs    | = | 74     |
| Zero outcomes    | = | 52     |
| Nonzero outcomes | = | 22     |
| Wald chi2(2)     | = | 29.74  |
| Prob > chi2      | = | 0.0000 |

Log pseudolikelihood = -27.742769

| foreign | Coef.     | Robust<br>Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|---------|-----------|---------------------|-------|-------|----------------------|-----------|
| weight  | -.0029153 | .0007484            | -3.90 | 0.000 | -.0043822            | -.0014484 |
| mpg     | -.1422911 | .1102466            | -1.29 | 0.197 | -.3583704            | .0737882  |
| _cons   | 10.09694  | 4.317305            | 2.34  | 0.019 | 1.635174             | 18.5587   |

Without `vce(robust)`, the standard error for the coefficient on `mpg` was reported to be 0.076, with a resulting confidence interval of  $[-0.29, 0.01]$ .

The `vce(cluster clustvar)` option can relax the independence assumption required by the complementary log-log estimator to being just independence between clusters. To demonstrate this ability, we will switch to a different dataset.

We are studying unionization of women in the United States by using the `union` dataset; see [\[XT\] xt](#). We fit the following model, ignoring that women are observed an average of 5.9 times each in this dataset:

```
. use http://www.stata-press.com/data/r11/union, clear
(NLS Women 14-24 in 1968)
```

```
. cloglog union age grade not_smsa south##c.year
```

Iteration 0: log likelihood = -13606.373  
Iteration 1: log likelihood = -13540.726  
Iteration 2: log likelihood = -13540.607  
Iteration 3: log likelihood = -13540.607

Complementary log-log regression

|                  |   |        |
|------------------|---|--------|
| Number of obs    | = | 26200  |
| Zero outcomes    | = | 20389  |
| Nonzero outcomes | = | 5811   |
| LR chi2(6)       | = | 647.24 |
| Prob > chi2      | = | 0.0000 |

Log likelihood = -13540.607

| union        | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|--------------|-----------|-----------|-------|-------|----------------------|-----------|
| age          | .0185346  | .0043616  | 4.25  | 0.000 | .009986              | .0270833  |
| grade        | .0452772  | .0057125  | 7.93  | 0.000 | .0340809             | .0564736  |
| not_smsa     | -.1886592 | .0317801  | -5.94 | 0.000 | -.2509471            | -.1263712 |
| 1.south      | -1.422292 | .3949381  | -3.60 | 0.000 | -2.196356            | -.648227  |
| year         | -.0133007 | .0049576  | -2.68 | 0.007 | -.0230174            | -.0035839 |
| south#c.year |           |           |       |       |                      |           |
| 1            | .0105659  | .0049234  | 2.15  | 0.032 | .0009161             | .0202157  |
| _cons        | -1.219801 | .2952374  | -4.13 | 0.000 | -1.798455            | -.6411462 |

The reported standard errors in this model are probably meaningless. Women are observed repeatedly, and so the observations are not independent. Looking at the coefficients, we find a large southern effect against unionization and a different time trend for the south. The `vce(cluster clustvar)` option provides a way to fit this model and obtains correct standard errors:

```
. cloglog union age grade not_smsa south##c.year, vce(cluster id) nolog
Complementary log-log regression Number of obs = 26200
 Zero outcomes = 20389
 Nonzero outcomes = 5811
 Wald chi2(6) = 160.76
Log pseudolikelihood = -13540.607 Prob > chi2 = 0.0000
 (Std. Err. adjusted for 4434 clusters in idcode)
```

| union        | Coef.     | Robust Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|--------------|-----------|------------------|-------|-------|----------------------|-----------|
| age          | .0185346  | .0084873         | 2.18  | 0.029 | .0018999             | .0351694  |
| grade        | .0452772  | .0125776         | 3.60  | 0.000 | .0206255             | .069929   |
| not_smsa     | -.1886592 | .0642068         | -2.94 | 0.003 | -.3145021            | -.0628162 |
| 1.south      | -1.422292 | .506517          | -2.81 | 0.005 | -2.415047            | -.4295365 |
| year         | -.0133007 | .0090628         | -1.47 | 0.142 | -.0310633            | .004462   |
| south#c.year |           |                  |       |       |                      |           |
| 1            | .0105659  | .0063175         | 1.67  | 0.094 | -.0018162            | .022948   |
| _cons        | -1.219801 | .5175129         | -2.36 | 0.018 | -2.234107            | -.2054942 |

These standard errors are larger than those reported by the inappropriate conventional calculation. By comparison, another way we could fit this model is with an equal-correlation population-averaged complementary log-log model:

```
. xtcloglog union age grade not_smsa south##c.year, pa nolog
GEE population-averaged model
Group variable: idcode Number of obs = 26200
Link: cloglog Number of groups = 4434
Family: binomial Obs per group: min = 1
Correlation: exchangeable avg = 5.9
 max = 12
 Wald chi2(6) = 234.66
Scale parameter: 1 Prob > chi2 = 0.0000
```

| union        | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|--------------|-----------|-----------|-------|-------|----------------------|-----------|
| age          | .0153737  | .0081156  | 1.89  | 0.058 | -.0005326            | .03128    |
| grade        | .0549518  | .0095093  | 5.78  | 0.000 | .0363139             | .0735897  |
| not_smsa     | -.1045232 | .0431082  | -2.42 | 0.015 | -.1890138            | -.0200326 |
| 1.south      | -1.714868 | .3384558  | -5.07 | 0.000 | -2.378229            | -1.051507 |
| year         | -.0115881 | .0084125  | -1.38 | 0.168 | -.0280763            | .0049001  |
| south#c.year |           |           |       |       |                      |           |
| 1            | .0149796  | .0041687  | 3.59  | 0.000 | .0068091             | .0231501  |
| _cons        | -1.488278 | .4468005  | -3.33 | 0.001 | -2.363991            | -.6125652 |

The coefficient estimates are similar, but these standard errors are smaller than those produced by `cloglog`, `vce(cluster clustvar)`. This finding is as we would expect. If the within-panel correlation assumptions are valid, the population-averaged estimator should be more efficient.

In addition to this estimator, we may use the `xtgee` command to fit a panel estimator (with complementary log-log link) and any number of assumptions on the within-idcode correlation.

`cloglog`, `vce(cluster clustvar)` is robust to assumptions about within-cluster correlation. That is, it inefficiently sums within cluster for the standard-error calculation rather than attempting to exploit what might be assumed about the within-cluster correlation (as do the `xtgee` population-averaged models).

## Saved results

`cloglog` saves the following in `e()`:

### Scalars

|                            |                                                |
|----------------------------|------------------------------------------------|
| <code>e(N)</code>          | number of observations                         |
| <code>e(k)</code>          | number of parameters                           |
| <code>e(k_eq)</code>       | number of equations in <code>e(b)</code>       |
| <code>e(k_eq_model)</code> | number of equations in model Wald test         |
| <code>e(k_dv)</code>       | number of dependent variables                  |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints |
| <code>e(N_f)</code>        | number of zero outcomes                        |
| <code>e(N_s)</code>        | number of nonzero outcomes                     |
| <code>e(df_m)</code>       | model degrees of freedom                       |
| <code>e(ll)</code>         | log likelihood                                 |
| <code>e(ll_0)</code>       | log likelihood, constant-only model            |
| <code>e(N_clust)</code>    | number of clusters                             |
| <code>e(chi2)</code>       | $\chi^2$                                       |
| <code>e(p)</code>          | significance                                   |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                      |
| <code>e(ic)</code>         | number of iterations                           |
| <code>e(rc)</code>         | return code                                    |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                    |

*(Continued on next page)*

**Macros**

|                                 |                                                                          |
|---------------------------------|--------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>cloglog</code>                                                     |
| <code>e(cmdline)</code>         | command as typed                                                         |
| <code>e(depvar)</code>          | name of dependent variable                                               |
| <code>e(wtype)</code>           | weight type                                                              |
| <code>e(wexp)</code>            | weight expression                                                        |
| <code>e(title)</code>           | title in estimation output                                               |
| <code>e(clustvar)</code>        | name of cluster variable                                                 |
| <code>e(offset)</code>          | offset                                                                   |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                  |
| <code>e(vce)</code>             | <code>vcetype</code> specified in <code>vce()</code>                     |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                            |
| <code>e(opt)</code>             | type of optimization                                                     |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>       | type of ml method                                                        |
| <code>e(user)</code>            | name of likelihood-evaluator program                                     |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular              |
| <code>e(technique)</code>       | maximization technique                                                   |
| <code>e(crittype)</code>        | optimization criterion                                                   |
| <code>e(properties)</code>      | <code>b V</code>                                                         |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                           |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>           |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>           |

**Matrices**

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(gradient)</code>     | gradient vector                              |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

**Functions**

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`cloglog` is implemented as an ado-file.

Complementary log-log analysis (related to the `gompit` model, so named because of its relationship to the Gompertz distribution) is an alternative to logit and probit analysis, but it is unlike these other estimators in that the transformation is not symmetric. Typically, this model is used when the positive (or negative) outcome is rare.

The log-likelihood function for complementary log-log is

$$\ln L = \sum_{j \in S} w_j \ln F(\mathbf{x}_j \mathbf{b}) + \sum_{j \notin S} w_j \ln \{1 - F(\mathbf{x}_j \mathbf{b})\}$$

where  $S$  is the set of all observations  $j$  such that  $y_j \neq 0$ ,  $F(z) = 1 - \exp\{-\exp(z)\}$ , and  $w_j$  denotes the optional weights.  $\ln L$  is maximized as described in [R] [maximize](#).

We can fit a gompit model by reversing the success–failure sense of the dependent variable and using `cloglog`.

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in *Maximum likelihood estimators* and *Methods and formulas*. The scores are calculated as  $\mathbf{u}_j = [\exp(\mathbf{x}_j\mathbf{b}) \exp\{-\exp(\mathbf{x}_j\mathbf{b})\}/F(\mathbf{x}_j\mathbf{b})]\mathbf{x}_j$  for the positive outcomes and  $\{-\exp(\mathbf{x}_j\mathbf{b})\}\mathbf{x}_j$  for the negative outcomes.

`cloglog` also supports estimation with survey data. For details on VCEs with survey data, see [SVY] [variance estimation](#).

## Acknowledgment

We thank Joseph Hilbe of Arizona State University for providing the inspiration for the `cloglog` command (Hilbe 1996, 1998).

## References

- Clayton, D. G., and M. Hills. 1993. *Statistical Models in Epidemiology*. Oxford: Oxford University Press.
- Hilbe, J. M. 1996. [sg53: Maximum-likelihood complementary log-log regression](#). *Stata Technical Bulletin* 32: 19–20. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 129–131. College Station, TX: Stata Press.
- . 1998. [sg53.2: Stata-like commands for complementary log-log regression](#). *Stata Technical Bulletin* 41: 23. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 166–167. College Station, TX: Stata Press.
- Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.
- Long, J. S., and J. Freese. 2006. *Regression Models for Categorical Dependent Variables Using Stata*. 2nd ed. College Station, TX: Stata Press.
- Xu, J., and J. S. Long. 2005. [Confidence intervals for predicted outcomes in regression models for categorical outcomes](#). *Stata Journal* 5: 537–559.

## Also see

- [R] [cloglog postestimation](#) — Postestimation tools for `cloglog`
- [R] [clogit](#) — Conditional (fixed-effects) logistic regression
- [R] [glm](#) — Generalized linear models
- [R] [logistic](#) — Logistic regression, reporting odds ratios
- [R] [scobit](#) — Skewed logistic regression
- [SVY] [svy estimation](#) — Estimation commands for survey data
- [XT] [xtcloglog](#) — Random-effects and population-averaged `cloglog` models
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `cloglog`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>            | link test for model specification                                                                   |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic nooffset]
```

| statistic          | description                                                       |
|--------------------|-------------------------------------------------------------------|
| Main               |                                                                   |
| <code>pr</code>    | probability of a positive outcome; the default                    |
| <code>xb</code>    | linear prediction                                                 |
| <code>stdp</code>  | standard error of the linear prediction                           |
| <code>score</code> | first derivative of the log likelihood with respect to $x_j\beta$ |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

**pr**, the default, calculates the probability of a positive outcome.

**xb** calculates the linear prediction.

**stdp** calculates the standard error of the linear prediction.

**score** calculates the equation-level score,  $\partial \ln L / \partial (\mathbf{x}_j \boldsymbol{\beta})$ .

**nooffset** is relevant only if you specified **offset**(*varname*) for **cloglog**. It modifies the calculations made by **predict** so that they ignore the offset variable; the linear prediction is treated as  $\mathbf{x}_j \mathbf{b}$  rather than as  $\mathbf{x}_j \mathbf{b} + \text{offset}_j$ .

## Remarks

Once you have fit a model, you can obtain the predicted probabilities by using the **predict** command for both the estimation sample and other samples; see [U] 20 [Estimation and postestimation commands](#) and [R] [predict](#). Here we will make only a few comments.

**predict** without arguments calculates the predicted probability of a positive outcome. With the **xb** option, it calculates the linear combination  $\mathbf{x}_j \mathbf{b}$ , where  $\mathbf{x}_j$  are the independent variables in the  $j$ th observation and  $\mathbf{b}$  is the estimated parameter vector.

With the **stdp** option, **predict** calculates the standard error of the linear prediction, which is not adjusted for replicated covariate patterns in the data.

## ► Example 1

In [example 1](#) in [R] [cloglog](#), we fit the complementary log-log model **cloglog** foreign weight mpg. To obtain predicted probabilities,

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. cloglog foreign weight mpg
(output omitted)
. predict p
(option pr assumed; Pr(foreign))
. summarize foreign p
```

| Variable | Obs | Mean     | Std. Dev. | Min      | Max      |
|----------|-----|----------|-----------|----------|----------|
| foreign  | 74  | .2972973 | .4601885  | 0        | 1        |
| p        | 74  | .2928348 | .29732    | .0032726 | .9446067 |

◀

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [cloglog](#) — Complementary log-log regression

[U] [20 Estimation and postestimation commands](#)



Syntax

```
cnsreg depvar indepvars [if] [in] [weight] , cconstraints(constraints) [options]
```

| <i>options</i>                               | description                                                                                                                  |
|----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| Model                                        |                                                                                                                              |
| * <u>c</u> constraints( <i>constraints</i> ) | apply specified linear constraints                                                                                           |
| <u>n</u> o <u>c</u> onstant                  | suppress constant term                                                                                                       |
| SE/Robust                                    |                                                                                                                              |
| vce( <i>vcetype</i> )                        | <i>vcetype</i> may be <u>ols</u> , <u>r</u> obust, <u>c</u> luster <i>clustvar</i> , <u>b</u> ootstrap, or <u>j</u> ackknife |
| Reporting                                    |                                                                                                                              |
| <u>l</u> evel(#)                             | set confidence level; default is <u>l</u> evel(95)                                                                           |
| <u>n</u> o <u>c</u> nsreport                 | do not display constraints                                                                                                   |
| <i>display_options</i>                       | control spacing and display of omitted variables and base and empty cells                                                    |
| <sup>†</sup> <u>m</u> se1                    | force MSE to be 1                                                                                                            |
| <sup>†</sup> <u>c</u> oef <u>l</u> egend     | display coefficients' legend instead of coefficient table                                                                    |

\*cconstraints(*constraints*) is required.

<sup>†</sup>mse1 and coeflegend do not appear in the dialog.

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

bootstrap, by, jackknife, mi estimate, rolling, statsby, and svy are allowed; see [U] 11.1.10 Prefix commands.

vce(bootstrap) and vce(jackknife) are not allowed with the mi estimate prefix.

Weights are not allowed with the bootstrap prefix.

aweights are not allowed with the jackknife prefix.

vce(), mse1, and weights are not allowed with the svy prefix.

aweights, fweights, pweights, and iweights are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Linear models and related > Constrained linear regression

Description

cnsreg fits constrained linear regression models. **cnsreg** typed without arguments redisplayes the previous **cnsreg** results.

## Options

### Model

`constraints(constraints)`, `noconstant`; see [R] [estimation options](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

`vce(ols)`, the default, uses the standard variance estimator for ordinary least-squares regression.

### Reporting

`level(#)`; see [R] [estimation options](#).

`nocnsreport`; see [R] [estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

The following options are available with `cnsreg` but are not shown in the dialog box:

`mse1` is used only in programs and ado-files that use `cnsreg` to fit models other than constrained linear regression. `mse1` sets the mean squared error to 1, thus forcing the variance–covariance matrix of the estimators to be  $(\mathbf{X}'\mathbf{D}\mathbf{X})^{-1}$  (see [Methods and formulas](#) in [R] [regress](#)) and affecting calculated standard errors. Degrees of freedom for  $t$  statistics are calculated as  $n$  rather than  $n - p + c$ , where  $p$  is the total number of parameters (prior to restrictions and including the constant) and  $c$  is the number of constraints.

`mse1` is not allowed with the `svy` prefix.

`coeflegend`; see [R] [estimation options](#).

## Remarks

For a discussion of constrained linear regression, see [Greene \(2008, 88\)](#); [Hill, Griffiths, and Lim \(2008, 146–148\)](#); or [Davidson and MacKinnon \(1993, 17\)](#).

### ► Example 1

In principle, we can obtain constrained linear regression estimates by modifying the list of independent variables. For instance, if we wanted to fit the model

$$\text{mpg} = \beta_0 + \beta_1 \text{price} + \beta_2 \text{weight} + u$$

and constrain  $\beta_1 = \beta_2$ , we could write

$$\text{mpg} = \beta_0 + \beta_1 (\text{price} + \text{weight}) + u$$

and run a regression of `mpg` on `price + weight`. The estimated coefficient on the sum would be the constrained estimate of  $\beta_1$  and  $\beta_2$ . Using `cnsreg`, however, is easier:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. constraint 1 price = weight
. cnsreg mpg price weight, constraint(1)
Constrained linear regression
```

```
Number of obs = 74
F(1, 72) = 37.59
Prob > F = 0.0000
Root MSE = 4.722
```

```
(1) price - weight = 0
```

| mpg    | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|--------|-----------|-----------|-------|-------|----------------------|-----------|
| price  | -.0009875 | .0001611  | -6.13 | 0.000 | -.0013086            | -.0006664 |
| weight | -.0009875 | .0001611  | -6.13 | 0.000 | -.0013086            | -.0006664 |
| _cons  | 30.36718  | 1.577958  | 19.24 | 0.000 | 27.22158             | 33.51278  |

We define constraints by using the `constraint` command; see [\[R\] constraint](#). We fit the model with `cnsreg` and specify the constraint number or numbers in the `constraints()` option.

Just to show that the results above are correct, here is the result of applying the constraint by hand:

```
. generate x = price + weight
. regress mpg x
```

| Source   | SS         | df | MS         |                 |        |  |
|----------|------------|----|------------|-----------------|--------|--|
| Model    | 838.065767 | 1  | 838.065767 | Number of obs = | 74     |  |
| Residual | 1605.39369 | 72 | 22.2971346 | F( 1, 72) =     | 37.59  |  |
|          |            |    |            | Prob > F =      | 0.0000 |  |
|          |            |    |            | R-squared =     | 0.3430 |  |
|          |            |    |            | Adj R-squared = | 0.3339 |  |
| Total    | 2443.45946 | 73 | 33.4720474 | Root MSE =      | 4.722  |  |

| mpg   | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|-------|-----------|-----------|-------|-------|----------------------|-----------|
| x     | -.0009875 | .0001611  | -6.13 | 0.000 | -.0013086            | -.0006664 |
| _cons | 30.36718  | 1.577958  | 19.24 | 0.000 | 27.22158             | 33.51278  |

◀

## ► Example 2

Models can be fit subject to multiple simultaneous constraints. We simply define the constraints and then include the constraint numbers in the `constraints()` option. For instance, say that we wish to fit the model

$$\text{mpg} = \beta_0 + \beta_1 \text{price} + \beta_2 \text{weight} + \beta_3 \text{displ} + \beta_4 \text{gear\_ratio} + \beta_5 \text{foreign} + \beta_6 \text{length} + u$$

subject to the constraints

$$\begin{aligned}\beta_1 &= \beta_2 = \beta_3 = \beta_6 \\ \beta_4 &= -\beta_5 = \beta_0/20\end{aligned}$$

(This model, like the one in example 1, is admittedly senseless.) We fit the model by typing

```
. constraint 1 price=weight
. constraint 2 displ=weight
. constraint 3 length=weight
```

```
. constraint 5 gear_ratio = -foreign
. constraint 6 gear_ratio = _cons/20
. cnsreg mpg price weight displ gear_ratio foreign length, c(1-3,5-6)
Constrained linear regression Number of obs = 74
 F(2, 72) = 785.20
 Prob > F = 0.0000
 Root MSE = 4.6823

(1) price - weight = 0
(2) - weight + displacement = 0
(3) - weight + length = 0
(4) gear_ratio + foreign = 0
(5) gear_ratio - .05 _cons = 0
```

| mpg          | Coef.     | Std. Err. | t      | P> t  | [95% Conf. Interval] |           |
|--------------|-----------|-----------|--------|-------|----------------------|-----------|
| price        | -.000923  | .0001534  | -6.02  | 0.000 | -.0012288            | -.0006172 |
| weight       | -.000923  | .0001534  | -6.02  | 0.000 | -.0012288            | -.0006172 |
| displacement | -.000923  | .0001534  | -6.02  | 0.000 | -.0012288            | -.0006172 |
| gear_ratio   | 1.326114  | .0687589  | 19.29  | 0.000 | 1.189046             | 1.463183  |
| foreign      | -1.326114 | .0687589  | -19.29 | 0.000 | -1.463183            | -1.189046 |
| length       | -.000923  | .0001534  | -6.02  | 0.000 | -.0012288            | -.0006172 |
| _cons        | 26.52229  | 1.375178  | 19.29  | 0.000 | 23.78092             | 29.26365  |

There are many ways we could have specified the `constraints()` option (which we abbreviated `c()` above). We typed `c(1-3,5-6)`, meaning that we want constraints 1 through 3 and 5 and 6; those numbers correspond to the constraints we defined. The only reason we did not use the number 4 was to emphasize that constraints do not have to be consecutively numbered. We typed `c(1-3,5-6)`, but we could have typed `c(1,2,3,5,6)` or `c(1-3,5,6)` or `c(1-2,3,5,6)` or even `c(1-6)`, which would have worked as long as constraint 4 was not defined. If we had previously defined a constraint 4, then `c(1-6)` would have included it.



## Saved results

`cnsreg` saves the following in `e()`:

| Scalars                   |                                                |
|---------------------------|------------------------------------------------|
| <code>e(N)</code>         | number of observations                         |
| <code>e(k_autoCns)</code> | number of base, empty, and omitted constraints |
| <code>e(df_m)</code>      | model degrees of freedom                       |
| <code>e(df_r)</code>      | residual degrees of freedom                    |
| <code>e(F)</code>         | <i>F</i> statistic                             |
| <code>e(rmse)</code>      | root mean squared error                        |
| <code>e(ll)</code>        | log likelihood                                 |
| <code>e(N_clust)</code>   | number of clusters                             |
| <code>e(rank)</code>      | rank of <code>e(V)</code>                      |

## Macros

|                            |                                                                |
|----------------------------|----------------------------------------------------------------|
| <code>e(cmd)</code>        | <code>cnsreg</code>                                            |
| <code>e(cmdline)</code>    | command as typed                                               |
| <code>e(depvar)</code>     | name of dependent variable                                     |
| <code>e(wtype)</code>      | weight type                                                    |
| <code>e(wexp)</code>       | weight expression                                              |
| <code>e(title)</code>      | title in estimation output                                     |
| <code>e(clustvar)</code>   | name of cluster variable                                       |
| <code>e(vce)</code>        | <i>vcetype</i> specified in <code>vce()</code>                 |
| <code>e(vcetype)</code>    | title used to label Std. Err.                                  |
| <code>e(properties)</code> | <code>b V</code>                                               |
| <code>e(predict)</code>    | program used to implement <code>predict</code>                 |
| <code>e(asbalanced)</code> | factor variables <code>fvset</code> as <code>asbalanced</code> |
| <code>e(asobserved)</code> | factor variables <code>fvset</code> as <code>asobserved</code> |

## Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`cnsreg` is implemented as an ado-file.

Let  $n$  be the number of observations,  $p$  be the total number of parameters (prior to restrictions and including the constant), and  $c$  be the number of constraints. The coefficients are calculated as  $\mathbf{b}' = \mathbf{T}'\{(\mathbf{T}'\mathbf{X}'\mathbf{W}\mathbf{X}\mathbf{T})^{-1}(\mathbf{T}'\mathbf{X}'\mathbf{W}\mathbf{y} - \mathbf{T}'\mathbf{X}'\mathbf{W}\mathbf{X}\mathbf{a}')\} + \mathbf{a}'$ , where  $\mathbf{T}$  and  $\mathbf{a}$  are as defined in [P] [makecns](#).  $\mathbf{W} = \mathbf{I}$  if no weights are specified. If weights are specified, let  $\mathbf{v}$ :  $1 \times n$  be the specified weights. If `fweight` frequency weights are specified,  $\mathbf{W} = \text{diag}(\mathbf{v})$ . If `aweight` analytic weights are specified, then  $\mathbf{W} = \text{diag}[\mathbf{v}/(\mathbf{1}'\mathbf{v})(\mathbf{1}'\mathbf{1})]$ , meaning that the weights are normalized to sum to the number of observations.

The mean squared error is  $s^2 = (\mathbf{y}'\mathbf{W}\mathbf{y} - 2\mathbf{b}'\mathbf{X}'\mathbf{W}\mathbf{y} + \mathbf{b}'\mathbf{X}'\mathbf{W}\mathbf{X}\mathbf{b})/(n - p + c)$ . The variance–covariance matrix is  $s^2\mathbf{T}(\mathbf{T}'\mathbf{X}'\mathbf{W}\mathbf{X}\mathbf{T})^{-1}\mathbf{T}'$ .

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Introduction](#) and [Methods and formulas](#).

`cnsreg` also supports estimation with survey data. For details on VCEs with survey data, see [SVY] variance estimation.

## References

- Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- Hill, R. C., W. E. Griffiths, and G. C. Lim. 2008. *Principles of Econometrics*. 3rd ed. Hoboken, NJ: Wiley.

## Also see

- [R] [cnsreg postestimation](#) — Postestimation tools for cnsreg
- [R] [regress](#) — Linear regression
- [SVY] [svy estimation](#) — Estimation commands for survey data
- [U] [20 Estimation and postestimation commands](#)

## Description

The following postestimation commands are available for `cnsreg`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>            | link test for model specification                                                                   |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

## Syntax for predict

`predict` [*type*] *newvar* [*if*] [*in*] [, *statistic*]

| <i>statistic</i>                      | description                                 |
|---------------------------------------|---------------------------------------------|
| Main                                  |                                             |
| <code>xb</code>                       | linear prediction; the default              |
| <code>_residuals</code>               | residuals                                   |
| <code>stdp</code>                     | standard error of the prediction            |
| <code>stdf</code>                     | standard error of the forecast              |
| <code>pr(<i>a</i>,<i>b</i>)</code>    | $\Pr(a < y_j < b)$                          |
| <code>e(<i>a</i>,<i>b</i>)</code>     | $E(y_j   a < y_j < b)$                      |
| <code>ystar(<i>a</i>,<i>b</i>)</code> | $E(y_j^*), y_j^* = \max\{a, \min(y_j, b)\}$ |
| <code>_score</code>                   | equivalent to <code>_residuals</code>       |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

`stdf` is not allowed with `svy` estimation results.

where  $a$  and  $b$  may be numbers or variables;  $a$  missing ( $a \geq .$ ) means  $-\infty$ , and  $b$  missing ( $b \geq .$ ) means  $+\infty$ ; see [U] [12.2.1 Missing values](#).

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

### Main

`xb`, the default, calculates the linear prediction.

`residuals` calculates the residuals, that is,  $y_j - \mathbf{x}_j\mathbf{b}$ .

`stdp` calculates the standard error of the prediction, which can be thought of as the standard error of the predicted expected value or mean for the observation's covariate pattern. The standard error of the prediction is also referred to as the standard error of the fitted value.

`stdf` calculates the standard error of the forecast, which is the standard error of the point prediction for 1 observation. It is commonly referred to as the standard error of the future or forecast value. By construction, the standard errors produced by `stdf` are always larger than those produced by `stdp`; see [Methods and formulas](#) in [R] [regress](#).

`pr(a,b)` calculates  $\Pr(a < \mathbf{x}_j\mathbf{b} + u_j < b)$ , the probability that  $y_j|\mathbf{x}_j$  would be observed in the interval  $(a, b)$ .

$a$  and  $b$  may be specified as numbers or variable names;  $lb$  and  $ub$  are variable names;

`pr(20,30)` calculates  $\Pr(20 < \mathbf{x}_j\mathbf{b} + u_j < 30)$ ;

`pr(lb,ub)` calculates  $\Pr(lb < \mathbf{x}_j\mathbf{b} + u_j < ub)$ ; and

`pr(20,ub)` calculates  $\Pr(20 < \mathbf{x}_j\mathbf{b} + u_j < ub)$ .

$a$  missing ( $a \geq .$ ) means  $-\infty$ ; `pr(.,30)` calculates  $\Pr(-\infty < \mathbf{x}_j\mathbf{b} + u_j < 30)$ ;

`pr(lb,30)` calculates  $\Pr(-\infty < \mathbf{x}_j\mathbf{b} + u_j < 30)$  in observations for which  $lb \geq .$  and calculates  $\Pr(lb < \mathbf{x}_j\mathbf{b} + u_j < 30)$  elsewhere.

$b$  missing ( $b \geq .$ ) means  $+\infty$ ; `pr(20,.)` calculates  $\Pr(+\infty > \mathbf{x}_j\mathbf{b} + u_j > 20)$ ;

`pr(20,ub)` calculates  $\Pr(+\infty > \mathbf{x}_j\mathbf{b} + u_j > 20)$  in observations for which  $ub \geq .$  and calculates  $\Pr(20 < \mathbf{x}_j\mathbf{b} + u_j < ub)$  elsewhere.

`e(a,b)` calculates  $E(\mathbf{x}_j\mathbf{b} + u_j | a < \mathbf{x}_j\mathbf{b} + u_j < b)$ , the expected value of  $y_j|\mathbf{x}_j$  conditional on  $y_j|\mathbf{x}_j$  being in the interval  $(a, b)$ , meaning that  $y_j|\mathbf{x}_j$  is censored.  $a$  and  $b$  are specified as they are for `pr()`.

`ystar(a,b)` calculates  $E(y_j^*)$ , where  $y_j^* = a$  if  $\mathbf{x}_j\mathbf{b} + u_j \leq a$ ,  $y_j^* = b$  if  $\mathbf{x}_j\mathbf{b} + u_j \geq b$ , and  $y_j^* = \mathbf{x}_j\mathbf{b} + u_j$  otherwise, meaning that  $y_j^*$  is truncated.  $a$  and  $b$  are specified as they are for `pr()`.

`score` is equivalent to `residuals` for linear regression models.



## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [cnsreg](#) — Constrained linear regression

[U] [20 Estimation and postestimation commands](#)

# Title

**constraint** — Define and list constraints

## Syntax

### Define constraints

```
constraint [define] # [exp=exp | coeflist]
```

### List constraints

```
constraint dir [numlist | _all]
```

```
constraint list [numlist | _all]
```

### Drop constraints

```
constraint drop { numlist | _all }
```

### Programmer's commands

```
constraint get #
```

```
constraint free
```

where *coeflist* is as defined in [R] **test** and # is restricted to the range 1–1,999, inclusive.

## Menu

Statistics > Other > Manage constraints

## Description

**constraint** defines, lists, and drops linear constraints. Constraints are for use by models that allow constrained estimation.

Constraints are defined by the **constraint** command. The currently defined constraints can be listed by either **constraint list** or **constraint dir**; both do the same thing. Existing constraints can be eliminated by **constraint drop**.

**constraint get** and **constraint free** are programmer's commands. **constraint get** returns the contents of the specified constraint in macro **r(contents)** and returns in scalar **r(defined)** 0 or 1—1 being returned if the constraint was defined. **constraint free** returns the number of a free (unused) constraint in macro **r(free)**.

## Remarks

Using constraints is discussed in [R] [cnsreg](#), [R] [mlogit](#), and [R] [reg3](#); this entry is concerned only with practical aspects of defining and manipulating constraints.

### ► Example 1

Constraints are numbered from 1 to 1,999, and we assign the number when we define the constraint:

```
. use http://www.stata-press.com/data/r11/sysdsn1
. constraint 2 [Indemnity]2.site = 0
```

The currently defined constraints can be listed by `constraint list`:

```
. constraint list
 2: [Indemnity]2.site = 0
```

`constraint drop` drops constraints:

```
. constraint drop 2
. constraint list
```

The empty list after `constraint list` indicates that no constraints are defined. Below we demonstrate the various syntaxes allowed by `constraint`:

```
. constraint 1 [Indemnity]
. constraint 10 [Indemnity]: 1.site 2.site
. constraint 11 [Indemnity]: 3.site
. constraint 21 [Prepaid=Uninsure]: nonwhite
. constraint 30 [Prepaid]
. constraint 31 [Insure]
. constraint list
 1: [Indemnity]
 10: [Indemnity]: 1.site 2.site
 11: [Indemnity]: 3.site
 21: [Prepaid=Uninsure]: nonwhite
 30: [Prepaid]
 31: [Insure]
. constraint drop 21-25, 31
. constraint list
 1: [Indemnity]
 10: [Indemnity]: 1.site 2.site
 11: [Indemnity]: 3.site
 30: [Prepaid]
. constraint drop _all
. constraint list
```



### □ Technical note

The `constraint` command does not check the syntax of the constraint itself because a constraint can be interpreted only in the context of a model. Thus `constraint` is willing to define constraints that later will not make sense. Any errors in the constraints will be detected and mentioned at the time of estimation.



## Reference

Weesie, J. 1999. [sg100: Two-stage linear constrained estimation](#). *Stata Technical Bulletin* 47: 24–30. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 217–225. College Station, TX: Stata Press.

## Also see

[R] [cnsreg](#) — Constrained linear regression

# Title

copyright — Display copyright information

## Syntax

copyright

## Description

copyright presents copyright notifications concerning tools, libraries, etc., used in the construction of Stata.

## Remarks

The correct form for a copyright notice is

Copyright *dates by author/owner*

The word “Copyright” is spelled out. You can use the © symbol, but “(C)” has never been given legal recognition. The phrase “All Rights Reserved” was historically required but is no longer needed.

Currently, most works are copyrighted from the moment they are written, and no copyright notice is required. Copyright concerns the protection of the expression and structure of facts and ideas, not the facts and ideas themselves. Copyright concerns the ownership of the expression and not the name given to the expression, which is covered under trademark law.

Copyright law as it exists today began in England in 1710 with the Statute of Anne, *An Act for the Encouragement of Learning, by Vesting the Copies of Printed Books in the Authors or Purchases of Such Copies, during the Times therein mentioned*. In 1672, Massachusetts introduced the first copyright law in what was to become the United States. After the Revolutionary War, copyright was introduced into the U.S. Constitution in 1787 and went into effect on May 31, 1790. On June 9, 1790, the first copyright in the United States was registered for *The Philadelphia Spelling Book* by John Barry.

There are significant differences in the understanding of copyright in the English- and non-English-speaking world. The Napoleonic or Civil Code, the dominant legal system in the non-English-speaking world, splits the rights into two classes: the author’s economic rights and the author’s moral rights. Moral rights are available only to “natural persons”. Legal persons (corporations) have economic rights but not moral rights.

## Also see

Copyright page of this book

# Title

copyright lapack — LAPACK copyright notification

## Description

Stata uses portions of LAPACK, a linear algebra package, with the express permission of the authors pursuant to the following notice:

Copyright © 1992–2008 The University of Tennessee. All rights reserved.

- Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the following disclaimer, listed in this license in the documentation or other materials provided with the distribution or both.
- Neither the names of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## Also see

[\[R\] copyright](#) — Display copyright information

# Title

copyright scintilla — Scintilla copyright notification

## Description

Stata uses portions of Scintilla with the express permission of the author, pursuant to the following notice:

Copyright © 1998–2002 by Neil Hodgson <neilh@scintilla.org>

All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

NEIL HODGSON DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL NEIL HODGSON BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

## Also see

[\[R\] copyright](#) — Display copyright information

# Title

copyright ttf2pt1 — ttf2pt1 copyright notification

## Description

Stata uses portions of ttf2pt1 to convert TrueType fonts to PostScript fonts, with express permission of the authors, pursuant to the following notice:

Copyright © 1997–2003 by the AUTHORS:

Andrew Weeks <ccsaw@bath.ac.uk>

Frank M. Siegert <fms@this.net>

Mark Heath <mheath@netspace.net.au>

Thomas Henlich <thenlich@rcs.urz.tu-dresden.de>

Sergey Babkin <babkin@users.sourceforge.net>, <sab123@hotmail.com>

Turgut Uyar <uyar@cs.itu.edu.tr>

Rihardas Hepas <rch@WriteMe.Com>

Szalay Tamas <tomek@elender.hu>

Johan Vromans <jvromans@squirrel.nl>

Petr Titera <P.Titera@sh.cvut.cz>

Lei Wang <lwang@amath8.amt.ac.cn>

Chen Xiangyang <chenxy@sun.ihep.ac.cn>

Zvezdan Petkovic <z.petkovic@computer.org>

Rigel <rigel863@yahoo.com>

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment: This product includes software developed by the TTF2PT1 Project and its contributors.

THIS SOFTWARE IS PROVIDED BY THE AUTHORS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL



DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## Also see

[\[R\] copyright](#) — Display copyright information

# Title

correlate — Correlations (covariances) of variables or coefficients

## Syntax

Display correlation matrix or covariance matrix

```
correlate [varlist] [if] [in] [weight] [, correlate_options]
```

Display all pairwise correlation coefficients

```
pwcorr [varlist] [if] [in] [weight] [, pwcorr_options]
```

| correlate_options  | description                                                            |
|--------------------|------------------------------------------------------------------------|
| Options            |                                                                        |
| <u>m</u> eans      | display means, standard deviations, minimums, and maximums with matrix |
| <u>n</u> oformat   | ignore display format associated with variables                        |
| <u>c</u> ovariance | display covariances                                                    |
| <u>w</u> rap       | allow wide matrices to wrap                                            |

| pwcorr_options     | description                                    |
|--------------------|------------------------------------------------|
| Main               |                                                |
| <u>o</u> bs        | print number of observations for each entry    |
| <u>s</u> ig        | print significance level for each entry        |
| <u>l</u> istwise   | use listwise deletion to handle missing values |
| <u>c</u> asewise   | synonym for listwise                           |
| <u>p</u> rint(#)   | significance level for displaying coefficients |
| <u>s</u> tar(#)    | significance level for displaying with a star  |
| <u>b</u> onferroni | use Bonferroni-adjusted significance level     |
| <u>s</u> idak      | use Šidák-adjusted significance level          |

*varlist* may contain time-series operators; see [U] 11.4.4 Time-series varlists.  
by is allowed with correlate and pwcorr; see [D] by.  
aweight and fweight are allowed; see [U] 11.1.6 weight.

## Menu

### correlate

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Correlations and covariances

### pwcorr

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Pairwise correlations

## Description

The `correlate` command displays the correlation matrix or covariance matrix for a group of variables. If *varlist* is not specified, the matrix is displayed for all variables in the dataset. Also see the `estat vce` command in [R] [estat](#).

`pwcorr` displays all the pairwise correlation coefficients between the variables in *varlist* or, if *varlist* is not specified, all the variables in the dataset.

## Options for correlate

### Options

`means` displays summary statistics (means, standard deviations, minimums, and maximums) with the matrix.

`noformat` displays the summary statistics requested by the `means` option in `g` format, regardless of the display formats associated with the variables.

`covariance` displays the covariances rather than the correlation coefficients.

`wrap` requests that no action be taken on wide correlation matrices to make them readable. It prevents Stata from breaking wide matrices into pieces to enhance readability. You might want to specify this option if you are displaying results in a window wider than 80 characters. Then you may need to set `linesize` to however many characters you can display across a line; see [R] [log](#).

## Options for pwcorr

### Main

`obs` adds a line to each row of the matrix reporting the number of observations used to calculate the correlation coefficient.

`sig` adds a line to each row of the matrix reporting the significance level of each correlation coefficient.

`listwise` handles missing values through listwise deletion, meaning that the entire observation is omitted from the estimation sample if any of the variables in *varlist* is missing for that observation. By default, `pwcorr` handles missing values by pairwise deletion; all available observations are used to calculate each pairwise correlation without regard to whether variables outside that pair are missing.

`correlate` uses listwise deletion. Thus `listwise` allows users of `pwcorr` to mimic `correlate`'s treatment of missing values while retaining access to `pwcorr`'s features.

`casewise` is a synonym for `listwise`.

`print(#)` specifies the significance level of correlation coefficients to be printed. Correlation coefficients with larger significance levels are left blank in the matrix. Typing `pwcorr, print(.10)` would list only correlation coefficients significant at the 10% level or better.

`star(#)` specifies the significance level of correlation coefficients to be starred. Typing `pwcorr, star(.05)` would star all correlation coefficients significant at the 5% level or better.

`bonferroni` makes the Bonferroni adjustment to calculated significance levels. This option affects printed significance levels and the `print()` and `star()` options. Thus `pwcorr, print(.05)` `bonferroni` prints coefficients with Bonferroni-adjusted significance levels of 0.05 or less.



To obtain the correlations between `mrgrate`, `dvcrate`, and `medage`, we type

```
. correlate mrgrate dvcrate medage
(obs=50)
```

|         | mrgrate | dvcrate | medage |
|---------|---------|---------|--------|
| mrgrate | 1.0000  |         |        |
| dvcrate | 0.7700  | 1.0000  |        |
| medage  | -0.0177 | -0.2229 | 1.0000 |

◀

## ► Example 2

The `pop` variable in our previous example represents the total population of the state. Thus, to obtain population-weighted correlations among `mrgrate`, `dvcrate`, and `medage`, we type

```
. correlate mrgrate dvcrate medage [w=pop]
(analytic weights assumed)
(sum of wgt is 2.2591e+08)
(obs=50)
```

|         | mrgrate | dvcrate | medage |
|---------|---------|---------|--------|
| mrgrate | 1.0000  |         |        |
| dvcrate | 0.5854  | 1.0000  |        |
| medage  | -0.1316 | -0.2833 | 1.0000 |

◀

With the `covariance` option, `correlate` can be used to obtain covariance matrices, as well as correlation matrices, for both weighted and unweighted data.

## ► Example 3

To obtain the matrix of covariances between `mrgrate`, `dvcrate`, and `medage`, we type `correlate mrgrate dvcrate medage, covariance`:

```
. correlate mrgrate dvcrate medage, covariance
(obs=50)
```

|         | mrgrate  | dvcrate  | medage  |
|---------|----------|----------|---------|
| mrgrate | .000662  |          |         |
| dvcrate | .000063  | 1.0e-05  |         |
| medage  | -.000769 | -.001191 | 2.86775 |

We could have obtained the `pop`-weighted covariance matrix by typing `correlate mrgrate dvcrate medage [w=pop], covariance`.

◀

## pwcorr

`correlate` calculates correlation coefficients by using casewise deletion; when you request correlations of variables  $x_1, x_2, \dots, x_k$ , any observation for which any of  $x_1, x_2, \dots, x_k$  is missing is not used. Thus if  $x_3$  and  $x_4$  have no missing values, but  $x_2$  is missing for half the data, the correlation between  $x_3$  and  $x_4$  is calculated using only the half of the data for which  $x_2$  is not missing. Of course, you can obtain the correlation between  $x_3$  and  $x_4$  by using all the data by typing `correlate x3 x4`.

`pwcorr` makes obtaining such pairwise correlation coefficients easier.

➤ Example 4

Using `auto.dta`, we investigate the correlation between several of the variables.

```
. use http://www.stata-press.com/data/r11/auto1
(Automobile Models)
. pwcorr mpg price rep78 foreign, obs sig
```

|         | mpg     | price  | rep78  | foreign |
|---------|---------|--------|--------|---------|
| mpg     | 1.0000  |        |        |         |
|         | 74      |        |        |         |
| price   | -0.4594 | 1.0000 |        |         |
|         | 0.0000  | 74     |        |         |
| rep78   | 0.3739  | 0.0066 | 1.0000 |         |
|         | 0.0016  | 0.9574 | 69     |         |
| foreign | 0.3613  | 0.0487 | 0.5922 | 1.0000  |
|         | 0.0016  | 0.6802 | 0.0000 | 74      |

```
. pwcorr mpg price headroom rear_seat trunk rep78 foreign, print(.05) star(.01)
```

|           | mpg      | price   | headroom | rear_s~t | trunk    | rep78   | foreign |
|-----------|----------|---------|----------|----------|----------|---------|---------|
| mpg       | 1.0000   |         |          |          |          |         |         |
| price     | -0.4594* | 1.0000  |          |          |          |         |         |
| headroom  | -0.4220* |         | 1.0000   |          |          |         |         |
| rear_seat | -0.5213* | 0.4194* | 0.5238*  | 1.0000   |          |         |         |
| trunk     | -0.5703* | 0.3143* | 0.6620*  | 0.6480*  | 1.0000   |         |         |
| rep78     | 0.3739*  |         |          |          |          | 1.0000  |         |
| foreign   | 0.3613*  |         | -0.2939  | -0.2409  | -0.3594* | 0.5922* | 1.0000  |

```
. pwcorr mpg price headroom rear_seat trunk rep78 foreign, print(.05) bon
```

|           | mpg     | price  | headroom | rear_s~t | trunk   | rep78  | foreign |
|-----------|---------|--------|----------|----------|---------|--------|---------|
| mpg       | 1.0000  |        |          |          |         |        |         |
| price     | -0.4594 | 1.0000 |          |          |         |        |         |
| headroom  | -0.4220 |        | 1.0000   |          |         |        |         |
| rear_seat | -0.5213 | 0.4194 | 0.5238   | 1.0000   |         |        |         |
| trunk     | -0.5703 |        | 0.6620   | 0.6480   | 1.0000  |        |         |
| rep78     | 0.3739  |        |          |          |         | 1.0000 |         |
| foreign   | 0.3613  |        |          |          | -0.3594 | 0.5922 | 1.0000  |

□ Technical note

The `correlate` command will report the correlation matrix of the data, but there are occasions when you need the matrix stored as a Stata matrix so that you can further manipulate it. You can obtain the matrix by typing

```
. matrix accum R = varlist, nocons dev
. matrix R = corr(R)
```

The first line places the cross-product matrix of the data in matrix `R`. The second line converts that to a correlation matrix. Also see [\[P\] matrix define](#) and [\[P\] matrix accum](#).



## Saved results

`correlate` saves the following in `r()`:

### Scalars

|                        |                                     |                       |                                               |
|------------------------|-------------------------------------|-----------------------|-----------------------------------------------|
| <code>r(N)</code>      | number of observations              | <code>r(Var_1)</code> | variance of first variable (covariance only)  |
| <code>r(rho)</code>    | $\rho$ (first and second variables) | <code>r(Var_2)</code> | variance of second variable (covariance only) |
| <code>r(cov_12)</code> | covariance (covariance only)        |                       |                                               |

### Matrices

|                   |                                  |
|-------------------|----------------------------------|
| <code>r(C)</code> | correlation or covariance matrix |
|-------------------|----------------------------------|

`pwcorr` will leave in its wake only the results of the last call that it makes internally to `correlate` for the correlation between the last variable and itself. Only rarely is this feature useful.

## Methods and formulas

`pwcorr` is implemented as an ado-file.

For a discussion of correlation, see, for instance, [Snedecor and Cochran \(1989, 177–195\)](#); for an introductory explanation using Stata examples, see [Acock \(2008, 168–174\)](#).

According to [Snedecor and Cochran \(1989, 180\)](#), the term “co-relation” was first proposed by [Galton \(1888\)](#). The product-moment correlation coefficient is often called the Pearson product-moment correlation coefficient because [Pearson \(1896\)](#) and [Pearson and Filon \(1898\)](#) were partially responsible for popularizing its use. See [Stigler \(1986\)](#) for information on the history of correlation.

The estimate of the product-moment correlation coefficient,  $\rho$ , is

$$\hat{\rho} = \frac{\sum_{i=1}^n w_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n w_i (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n w_i (y_i - \bar{y})^2}}$$

where  $w_i$  are the weights, if specified, or  $w_i = 1$  if weights are not specified.  $\bar{x} = (\sum w_i x_i) / (\sum w_i)$  is the mean of  $x$ , and  $\bar{y}$  is similarly defined.

The unadjusted significance level is calculated by `pwcorr` as

$$p = 2 * \text{ttail}(n - 2, |\hat{\rho}| \sqrt{n - 2} / \sqrt{1 - \hat{\rho}^2})$$

Let  $v$  be the number of variables specified so that  $k = v(v - 1)/2$  correlation coefficients are to be estimated. If `bonferroni` is specified, the adjusted significance level is  $p' = \min(1, kp)$ . If `sidak` is specified,  $p' = \min\{1, 1 - (1 - p)^k\}$ . In both cases, see [Methods and formulas](#) in [\[R\] oneway](#) for a more complete description of the logic behind these adjustments.

Carlo Emilio Bonferroni (1892–1960) studied in Turin and taught there and in Bari and Florence. He published on actuarial mathematics, probability, statistics, analysis, geometry, and mechanics. His work on probability inequalities has been applied to simultaneous statistical inference, although the method known as Bonferroni adjustment usually relies only on an inequality established earlier by Boole.

Karl Pearson (1857–1936) studied mathematics at Cambridge. He was professor of applied mathematics (1884–1911) and eugenics (1911–1933) at University College London. His publications include literary, historical, philosophical, and religious topics. Statistics became his main interest in the early 1890s after he learned about its application to biological problems. His work centered on distribution theory, the method of moments, correlation, and regression. Pearson introduced the chi-squared test and the terms coefficient of variation, contingency table, heteroskedastic, histogram, homoskedastic, kurtosis, mode, random sampling, random walk, skewness, standard deviation, and truncation. Despite many strong qualities, he also fell into prolonged disagreements with others, most notably, William Bateson and R. A. Fisher.

Zbyněk Šidák (1933–1999) was an outstanding Czech statistician and probabilist who worked on Markov chains, rank tests, multivariate distribution theory and multiple comparison methods.

## References

- Acock, A. C. 2008. *A Gentle Introduction to Stata*. 2nd ed. College Station, TX: Stata Press.
- Dewey, M. E., and E. Seneta. 2001. Carlo Emilio Bonferroni. In *Statisticians of the Centuries*, ed. C. C. Heyde and E. Seneta, 411–414. New York: Springer.
- Eisenhart, C. 1974. Pearson, Karl. In Vol. 10 of *Dictionary of Scientific Biography*, ed. C. C. Gillispie, 447–473. New York: Charles Scribner's Sons.
- Galton, F. 1888. Co-relations and their measurement, chiefly from anthropometric data. *Proceedings of the Royal Society of London* 45: 135–145.
- Gleason, J. R. 1996. [sg51: Inference about correlations using the Fisher z-transform](#). *Stata Technical Bulletin* 32: 13–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 121–128. College Station, TX: Stata Press.
- Goldstein, R. 1996. [sg52: Testing dependent correlation coefficients](#). *Stata Technical Bulletin* 32: 18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 128–129. College Station, TX: Stata Press.
- Pearson, K. 1896. Mathematical contributions to the theory of evolution—III. Regression, heredity, and panmixia. *Philosophical Transactions of the Royal Society of London, Series A* 187: 253–318.
- Pearson, K., and L. N. G. Filon. 1898. Mathematical contributions to the theory of evolution. IV. On the probable errors of frequency constants and on the influence of random selection on variation and correlation. *Philosophical Transactions of the Royal Society of London, Series A* 191: 229–311.
- Porter, T. M. 2004. *Karl Pearson: The Scientific Life in a Statistical Age*. Princeton, NJ: Princeton University Press.
- Rodgers, J. L., and W. A. Nicewander. 1988. Thirteen ways to look at the correlation coefficient. *American Statistician* 42: 59–66.
- Rovine, M. J., and A. von Eye. 1997. A 14th way to look at the correlation coefficient: Correlation as the proportion of matches. *American Statistician* 51: 42–46.
- Seed, P. T. 2001. [sg159: Confidence intervals for correlations](#). *Stata Technical Bulletin* 59: 27–28. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 267–269. College Station, TX: Stata Press.
- Seidler, J., J. Vondráček, and I. Saxl. 2000. The life and work of Zbyněk Šidák (1933–1999). *Applications of Mathematics* 45: 321–336.
- Snedecor, G. W., and W. G. Cochran. 1989. *Statistical Methods*. 8th ed. Ames, IA: Iowa State University Press.
- Stigler, S. M. 1986. *The History of Statistics: The Measurement of Uncertainty before 1900*. Cambridge, MA: Belknap Press.



- Wolfe, F. 1997. [sg64: pwcrrs: Enhanced correlation display](#). *Stata Technical Bulletin* 35: 22–25. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 163–167. College Station, TX: Stata Press.
- . 1999. [sg64.1: Update to pwcrrs](#). *Stata Technical Bulletin* 49: 17. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, p. 159. College Station, TX: Stata Press.

## Also see

- [R] [pcorr](#) — Partial and semipartial correlation coefficients
- [R] [spearman](#) — Spearman’s and Kendall’s correlations
- [R] [summarize](#) — Summary statistics
- [R] [tetrachoric](#) — Tetrachoric correlations for binary variables

Syntax

```
cumul varname [if] [in] [weight] , generate(newvar) [options]
```

| <i>options</i>                                                             | description                                |
|----------------------------------------------------------------------------|--------------------------------------------|
| Main                                                                       |                                            |
| * <u>generate</u> ( <i>newvar</i> )                                        | create variable <i>newvar</i>              |
| <u>freq</u>                                                                | use frequency units for cumulative         |
| <u>equal</u>                                                               | generate equal cumulatives for tied values |
| * <u>generate</u> ( <i>newvar</i> ) is required.                           |                                            |
| by is allowed; see <a href="#">[D] by</a> .                                |                                            |
| fweights and aweights are allowed; see <a href="#">[U] 11.1.6 weight</a> . |                                            |

Menu

Statistics > Summaries, tables, and tests > Distributional plots and tests > Cumulative distribution graph

Description

cumul creates *newvar*, defined as the empirical cumulative distribution function of *varname*.

Options

- Main
- generate(*newvar*) is required. It specifies the name of the new variable to be created.
  - freq specifies that the cumulative be in frequency units; otherwise, it is normalized so that *newvar* is 1 for the largest value of *varname*.
  - equal requests that observations with equal values in *varname* get the same cumulative value in *newvar*.

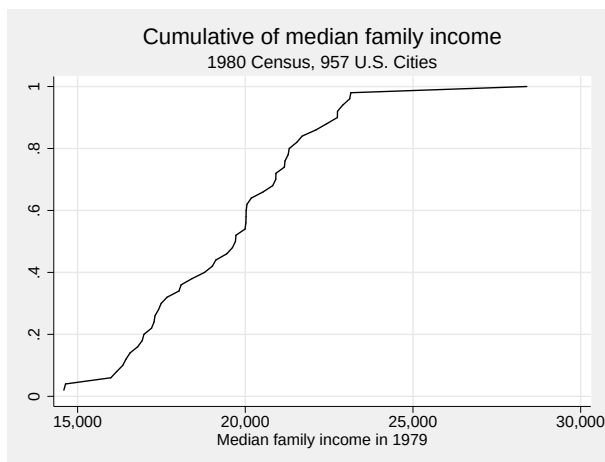
Jean Baptiste Joseph Fourier (1768–1830) was born in Auxerre in France. He got caught up in the Revolution and its aftermath and was twice arrested and imprisoned between periods of studying and teaching mathematics. Fourier joined Napoleon’s army in its invasion of Egypt in 1798 as a scientific adviser, returning to France in 1801, when he was appointed Prefect of the Department of Isère. While Prefect, Fourier did his important mathematical work on the theory of heat, based on what are now called Fourier series. This work was published in 1822, despite the skepticism of Lagrange, Laplace, Legendre, and others—who found the work lacking in generality and even rigor—and disagreements of both priority and substance with Biot and Poisson.

## Remarks

### ► Example 1

`cumul` is most often used with `graph` to graph the empirical cumulative distribution. For instance, we have data on the median family income of 957 U.S. cities:

```
. use http://www.stata-press.com/data/r11/hsng
(1980 Census housing data)
. cumul faminc, gen(cum)
. sort cum
. line cum faminc, ylab(, grid) ytitle("") xlab(, grid)
> title("Cumulative of median family income")
> subtitle("1980 Census, 957 U.S. Cities")
```



It would have been enough to type `line cum faminc`, but we wanted to make the graph look better; see [\[G\] graph twoway line](#).

If we had wanted a weighted cumulative, we would have typed `cumul faminc [w=pop]` at the first step.



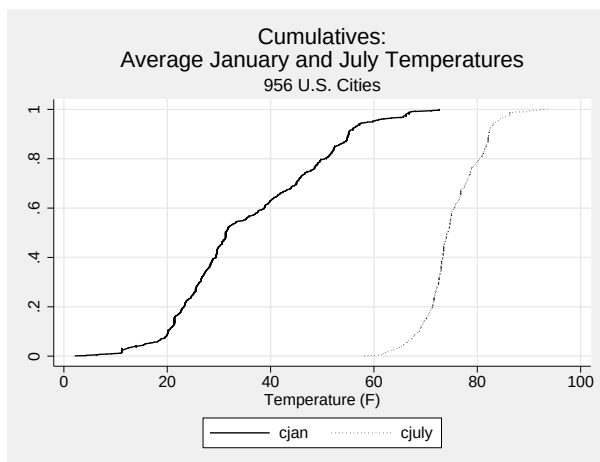
### ► Example 2

To graph two (or more) cumulatives on the same graph, use `cumul` and `stack`; see [\[D\] stack](#). For instance, we have data on the average January and July temperatures of 956 U.S. cities:

```

. use http://www.stata-press.com/data/r11/citytemp
(City Temperature Data)
. cumul tempjan, gen(cjan)
. cumul tempjuly, gen(cjuly)
. stack cjan tempjan cjuly tempjuly, into(c temp) wide clear
. line cjan cjuly temp, sort ylab(, grid) ytitle("") xlab(, grid)
> xtitle("Temperature (F)")
> title("Cumulatives:" "Average January and July Temperatures")
> subtitle("956 U.S. Cities") clstyle(. dot)

```



As before, it would have been enough to type `line cjan cjuly temp, sort`. See [\[D\] stack](#) for an explanation of how the `stack` command works.

◀

### □ Technical note

According to [Beniger and Robyn \(1978\)](#), [Fourier \(1821\)](#) published the first graph of a cumulative frequency distribution, which was later given the name “ogive” by [Galton \(1875\)](#).

□

## Methods and formulas

`cumul` is implemented as an ado-file.

## Acknowledgment

The `equal` option was added by Nicholas J. Cox, Durham University, Durham, UK.

## References

- Beniger, J. R., and D. L. Robyn. 1978. Quantitative graphics in statistics: A brief history. *American Statistician* 32: 1–11.
- Clayton, D. G., and M. Hills. 1999. [gr37: Cumulative distribution function plots](#). *Stata Technical Bulletin* 49: 10–12. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 96–98. College Station, TX: Stata Press.
- Cox, N. J. 1999. [gr41: Distribution function plots](#). *Stata Technical Bulletin* 51: 12–16. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 108–112. College Station, TX: Stata Press.
- Fourier, J. B. J. 1821. Notions générales, sur la population. *Recherches Statistiques sur la Ville de Paris et le Département de la Seine* 1: 1–70.
- Galton, F. 1875. Statistics by intercomparison, with remarks on the law of frequency of error. *Philosophical Magazine* 49: 33–46.
- Wilk, M. B., and R. Gnanadesikan. 1968. Probability plotting methods for the analysis of data. *Biometrika* 55: 1–17.

## Also see

- [D] [stack](#) — Stack data
- [R] [diagnostic plots](#) — Distributional diagnostic plots
- [R] [kdensity](#) — Univariate kernel density estimation

# Title

cusum — Graph cumulative spectral distribution

## Syntax

```
cusum yvar xvar [if] [in] [, options]
```

| options                                 | description                                                                               |
|-----------------------------------------|-------------------------------------------------------------------------------------------|
| Main                                    |                                                                                           |
| <code>generate(newvar)</code>           | save cumulative sum in <i>newvar</i>                                                      |
| <code>yfit (fitvar)</code>              | calculate cumulative sum against <i>fitvar</i>                                            |
| <code>nograph</code>                    | suppress the plot                                                                         |
| <code>nocalc</code>                     | suppress cusum test statistics                                                            |
| Cusum plot                              |                                                                                           |
| <code>connect_options</code>            | affect the rendition of the plotted line                                                  |
| Add plots                               |                                                                                           |
| <code>addplot(plot)</code>              | add plots to the generated graph                                                          |
| Y axis, X axis, Titles, Legend, Overall |                                                                                           |
| <code>twoway_options</code>             | any options other than <code>by()</code> documented in <a href="#">[G] twoway_options</a> |

## Menu

Statistics > Other > Quality control > Cusum plots and tests for binary variables

## Description

`cusum` graphs the cumulative sum (cusum) of a binary (0/1) variable, *yvar*, against a (usually) continuous variable, *xvar*.

## Options

|                                                                                                                                                                                                                                                                               |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Main                                                                                                                                                                                                                                                                          |
| <code>generate(newvar)</code> saves the cusum in <i>newvar</i> .                                                                                                                                                                                                              |
| <code>yfit(fitvar)</code> calculates a cusum against <i>fitvar</i> , that is, the running sums of the “residuals” <i>fitvar</i> minus <i>yvar</i> . Typically, <i>fitvar</i> is the predicted probability of a positive outcome obtained from a logistic regression analysis. |
| <code>nograph</code> suppresses the plot.                                                                                                                                                                                                                                     |
| <code>nocalc</code> suppresses calculation of the cusum test statistics.                                                                                                                                                                                                      |
| Cusum plot                                                                                                                                                                                                                                                                    |
| <code>connect_options</code> affect the rendition of the plotted line; see <a href="#">[G] connect_options</a> .                                                                                                                                                              |

## Add plots

`addplot(plot)` provides a way to add other plots to the generated graph. See [G] [addplot\\_option](#).

## Y axis, X axis, Titles, Legend, Overall

`twoway_options` are any of the options documented in [G] [twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

## Remarks

The `cusum` is the running sum of the proportion of ones in the sample, a constant number, minus `yvar`,

$$c_j = \sum_{k=1}^j f - yvar_{(k)}, \quad 1 \leq j \leq N$$

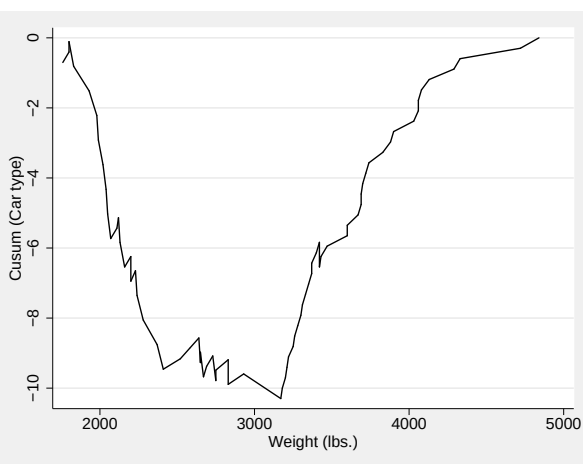
where  $f = (\sum yvar)/N$  and  $yvar_{(k)}$  refers to the corresponding value of `yvar` when `xvar` is placed in ascending order:  $xvar_{(k+1)} \geq xvar_{(k)}$ . Tied values of `xvar` are broken at random. If you want them broken the same way in two runs, you must set the random-number seed to the same value before giving the `cusum` command; see [R] [set seed](#).

A U-shaped or inverted U-shaped `cusum` indicates, respectively, a negative or a positive trend of `yvar` with `xvar`. A sinusoidal shape is evidence of a nonmonotonic (for example, quadratic) trend. `cusum` displays the maximum absolute `cusum` for monotonic and nonmonotonic trends of `yvar` on `xvar`. These are nonparametric tests of departure from randomness of `yvar` with respect to `xvar`. Approximate values for the tests are given.

### ► Example 1

For the automobile dataset, `auto.dta`, we wish to investigate the relationship between `foreign` (0 = domestic, 1 = foreign) and car weight as follows:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. cusum foreign weight
```

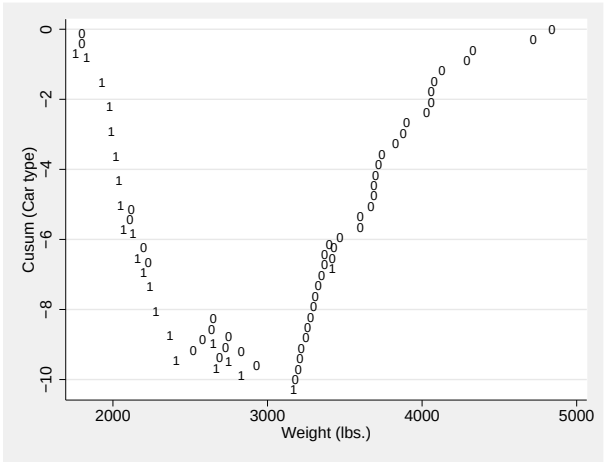


| Variable | Obs | Pr(1)  | CusumL | zL    | Pr>zL | CusumQ | zQ    | Pr>zQ |
|----------|-----|--------|--------|-------|-------|--------|-------|-------|
| foreign  | 74  | 0.2973 | 10.30  | 3.963 | 0.000 | 3.32   | 0.469 | 0.320 |

The resulting plot, which is U-shaped, suggests a negative monotonic relationship. The trend is confirmed by a highly significant linear cusum statistic, labeled CusumL in the output above.

Some 29.73% of the cars are foreign (coded 1). The proportion of foreign cars diminishes with increasing weight. The domestic cars are crudely heavier than the foreign ones. We could have discovered that by typing `table foreign, stats(mean weight)`, but such an approach does not give the full picture of the relationship. The quadratic cusum (CusumQ) is not significant, so we do not suspect any tendency for the very heavy cars to be foreign rather than domestic. A slightly enhanced version of the plot shows the preponderance of domestic (coded 0) cars at the heavy end of the weight axis:

```
. label values foreign
. cusum foreign weight, s(none) recast(scatter) mlabel(foreign) mlabp(0)
```



| Variable | Obs | Pr(1)  | CusumL | zL    | Pr>zL | CusumQ | zQ    | Pr>zQ |
|----------|-----|--------|--------|-------|-------|--------|-------|-------|
| foreign  | 74  | 0.2973 | 10.30  | 3.963 | 0.000 | 2.92   | 0.064 | 0.475 |

The example is, of course, artificial, because we would not really try to model the probability of a car being foreign given its weight.

## Saved results

cusum saves the following in `r()`:

|                        |                                 |                        |                                      |
|------------------------|---------------------------------|------------------------|--------------------------------------|
| Scalars                |                                 |                        |                                      |
| <code>r(N)</code>      | number of observations          | <code>r(P_zl)</code>   | <i>p</i> -value for test (linear)    |
| <code>r(prop1)</code>  | proportion of positive outcomes | <code>r(cusumq)</code> | quadratic cusum                      |
| <code>r(cusuml)</code> | cusum                           | <code>r(zq)</code>     | test (quadratic)                     |
| <code>r(zl)</code>     | test (linear)                   | <code>r(P_zq)</code>   | <i>p</i> -value for test (quadratic) |



## Methods and formulas

`cusum` is implemented as an ado-file.

## Acknowledgment

`cusum` was written by Patrick Royston, MRC Clinical Trials Unit, London.

## References

- Royston, P. 1992. The use of cusums and other techniques in modelling continuous covariates in logistic regression. *Statistics in Medicine* 11: 1115–1129.
- . 1993. [sqv7: Cusum plots and tests for binary variables](#). *Stata Technical Bulletin* 12: 16–17. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 175–177. College Station, TX: Stata Press.

## Also see

- [R] [logistic](#) — Logistic regression, reporting odds ratios
- [R] [logit](#) — Logistic regression, reporting coefficients
- [R] [probit](#) — Probit regression

# Title

**db** — Launch dialog

## Syntax

*Syntax for db*

**db** *commandname*

*For programmers*

**db** *commandname* [ , **message**(*string*) **debug** **dryrun** ]

*Set system parameter*

**set** **maxdb** # [ , **permanently** ]

where # must be between 5 and 1,000.

## Description

**db** is the command-line way to launch a dialog for a Stata command.

The second syntax (which is the same but includes options) is for use by programmers.

If you wish to allow the launching of dialogs from a help file, see [\[P\] smcl](#) for information on the **dialog** SMCL directive.

**set maxdb** sets the maximum number of dialog boxes whose contents are remembered from one invocation to the next during a session. The default value of **maxdb** is 50.

## Options

**message**(*string*) specifies that *string* be passed to the dialog box, where it can be referred to from the `__MESSAGE STRING` property.

**debug** specifies that the underlying dialog box be loaded with debug messaging turned on.

**dryrun** specifies that, rather than launching the dialog, **db** show the commands it would issue to launch the dialog.

**permanently** specifies that, in addition to making the change right now, the **maxdb** setting be remembered and become the default setting when you invoke Stata.

## Remarks

The usual way to launch a dialog is to open the **Data**, **Graphics**, or **Statistics** menu and to make your selection from there. When you know the name of the command that you want to run, however, **db** provides a way to invoke the dialog from the command line.

**db** follows the same abbreviation rules that Stata's command-line interface follows. So, to launch the dialog for **regress**, you can type

```
. db regress
```

or

```
. db reg
```

Say that you use the dialog box for **regress**, either by selecting

**Statistics > Linear models and related > Linear regression**

or by typing

```
. db regress
```

You fit a regression.

Much later during the session, you return to the **regress** dialog box. It will have the contents as you left them if 1) you have not typed **clear all** between the first and second invocations; 2) you have not typed **discard** between the two invocations; and 3) you have not used more than 50 different dialog boxes—regardless of how many times you have used each—between the first and second invocations of **regress**. If you use 51 or more, the contents of the **regress** dialog box will be forgotten.

**set maxdb** determines how many different dialog boxes are remembered. A dialog box takes, on average, about 20 KB of memory, so the 50 default corresponds to allowing dialog boxes to consume about 1 MB of memory.

## Methods and formulas

**db** is implemented as an ado-file.

## Also see

[\[R\] query](#) — Display system parameters

Syntax

Symmetry plot

```
sympplot varname [if] [in] [, options1]
```

Ordered values of varname against quantiles of uniform distribution

```
quantile varname [if] [in] [, options1]
```

Quantiles of varname<sub>1</sub> against quantiles of varname<sub>2</sub>

```
qqplot varname1 varname2 [if] [in] [, options1]
```

Quantiles of varname against quantiles of normal distribution

```
qnorm varname [if] [in] [, options2]
```

Standardized normal probability plot

```
pnorm varname [if] [in] [, options2]
```

Quantiles of varname against quantiles of  $\chi^2$  distribution

```
qchi varname [if] [in] [, options3]
```

$\chi^2$  probability plot

```
pchi varname [if] [in] [, options3]
```

| options <sub>1</sub>                    | description                                                                         |
|-----------------------------------------|-------------------------------------------------------------------------------------|
| Plot                                    |                                                                                     |
| marker_options                          | change look of markers (color, size, etc.)                                          |
| marker_label_options                    | add marker labels; change look or position                                          |
| Reference line                          |                                                                                     |
| <u>rlopts</u> (cline_options)           | affect rendition of the reference line                                              |
| Add plots                               |                                                                                     |
| addplot(plot)                           | add other plots to the generated graph                                              |
| Y axis, X axis, Titles, Legend, Overall |                                                                                     |
| twoway_options                          | any options other than by() documented in <a href="#">[G]</a> <i>twoway_options</i> |

| <i>options</i> <sub>2</sub>             | description                                                                               |
|-----------------------------------------|-------------------------------------------------------------------------------------------|
| Main                                    |                                                                                           |
| <code>grid</code>                       | add grid lines                                                                            |
| Plot                                    |                                                                                           |
| <code>marker_options</code>             | change look of markers (color, size, etc.)                                                |
| <code>marker_label_options</code>       | add marker labels; change look or position                                                |
| Reference line                          |                                                                                           |
| <code>rlopts(cline_options)</code>      | affect rendition of the reference line                                                    |
| Add plots                               |                                                                                           |
| <code>addplot(plot)</code>              | add other plots to the generated graph                                                    |
| Y axis, X axis, Titles, Legend, Overall |                                                                                           |
| <code>twoway_options</code>             | any options other than <code>by()</code> documented in <a href="#">[G] twoway_options</a> |

| <i>options</i> <sub>3</sub>             | description                                                                               |
|-----------------------------------------|-------------------------------------------------------------------------------------------|
| Main                                    |                                                                                           |
| <code>grid</code>                       | add grid lines                                                                            |
| <code>df(#)</code>                      | degrees of freedom of $\chi^2$ distribution; default is <code>df(1)</code>                |
| Plot                                    |                                                                                           |
| <code>marker_options</code>             | change look of markers (color, size, etc.)                                                |
| <code>marker_label_options</code>       | add marker labels; change look or position                                                |
| Reference line                          |                                                                                           |
| <code>rlopts(cline_options)</code>      | affect rendition of the reference line                                                    |
| Add plots                               |                                                                                           |
| <code>addplot(plot)</code>              | add other plots to the generated graph                                                    |
| Y axis, X axis, Titles, Legend, Overall |                                                                                           |
| <code>twoway_options</code>             | any options other than <code>by()</code> documented in <a href="#">[G] twoway_options</a> |

Menu

symplot

Statistics > Summaries, tables, and tests > Distributional plots and tests > Symmetry plot

quantile

Statistics > Summaries, tables, and tests > Distributional plots and tests > Quantiles plot

qqplot

Statistics > Summaries, tables, and tests > Distributional plots and tests > Quantile-quantile plot

## qnorm

Statistics > Summaries, tables, and tests > Distributional plots and tests > Normal quantile plot

## pnorm

Statistics > Summaries, tables, and tests > Distributional plots and tests > Normal probability plot, standardized

## qchi

Statistics > Summaries, tables, and tests > Distributional plots and tests > Chi-squared quantile plot

## pchi

Statistics > Summaries, tables, and tests > Distributional plots and tests > Chi-squared probability plot

# Description

`symplot` graphs a symmetry plot of *varname*.

`quantile` plots the ordered values of *varname* against the quantiles of a uniform distribution.

`qqplot` plots the quantiles of *varname*<sub>1</sub> against the quantiles of *varname*<sub>2</sub> (Q–Q plot).

`qnorm` plots the quantiles of *varname* against the quantiles of the normal distribution (Q–Q plot).

`pnorm` graphs a standardized normal probability plot (P–P plot).

`qchi` plots the quantiles of *varname* against the quantiles of a  $\chi^2$  distribution (Q–Q plot).

`pchi` graphs a  $\chi^2$  probability plot (P–P plot).

See [\[R\] regress postestimation](#) for regression diagnostic plots and [\[R\] logistic postestimation](#) for logistic regression diagnostic plots.

# Options for `symplot`, `quantile`, and `qqplot`

## Plot

*marker\_options* affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [\[G\] marker\\_options](#).

*marker\_label\_options* specify if and how the markers are to be labeled; see [\[G\] marker\\_label\\_options](#).

## Reference line

`rlopts(cline_options)` affect the rendition of the reference line; see [\[G\] cline\\_options](#).

## Add plots

`addplot(plot)` provides a way to add other plots to the generated graph; see [\[G\] addplot\\_option](#).

## Y axis, X axis, Titles, Legend, Overall

*twoway\_options* are any of the options documented in [\[G\] twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [\[G\] title\\_options](#)) and for saving the graph to disk (see [\[G\] saving\\_option](#)).

## Options for qnorm and pnorm

### Main

`grid` adds grid lines at the 0.05, 0.10, 0.25, 0.50, 0.75, 0.90, and 0.95 quantiles when specified with `qnorm`. With `pnorm`, `grid` is equivalent to `yline(.25,.5,.75) xline(.25,.5,.75)`.

### Plot

`marker_options` affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [G] [marker\\_options](#).

`marker_label_options` specify if and how the markers are to be labeled; see [G] [marker\\_label\\_options](#).

### Reference line

`rlopts(cline_options)` affect the rendition of the reference line; see [G] [cline\\_options](#).

### Add plots

`addplot(plot)` provides a way to add other plots to the generated graph; see [G] [addplot\\_option](#).

### Y axis, X axis, Titles, Legend, Overall

`twoway_options` are any of the options documented in [G] [twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

## Options for qchi and pchi

### Main

`grid` adds grid lines at the 0.05, 0.10, 0.25, 0.50, 0.75, 0.90, and .95 quantiles when specified with `qchi`. With `pchi`, `grid` is equivalent to `yline(.25,.5,.75) xline(.25,.5,.75)`.

`df(#)` specifies the degrees of freedom of the  $\chi^2$  distribution. The default is 1.

### Plot

`marker_options` affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [G] [marker\\_options](#).

`marker_label_options` specify if and how the markers are to be labeled; see [G] [marker\\_label\\_options](#).

### Reference line

`rlopts(cline_options)` affect the rendition of the reference line; see [G] [cline\\_options](#).

### Add plots

`addplot(plot)` provides a way to add other plots to the generated graph; see [G] [addplot\\_option](#).

Y axis, X axis, Titles, Legend, Overall

`twoway_options` are any of the options documented in [G] [twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

## Remarks

Remarks are presented under the following headings:

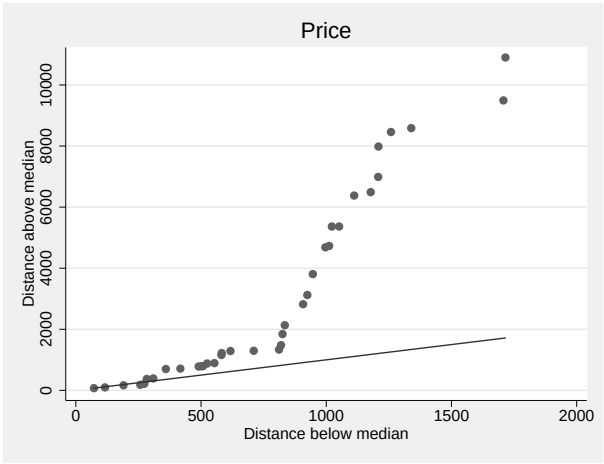
[symplot](#)  
[quantile](#)  
[qqplot](#)  
[qnorm](#)  
[pnorm](#)  
[qchi](#)  
[pchi](#)

### symplot

#### ► Example 1

We have data on 74 automobiles. To make a symmetry plot of the variable `price`, we type

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. symplot price
```



All points would lie along the reference line (defined as  $y = x$ ) if car prices were symmetrically distributed. The points in this plot lie above the reference line, indicating that the distribution of car prices is skewed to the right—the most expensive cars are far more expensive than the least expensive cars are inexpensive.

The logic works as follows: a variable,  $z$ , is distributed symmetrically if

$$\text{median} - z_{(i)} = z_{(N+1-i)} - \text{median}$$



where  $z_{(i)}$  indicates the  $i$ th-order statistic of  $z$ . `symplot` graphs  $y_i = \text{median} - z_{(i)}$  versus  $x_i = z_{(N+1-i)} - \text{median}$ .

For instance, consider the largest and smallest values of `price` in the example above. The most expensive car costs \$15,906 and the least expensive, \$3,291. Let's compare these two cars with the typical car in the data and see how much more it costs to buy the most expensive car, and compare that with how much less it costs to buy the least expensive car. If the automobile price distribution is symmetric, the price differences would be the same.

Before we can make this comparison, we must agree on a definition for the word “typical”. Let's agree that “typical” means median. The price of the median car is \$5,006.50, so the most expensive car costs \$10,899.50 more than the median car, and the least expensive car costs \$1,715.50 less than the median car. We now have one piece of evidence that the car price distribution is not symmetric. We can repeat the experiment for the second-most-expensive car and the second-least-expensive car. We find that the second-most-expensive car costs \$9,494.50 more than the median car, and the second-least-expensive car costs \$1,707.50 less than the median car. We now have more evidence. We can continue doing this with the third most expensive and the third least expensive, and so on.

Once we have all these numbers, we want to compare each pair and ask how similar, on average, they are. The easiest way to do that is to plot all the pairs.

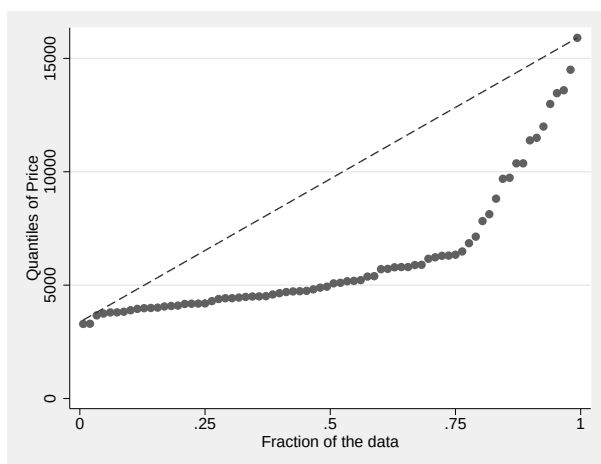


## quantile

### ► Example 2

We have data on the prices of 74 automobiles. To make a quantile plot of price, we type

```
. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)
. quantile price, rlopts(clpattern(dash))
```



We changed the pattern of the reference line by specifying `rlopts(clpattern(dash))`.

In a quantile plot, each value of the variable is plotted against the fraction of the data that have values less than that fraction. The diagonal line is a reference line. If automobile prices were rectangularly distributed, all the data would be plotted along the line. Because all the points are below the reference line, we know that the price distribution is skewed right.

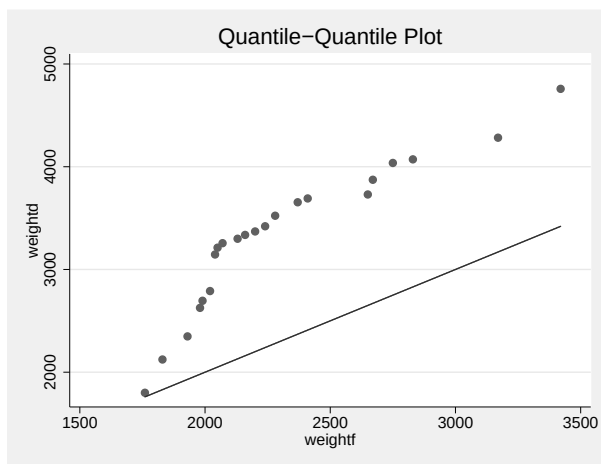


## qqplot

### ➤ Example 3

We have data on the weight and country of manufacture of 74 automobiles. We wish to compare the distributions of weights for domestic and foreign automobiles:

```
. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)
. generate weightd=weight if !foreign
(22 missing values generated)
. generate weightf=weight if foreign
(52 missing values generated)
. qqplot weightd weightf
```

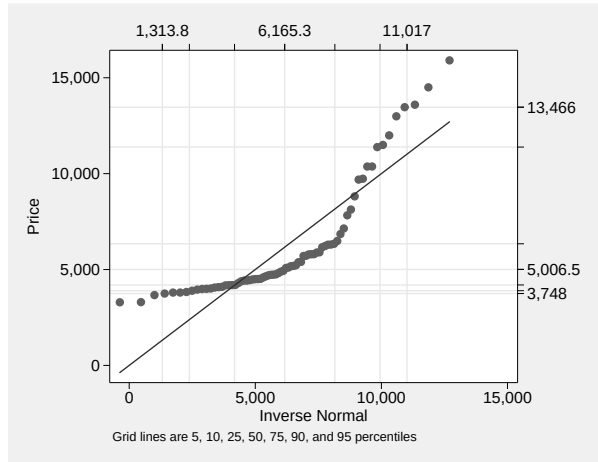


## qnorm

### ➤ Example 4

Continuing with our price data on 74 automobiles, we now wish to compare the distribution of price with the normal distribution:

```
. qnorm price, grid ylabel(, angle(horizontal) axis(1))
> ylabel(, angle(horizontal) axis(2))
```



The result shows that the distributions are different.

◀

## □ Technical note

The idea behind `qnorm` is recommended strongly by [Miller Jr. \(1997\)](#): he calls it probit plotting. His recommendations from much practical experience should interest many users. “My recommendation for detecting nonnormality is *probit plotting*” ([Miller Jr. 1997](#), 10). “If a deviation from normality cannot be spotted by eye on probit paper, it is not worth worrying about. I never use the Kolmogorov–Smirnov test (or one of its cousins) or the  $\chi^2$  test as a preliminary test of normality. They do not tell you how the sample is differing from normality, and I have a feeling they are more likely to detect irregularities in the middle of the distribution than in the tails” ([Miller Jr. 1997](#), 13–14).

□

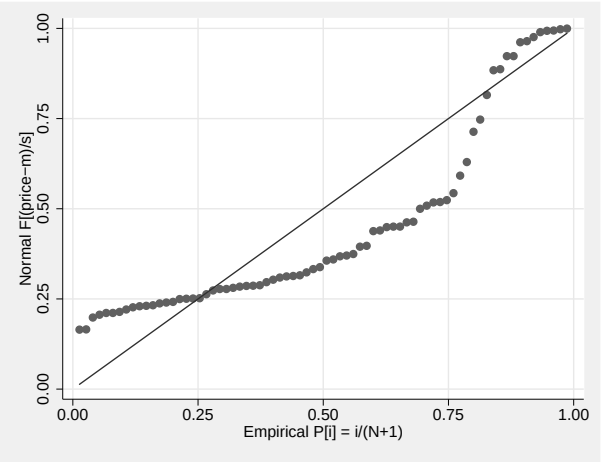
## pnorm

### ► Example 5

Quantile–normal plots emphasize the tails of the distribution. Normal probability plots put the focus on the center of the distribution:

(Continued on next page)

```
. pnorm price, grid
```

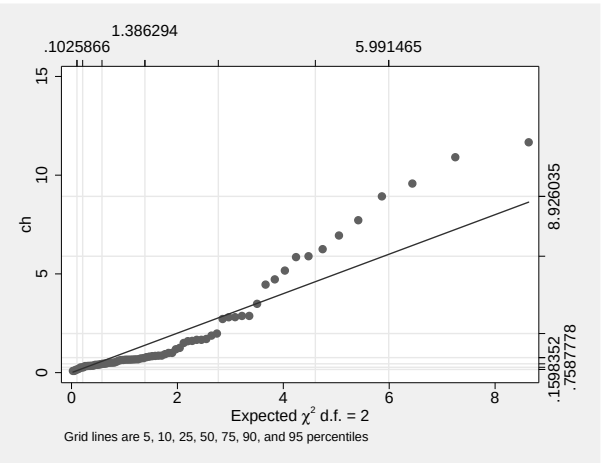


qchi

Example 6

Suppose that we want to examine the distribution of the sum of squares of price and mpg, standardized for their variances.

```
. egen c1 = std(price)
. egen c2 = std(mpg)
. generate ch = c1^2 + c2^2
. qchi ch, df(2) grid ylabel(, alt axis(2)) xlabel(, alt axis(2))
```



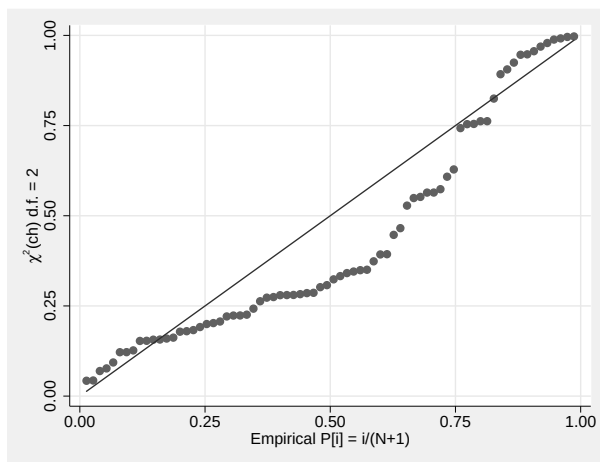
The quadratic form is clearly not  $\chi^2$  with 2 degrees of freedom.

## pchi

### ► Example 7

We can focus on the center of the distribution by doing a probability plot:

```
. pchi ch, df(2) grid
```



◀

## Methods and formulas

`symplot`, `quantile`, `qqplot`, `qnorm`, `pnorm`, `qchi`, and `pchi` are implemented as ado-files. Let  $x_{(1)}, x_{(2)}, \dots, x_{(N)}$  be the data sorted in ascending order.

If a continuous variable,  $x$ , has a cumulative distribution function  $F(x) = P(X \leq x) = p$ , the quantiles  $x_{p_i}$  are such that  $F(x_{p_i}) = p_i$ . For example, if  $p_i = 0.5$ , then  $x_{0.5}$  is the median. When we plot data, the probabilities,  $p_i$ , are often referred to as plotting positions. There are many different conventions for choice of plotting positions, given  $x_{(1)} \leq \dots \leq x_{(N)}$ . Most belong to the family  $(i - a)/(N - 2a + 1)$ .  $a = 0.5$  (suggested by Hazen) and  $a = 0$  (suggested by Weibull) are popular choices.

For a wider discussion of the calculation of plotting positions, see [Cox \(2002\)](#).

`symplot` plots median  $- x_{(i)}$  versus  $x_{(N+1-i)} - \text{median}$ .

`quantile` plots  $x_{(i)}$  versus  $(i - 0.5)/N$  (the Hazen position).

`qnorm` plots  $x_{(i)}$  against  $q_i$ , where  $q_i = \Phi^{-1}(p_i)$ ,  $\Phi$  is the cumulative normal distribution, and  $p_i = i/(N + 1)$  (the Weibull position).

`pnorm` plots  $\Phi\{(x_i - \hat{\mu})/\hat{\sigma}\}$  versus  $p_i = i/(N + 1)$ , where  $\hat{\mu}$  is the mean of the data and  $\hat{\sigma}$  is the standard deviation.

`qchi` and `pchi` are similar to `qnorm` and `pnorm`; the cumulative  $\chi^2$  distribution is used in place of the cumulative normal distribution.

`qqplot` is just a two-way scatterplot of one variable against the other after both variables have been sorted into ascending order, and both variables have the same number of nonmissing observations. If the variables have unequal numbers of nonmissing observations, interpolated values of the variable with more data are plotted against the variable with fewer data.

Ramanathan Gnanadesikan (1932– ) was born in Madras. He obtained degrees from the Universities of Madras and North Carolina. He worked in industry at Procter and Gamble, Bell Labs, and Bellcore, as well as in universities, retiring from Rutgers in 1998. Among many contributions to statistics he is especially well known for work on probability plotting, robustness, outlier detection, clustering, classification, and pattern recognition.

Martin Bradbury Wilk (1922– ) was born in Montreal. He obtained degrees in chemical engineering and statistics from McGill and Iowa State Universities. After several posts in statistics in industry and universities (including periods at Princeton, Bell Labs, and Rutgers), Wilk was appointed Chief Statistician at Statistics Canada (1980–1986). He is especially well known for his work with Gnanadesikan on probability plotting and with Shapiro on tests for normality.

## Acknowledgments

We thank Peter A. Lachenbruch of the Department of Public Health, Oregon State University, for writing the original version of `qchi` and `pchi`. Patrick Royston of the MRC Clinical Trials Unit, London, also published a similar command in the *Stata Technical Bulletin* (Royston 1996).

## References

- Chambers, J. M., W. S. Cleveland, B. Kleiner, and P. A. Tukey. 1983. *Graphical Methods for Data Analysis*. Belmont, CA: Wadsworth.
- Cox, N. J. 1999. [gr42: Quantile plots, generalized](#). *Stata Technical Bulletin* 51: 16–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 113–116. College Station, TX: Stata Press.
- . 2001. [gr42.1: Quantile plots, generalized: Update to Stata 7](#). *Stata Technical Bulletin* 61: 10. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 55–56. College Station, TX: Stata Press.
- . 2002. [Speaking Stata: On getting functions to do the work](#). *Stata Journal* 2: 411–427.
- . 2004a. [Speaking Stata: Graphing distributions](#). *Stata Journal* 4: 66–88.
- . 2004b. [gr42\\_2: Software update: Quantile plots, generalized](#). *Stata Journal* 4: 97.
- . 2005a. [Speaking Stata: Density probability plots](#). *Stata Journal* 5: 259–273.
- . 2005b. [Speaking Stata: The protean quantile plot](#). *Stata Journal* 5: 442–460.
- . 2005c. [Speaking Stata: Smoothing in various directions](#). *Stata Journal* 5: 574–593.
- . 2007. [Stata tip 47: Quantile–quantile plots without programming](#). *Stata Journal* 7: 275–279.
- Daniel, C., and F. S. Wood. 1980. *Fitting Equations to Data: Computer Analysis of Multifactor Data*. 2nd ed. New York: Wiley.
- Gan, F. F., K. J. Koehler, and J. C. Thompson. 1991. Probability plots and distribution curves for assessing the fit of probability models. *American Statistician* 45: 14–21.
- Hamilton, L. C. 1992. *Regression with Graphics: A Second Course in Applied Statistics*. Belmont, CA: Duxbury.
- . 2009. [Statistics with Stata \(Updated for Version 10\)](#). Belmont, CA: Brooks/Cole.
- Hoaglin, D. C. 1985. Using quantiles to study shape. In *Exploring Data Tables, Trends, and Shapes*, ed. D. C. Hoaglin, F. Mosteller, and J. W. Tukey, 417–460. New York: Wiley.

- Kettenring, J. R. 2001. A conversation with Ramanathan Gnanadesikan. *Statistical Science* 16: 295–309.
- Miller Jr., R. G. 1997. *Beyond ANOVA: Basics of Applied Statistics*. London: Chapman & Hall.
- Nolan, D., and T. Speed. 2000. *Stat Labs: Mathematical Statistics Through Applications*. New York: Springer.
- Royston, P. 1996. [sg47: A plot and a test for the  \$\chi^2\$  distribution](#). *Stata Technical Bulletin* 29: 26–27. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 142–144. College Station, TX: Stata Press.
- Scotto, M. G. 2000. [sg140: The Gumbel quantile plot and a test for choice of extreme models](#). *Stata Technical Bulletin* 55: 23–25. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 156–159. College Station, TX: Stata Press.
- Wilk, M. B., and R. Gnanadesikan. 1968. Probability plotting methods for the analysis of data. *Biometrika* 55: 1–17.

## Also see

- [\[R\] \*\*cumul\*\*](#) — Cumulative distribution
- [\[R\] \*\*kdensity\*\*](#) — Univariate kernel density estimation
- [\[R\] \*\*logistic postestimation\*\*](#) — Postestimation tools for logistic
- [\[R\] \*\*lv\*\*](#) — Letter-value displays
- [\[R\] \*\*regress postestimation\*\*](#) — Postestimation tools for regress

# Title

**display** — Substitute for a hand calculator

## Syntax

`display exp`

## Description

`display` displays strings and values of scalar expressions.

`display` really has many more features and a more complex syntax diagram, but the diagram shown above is adequate for interactive use. For a full discussion of `display`'s capabilities, see [\[P\] `display`](#).

## Remarks

`display` can be used as a substitute for a hand calculator.

### ► Example 1

`display 2+2` produces the output 4. Stata variables may also appear in the expression, such as in `display myvar/2`. Because `display` works only with scalars, the resulting calculation is performed only for the first observation. You could type `display myvar[10]/2` to display the calculation for the 10th observation. Here are more examples:

```
. display sqrt(2)/2
.70710678

. display normal(-1.1)
.13566606

. di (57.2-3)/(12-2)
5.42

. display myvar/10
7

. display myvar[10]/2
3.5
```

◀

## Also see

[\[P\] `display`](#) — Display strings and values of scalar expressions

[\[U\] 13 Functions and expressions](#)



# Title

**do** — Execute commands from a file

## Syntax

`{do | run} filename [arguments] [ , nostop]`

## Menu

File > Do...

## Description

`do` and `run` cause Stata to execute the commands stored in *filename* just as if they were entered from the keyboard. `do` echoes the commands as it executes them, whereas `run` is silent. If *filename* is specified without an extension, `.do` is assumed.

## Option

`nostop` allows the do-file to continue executing even if an error occurs. Normally, Stata stops executing the do-file when it detects an error (nonzero return code).

## Remarks

You can create *filename* (called a *do-file*) using Stata's Do-file Editor; see [\[R\] doedit](#). This file will be a standard ASCII (text) file. A complete discussion of do-files can be found in [\[U\] 16 Do-files](#).

You can also create *filename* by using a non-Stata text editor; see [\[D\] shell](#) for a way to invoke your favorite editor from inside Stata. Make sure that you save the file in ASCII format.

If the path or *filename* contains spaces, it should be enclosed in double quotes.

## Reference

Jenkins, S. P. 2006. [Stata tip 32: Do not stop](#). *Stata Journal* 6: 281.

## Also see

[\[R\] doedit](#) — Edit do-files and other text files

[\[P\] include](#) — Include commands from file

[\[GSM\] 13 Using the Do-file Editor—automating Stata](#)

[\[GSU\] 13 Using the Do-file Editor—automating Stata](#)

[\[GSW\] 13 Using the Do-file Editor—automating Stata](#)

[\[U\] 15 Saving and printing output—log files](#)

[\[U\] 16 Do-files](#)

# Title

doedit — Edit do-files and other text files

## Syntax

`doedit` [*filename*]

## Menu

Window > Do-file Editor

## Description

`doedit` opens a text editor that lets you edit do-files and other text files.

The Do-file Editor lets you submit several commands to Stata at once.

## Remarks

Clicking on the **Do-file Editor** button is equivalent to typing `doedit`.

`doedit`, typed by itself, invokes the Editor with an empty document. If you specify *filename*, that file is displayed in the Editor.

You may have more than one Do-file Editor open at once. Each time you submit the `doedit` command, a new window will be opened.

A tutorial discussion of `doedit` can be found in the *Getting Started with Stata* manual. Read [\[U\] 16 Do-files](#) for an explanation of do-files, and then read [\[GSW\] 13 Using the Do-file Editor—automating Stata](#) to learn how to use the Do-file Editor to create and execute do-files.

## Also see

[\[GSM\] 13 Using the Do-file Editor—automating Stata](#)

[\[GSU\] 13 Using the Do-file Editor—automating Stata](#)

[\[GSW\] 13 Using the Do-file Editor—automating Stata](#)

[\[U\] 16 Do-files](#)

Syntax

*Dotplot of varname, with one column per value of groupvar*

```
dotplot varname [if] [in] [, options]
```

*Dotplot for each variable in varlist, with one column per variable*

```
dotplot varlist [if] [in] [, options]
```

| options                                 | description                                                                  |
|-----------------------------------------|------------------------------------------------------------------------------|
| Options                                 |                                                                              |
| over(groupvar)                          | display one columnar dotplot for each value of <i>groupvar</i>               |
| nx(#)                                   | horizontal dot density; default is nx(0)                                     |
| ny(#)                                   | vertical dot density; default is ny(35)                                      |
| incr(#)                                 | label every # group; default is incr(1)                                      |
| mean median                             | plot a horizontal line of pluses at the mean or median                       |
| bounded                                 | use minimum and maximum as boundaries                                        |
| bar                                     | plot horizontal dashed lines at shoulders of each group                      |
| nogroup                                 | use the actual values of <i>yvar</i>                                         |
| center                                  | center the dot for each column                                               |
| Plot                                    |                                                                              |
| marker_options                          | change look of markers (color, size, etc.)                                   |
| marker_label_options                    | add marker labels; change look or position                                   |
| Y axis, X axis, Titles, Legend, Overall |                                                                              |
| twoway_options                          | any options other than by() documented in [G] <a href="#">twoway_options</a> |

Menu

Graphics > Distributional graphs > Distribution dotplot

Description

A dotplot is a scatterplot with values grouped together vertically (“binning”, as in a histogram) and with plotted points separated horizontally. The aim is to display all the data for several variables or groups in one compact graphic.

In the first syntax, `dotplot` produces a columnar dotplot of *varname*, with one column per value of *groupvar*. In the second syntax, `dotplot` produces a columnar dotplot for each variable in *varlist*, with one column per variable; `over(groupvar)` is not allowed. In each case, the “dots” are plotted as small circles to increase readability.

## Options

### Options

`over(groupvar)` identifies the variable for which `dotplot` will display one columnar dotplot for each value of *groupvar*.

`nx(#)` sets the horizontal dot density. A larger value of `#` will increase the dot density, reducing the horizontal separation between dots. This option will increase the separation between columns if two or more groups or variables are used.

`ny(#)` sets the vertical dot density (number of “bins” on the *y* axis). A larger value of `#` will result in more bins and a plot that is less spread out horizontally. `#` should be determined in conjunction with `nx()` to give the most pleasing appearance.

`incr(#)` specifies how the *x* axis is to be labeled. `incr(1)`, the default, labels all groups. `incr(2)` labels every second group.

`[mean|median]` plots a horizontal line of pluses at the mean or median of each group.

`bounded` forces the minimum and maximum of the variable to be used as boundaries of the smallest and largest bins. It should be used with one variable whose support is not the whole of the real line and whose density does not tend to zero at the ends of its support, e.g., a uniform random variable or an exponential random variable.

`bar` plots horizontal dashed lines at the shoulders of each group. The “shoulders” are taken to be the upper and lower quartiles unless `mean` has been specified; here they will be the mean plus or minus the standard deviation.

`nogroup` uses the actual values of *yvar* rather than grouping them (the default). This option may be useful if *yvar* takes on only a few values.

`center` centers the dots for each column on a hidden vertical line.

### Plot

*marker\_options* affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [\[G\] marker\\_options](#).

*marker\_label\_options* specify if and how the markers are to be labeled; see [\[G\] marker\\_label\\_options](#).

### Y axis, X axis, Titles, Legend, Overall

*twoway\_options* are any of the options documented in [\[G\] twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [\[G\] title\\_options](#)) and for saving the graph to disk (see [\[G\] saving\\_option](#)).

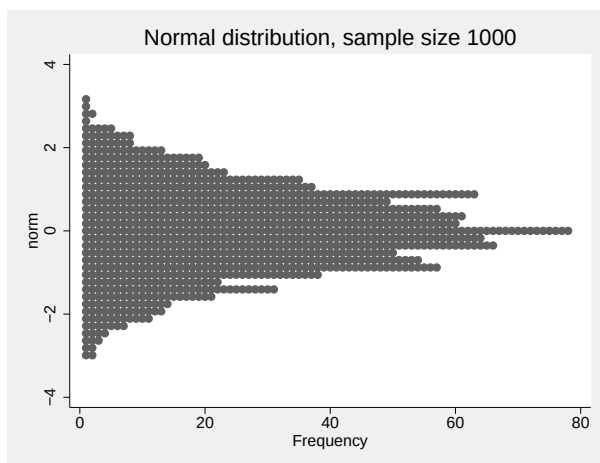
## Remarks

`dotplot` produces a figure that has elements of a boxplot, a histogram, and a scatterplot. Like a boxplot, it is most useful for comparing the distributions of several variables or the distribution of 1 variable in several groups. Like a histogram, the figure provides a crude estimate of the density, and, as with a scatterplot, each symbol (dot) represents 1 observation.

## ► Example 1

`dotplot` may be used as an alternative to Stata's histogram graph for displaying the distribution of one variable.

```
. set seed 123456789
. set obs 1000
. generate norm = rnormal()
. dotplot norm, title("Normal distribution, sample size 1000")
```

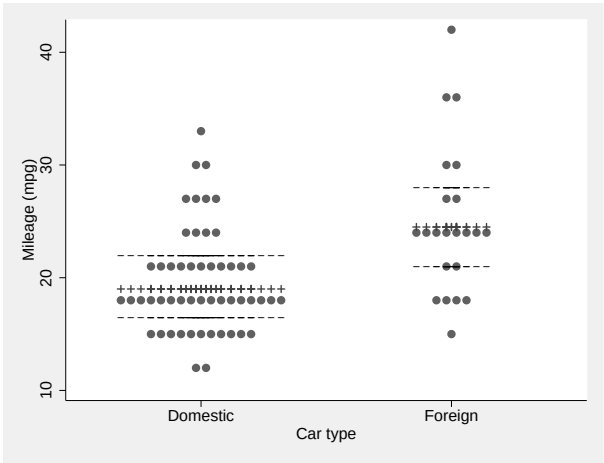


## ► Example 2

The `over()` option lets us use `dotplot` to compare the distribution of one variable within different levels of a grouping variable. The `center`, `median`, and `bar` options create a graph that may be compared with Stata's boxplot; see [\[G\] graph box](#). The next graph illustrates this option with Stata's automobile dataset.

*(Continued on next page)*

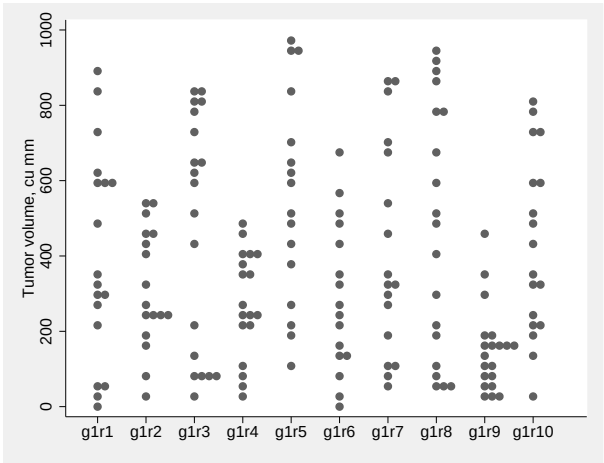
```
. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)
. dotplot mpg, over(foreign) nx(25) ny(10) center median bar
```



Example 3

The second version of dotplot lets us compare the distribution of several variables. In the next graph, all 10 variables contain measurements on tumor volume.

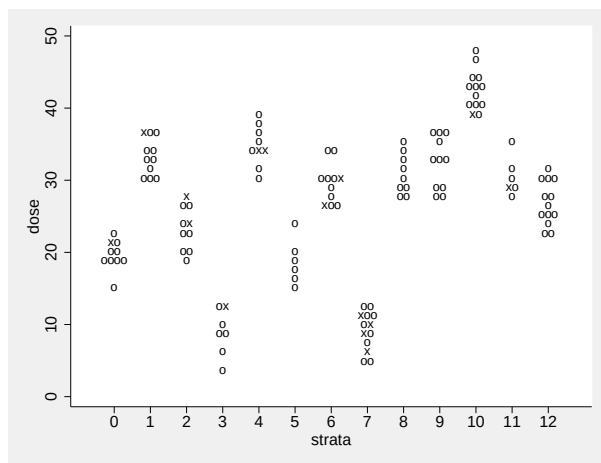
```
. use http://www.stata-press.com/data/r11/dotgr
. dotplot g1r1-g1r10, ytitle("Tumor volume, cu mm")
```



## ► Example 4

When using the first form with the `over()` option, we can encode a third dimension in a dotplot by using a different plotting symbol for different groups. The third dimension cannot be encoded with a varlist. The example is of a hypothetical matched case–control study. The next graph shows the exposure of each individual in each matched stratum. Cases are marked by the letter ‘x’, and controls are marked by the letter ‘o’.

```
. use http://www.stata-press.com/data/r11/dotdose
. label define symbol 0 "o" 1 "x"
. label values case symbol
. dotplot dose, over(strata) m(none) mlabel(case) mlabp(0) center
```



◀

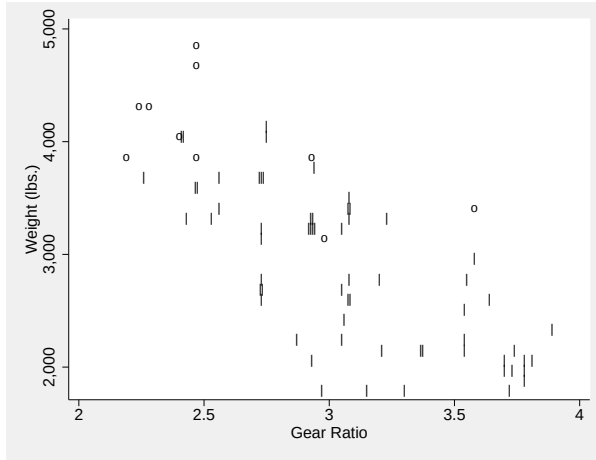
## ► Example 5

`dotplot` can also be used with two virtually continuous variables as an alternative to jittering the data to distinguish ties. We must use the `xlab` option, because otherwise `dotplot` will attempt to label too many points on the  $x$  axis. It is often useful in such instances to use a value of `nx` that is smaller than the default. That was not necessary in this example, partly because of our choice of symbols.

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. generate byte hi_price = (price>10000) if price < .
. label define symbol 0 "|" 1 "o"
. label values hi_price symbol
```

(Continued on next page)

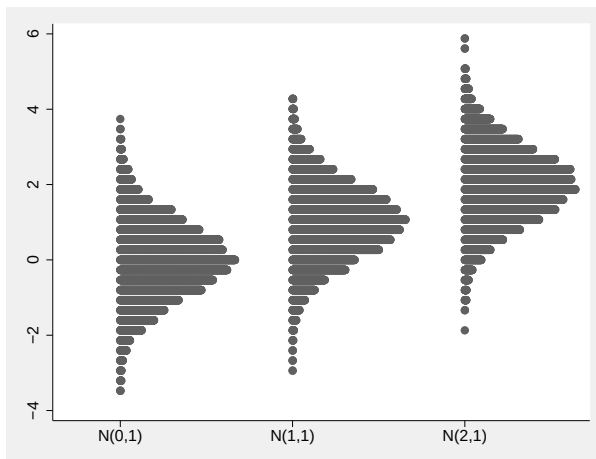
```
. dotplot weight, over(gear_ratio) m(none) mlabel(hi_price) mlabp(0) center
> xlabel(#5)
```



## ► Example 6

The following figure is included mostly for aesthetic reasons. It also demonstrates `dotplot`'s ability to cope with even very large datasets. The sample size for each variable is 10,000, so it may take a long time to print.

```
. clear all
. set seed 123456789
. set obs 10000
. gen norm0 = rnormal()
. gen norm1 = rnormal() + 1
. gen norm2 = rnormal() + 2
. label variable norm0 "N(0,1)"
. label variable norm1 "N(1,1)"
. label variable norm2 "N(2,1)"
. dotplot norm0 norm1 norm2
```





## Saved results

`dotplot` saves the following in `r()`:

Scalars

`r(nx)` horizontal dot density

`r(ny)` vertical dot density

## Methods and formulas

`dotplot` is implemented as an ado-file.

## Acknowledgments

`dotplot` was written by Peter Sasieni of the Wolfson Institute of Preventive Medicine, London, and Patrick Royston of the MRC Clinical Trials Unit, London.

## References

- Sasieni, P., and P. Royston. 1994. [gr14: dotplot: Comparative scatterplots](#). *Stata Technical Bulletin* 19: 8–10. Reprinted in *Stata Technical Bulletin Reprints*, vol. 4, pp. 50–54. College Station, TX: Stata Press.
- . 1996. Dotplots. *Applied Statistics* 45: 219–234.

Syntax

Direct standardization

```
dstdize charvar popvar stratavars [if] [in] , by(groupvars) [dstdize_options]
```

Indirect standardization

```
istdize casevars popvars stratavars [if] [in] using filename,
{ popvars(casevarp popvarp) | rate(ratevarp {# | crudevarp}) }
[istdize_options]
```

| dstdize_options  | description                                                       |
|------------------|-------------------------------------------------------------------|
| Main             |                                                                   |
| *by(groupvars)   | study populations                                                 |
| using(filename)  | use standard population from Stata dataset                        |
| base(#   string) | use standard population from a value of grouping variable         |
| level(#)         | set confidence level; default is level(95)                        |
| Options          |                                                                   |
| saving(filename) | save computed standard population distribution as a Stata dataset |
| format(%fmt)     | final summary table display format; default is %10.0g             |
| print            | include table summary of standard population in output            |
| nores            | suppress saving results in r()                                    |

\*by(groupvars) is required.

| istdize_options                                          | description                                                                                                            |
|----------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| Main                                                     |                                                                                                                        |
| *popvars(casevar <sub>p</sub> popvar <sub>p</sub> )      | for standard population, casevar <sub>p</sub> is number of cases and popvar <sub>p</sub> is number of individuals      |
| *rate(ratevar <sub>p</sub> {#   crudevar <sub>p</sub> }) | ratevar <sub>p</sub> is stratum-specific rates and # or crudevar <sub>p</sub> is the crude case rate value or variable |
| level(#)                                                 | set confidence level; default is level(95)                                                                             |
| Options                                                  |                                                                                                                        |
| by(groupvars)                                            | variables identifying study populations                                                                                |
| format(%fmt)                                             | final summary table display format; default is %10.0g                                                                  |
| print                                                    | include table summary of standard population in output                                                                 |

\*Either popvars(casevar<sub>p</sub> popvar<sub>p</sub>) or rate(ratevar<sub>p</sub> {# | crudevar<sub>p</sub>}) must be specified.

## Menu

### dstdize

Statistics > Epidemiology and related > Other > Direct standardization

### istdize

Statistics > Epidemiology and related > Other > Indirect standardization

## Description

`dstdize` produces standardized rates for *charvar*, which are defined as a weighted average of the stratum-specific rates. These rates can be used to compare the characteristic *charvar* across different populations identified by *groupvars*. Weights used in the standardization are given by *popvar*; the strata across which the weights are to be averaged are defined by *stratavars*.

`istdize` produces indirectly standardized rates for a study population based on a standard population. This standardization method is appropriate when the stratum-specific rates for the population being studied are either unavailable or based on small samples and thus are unreliable. The standardization uses the stratum-specific rates of a standard population to calculate the expected number of cases in the study population(s), sums them, and then compares them with the actual number of cases observed. The standard population is in another Stata data file specified by using *filename*, and it must contain *popvar* and *stratavars*.

In addition to calculating rates, the indirect standardization command produces point estimates and exact confidence intervals of the study population's standardized mortality ratio (SMR), if death is the event of interest, or the standardized incidence ratio (SIR) for studies of incidence. Here we refer to both ratios as SMR.

*casevar<sub>s</sub>* is the variable name for the study population's number of cases (usually deaths). It must contain integers, and for each group, defined by *groupvar*, each subpopulation identified by *stratavars* must have the same values or missing.

*popvar<sub>s</sub>* identifies the number of subjects represented by each observation in the study population. *stratavars* define the strata.

## Options for dstdize

### Main

`by(groupvars)` is required for the `dstdize` command; it specifies the variables identifying the study populations. If `base()` is also specified, there must be only one variable in the `by()` group. If you do not have a variable for this option, you can generate one by using something like `gen newvar=1` and then use `newvar` as the argument to this option.

`using(filename)` or `base(#|string)` may be used to specify the standard population. You may not specify both options. `using(filename)` supplies the name of a `.dta` file containing the standard population. The standard population must contain the *popvar* and the *stratavars*. If `using()` is not specified, the standard population distribution will be obtained from the data. `base(#|string)` lets you specify one of the values of *groupvar*—either a numeric value or a string—to be used as the standard population. If neither `base()` nor `using()` is specified, the entire dataset is used to determine an estimate of the standard population.

`level(#)` specifies the confidence level, as a percentage, for a confidence interval of the adjusted rate. The default is `level(95)` or as set by `set level`; see [U] 20.7 Specifying the width of confidence intervals.

## Options

`saving(filename)` saves the computed standard population distribution as a Stata dataset that can be used in further analyses.

`format(%fmt)` specifies the format in which to display the final summary table. The default is `%10.0g`.

`print` includes a table summary of the standard population before displaying the study population results.

`nores` suppresses saving results in `r()`. This option is seldom specified. Some saved results are stored in matrices. If there are more groups than `matsize`, `dstdize` will report “matsize too small”. Then you can either increase `matsize` or specify `nores`. The `nores` option does not change how results are calculated but specifies that results need not be left behind for use by other programs.

## Options for `istdize`

## Main

`popvars(casevarp popvarp)` or `rate(ratevarp #|ratevarp crudevarp)` must be specified with the `istdize` command. Only one of these two options is allowed. These options are used to describe the standard population’s data.

With `popvars(casevarp popvarp)`, `casevarp` records the number of cases (deaths) for each stratum in the standard population, and `popvarp` records the total number of individuals in each stratum (individuals at risk).

With `rate(ratevarp {#|crudevarp})`, `ratevarp` contains the stratum-specific rates. `#|crudevarp` specifies the crude case rate either by a variable name or, optionally, by the crude case rate value. If a crude rate variable is used, it must be the same for all observations, although it could be missing for some.

`level(#)` specifies the confidence level, as a percentage, for a confidence interval of the adjusted rate. The default is `level(95)` or as set by `set level`; see [U] 20.7 Specifying the width of confidence intervals.

## Options

`by(groupvars)` specifies variables identifying study populations when more than one exists in the data. If this option is not specified, the entire study population is treated as one group.

`format(%fmt)` specifies the format in which to display the final summary table. The default is `%10.0g`.

`print` outputs a table summary of the standard population before displaying the study population results.

## Remarks

Remarks are presented under the following headings:

*Direct standardization*

*Indirect standardization*

In epidemiology and other fields, you will often need to compare rates for some characteristic across different populations. These populations often differ on factors associated with the characteristic under study; thus directly comparing overall rates may be misleading.

See van Belle et al. (2004, 642–684), [Fleiss, Levin, and Paik \(2003, chap. 19\)](#), or [Kirkwood and Sterne \(2003, chap. 25\)](#) for a discussion of direct and indirect standardization.

## Direct standardization

The direct method of adjusting for differences among populations involves computing the overall rates that would result if, instead of having different distributions, all populations had the same standard distribution. The standardized rate is defined as a weighted average of the stratum-specific rates, with the weights taken from the standard distribution. Direct standardization may be applied only when the specific rates for a given population are available.

`dstdize` generates adjusted summary measures of occurrence, which can be used to compare prevalence, incidence, or mortality rates between populations that may differ on certain characteristics (e.g., age, gender, race). These underlying differences may affect the crude prevalence, mortality, or incidence rates.

### ► Example 1

We have data ([Rothman 1986, 42](#)) on mortality rates for Sweden and Panama for 1962, and we wish to compare mortality in these two countries:

```
. use http://www.stata-press.com/data/r11/mortality
(1962 Mortality, Sweden & Panama)

. describe

Contains data from http://www.stata-press.com/data/r11/mortality.dta
 obs: 6 1962 Mortality, Sweden & Panama
 vars: 4 14 Apr 2009 16:18
 size: 114 (99.9% of memory free)
```

| variable name | storage<br>type | display<br>format | value<br>label | variable label             |
|---------------|-----------------|-------------------|----------------|----------------------------|
| nation        | str6            | %9s               |                | Nation                     |
| age_category  | byte            | %9.0g             | age_lbl        | Age Category               |
| population    | float           | %10.0gc           |                | Population in Age Category |
| deaths        | float           | %9.0gc            |                | Deaths in Age Category     |

Sorted by:

```
. list, sepby(nation) abbrev(12) divider
```

|    | nation | age_category | population | deaths |
|----|--------|--------------|------------|--------|
| 1. | Sweden | 0 - 29       | 3145000    | 3,523  |
| 2. | Sweden | 30 - 59      | 3057000    | 10,928 |
| 3. | Sweden | 60+          | 1294000    | 59,104 |
| 4. | Panama | 0 - 29       | 741,000    | 3,904  |
| 5. | Panama | 30 - 59      | 275,000    | 1,421  |
| 6. | Panama | 60+          | 59,000     | 2,456  |

We divide the total number of cases in the population by the population to obtain the *crude rate*:

```
. collapse (sum) pop deaths, by(nation)
. list, abbrev(10) divider
```

|    | nation | population | deaths |
|----|--------|------------|--------|
| 1. | Panama | 1075000    | 7,781  |
| 2. | Sweden | 7496000    | 73,555 |

```
. generate crude = deaths/pop
. list, abbrev(10) divider
```

|    | nation | population | deaths | crude    |
|----|--------|------------|--------|----------|
| 1. | Panama | 1075000    | 7,781  | .0072381 |
| 2. | Sweden | 7496000    | 73,555 | .0098126 |

If we examine the total number of deaths in the two nations, the total crude mortality rate in Sweden is higher than that in Panama. From the original data, we see one possible explanation: Swedes are older than Panamanians, making direct comparison of the mortality rates difficult.

Direct standardization lets us remove the distortion caused by the different age distributions. The adjusted rate is defined as the weighted sum of the crude rates, where the weights are given by the standard distribution. Suppose that we wish to standardize these mortality rates to the following age distribution:

```
. use http://www.stata-press.com/data/r11/1962, clear
(Standard Population Distribution)
. list, abbrev(12) divider
```

|    | age_category | population |
|----|--------------|------------|
| 1. | 0 - 29       | .35        |
| 2. | 30 - 59      | .35        |
| 3. | 60+          | .3         |

```
. sort age_cat
. save 1962
file 1962.dta saved
```

If we multiply the above weights for the age strata by the crude rate for the corresponding age category, the sum gives us the standardized rate.

```
. use http://www.stata-press.com/data/r11/mortality
(1962 Mortality, Sweden & Panama)
. generate crude=deaths/pop
. drop pop
. sort age_cat
. merge m:1 age_cat using 1962
age_category was byte now float
```

| Result      | # of obs.     |
|-------------|---------------|
| not matched | 0             |
| matched     | 6 (_merge==3) |

```
. list, sepby(age_category) abbrev(12)
```

|    | nation | age_category | deaths | crude    | population | _merge      |
|----|--------|--------------|--------|----------|------------|-------------|
| 1. | Sweden | 0 - 29       | 3,523  | .0011202 | .35        | matched (3) |
| 2. | Panama | 0 - 29       | 3,904  | .0052686 | .35        | matched (3) |
| 3. | Panama | 30 - 59      | 1,421  | .0051673 | .35        | matched (3) |
| 4. | Sweden | 30 - 59      | 10,928 | .0035747 | .35        | matched (3) |
| 5. | Panama | 60+          | 2,456  | .0416271 | .3         | matched (3) |
| 6. | Sweden | 60+          | 59,104 | .0456754 | .3         | matched (3) |

```
. generate product = crude*pop
. by nation, sort: egen adj_rate = sum(product)
. drop _merge
. list, sepby(nation)
```

|    | nation | age_cat-y | deaths | crude    | popula~n | product  | adj_rate |
|----|--------|-----------|--------|----------|----------|----------|----------|
| 1. | Panama | 0 - 29    | 3,904  | .0052686 | .35      | .001844  | .0161407 |
| 2. | Panama | 30 - 59   | 1,421  | .0051673 | .35      | .0018085 | .0161407 |
| 3. | Panama | 60+       | 2,456  | .0416271 | .3       | .0124881 | .0161407 |
| 4. | Sweden | 60+       | 59,104 | .0456754 | .3       | .0137026 | .0153459 |
| 5. | Sweden | 30 - 59   | 10,928 | .0035747 | .35      | .0012512 | .0153459 |
| 6. | Sweden | 0 - 29    | 3,523  | .0011202 | .35      | .0003921 | .0153459 |

Comparing the standardized rates indicates that the Swedes have a slightly lower mortality rate.

(Continued on next page)

To perform the above analysis with dstdize, type

```
. use http://www.stata-press.com/data/r11/mortality, clear
(1962 Mortality, Sweden & Panama)
. dstdize deaths pop age_cat, by(nation) using(1962)
```

-> nation= Panama

| Stratum | Pop.    | Cases | ——Unadjusted——                       |         | Std.<br>Pop. Stratum<br>Dist. Rate[s] | Pop.<br>Dst[P] | s*P |
|---------|---------|-------|--------------------------------------|---------|---------------------------------------|----------------|-----|
|         |         |       | Pop.                                 | Stratum |                                       |                |     |
| 0 - 29  | 741000  | 3904  | 0.689                                | 0.0053  | 0.350                                 | 0.0018         |     |
| 30 - 59 | 275000  | 1421  | 0.256                                | 0.0052  | 0.350                                 | 0.0018         |     |
| 60+     | 59000   | 2456  | 0.055                                | 0.0416  | 0.300                                 | 0.0125         |     |
| Totals: | 1075000 | 7781  | Adjusted Cases: 17351.2              |         |                                       |                |     |
|         |         |       | Crude Rate: 0.0072                   |         |                                       |                |     |
|         |         |       | Adjusted Rate: 0.0161                |         |                                       |                |     |
|         |         |       | 95% Conf. Interval: [0.0156, 0.0166] |         |                                       |                |     |

-> nation= Sweden

| Stratum | Pop.    | Cases | ——Unadjusted——                       |         | Std.<br>Pop. Stratum<br>Dist. Rate[s] | Pop.<br>Dst[P] | s*P |
|---------|---------|-------|--------------------------------------|---------|---------------------------------------|----------------|-----|
|         |         |       | Pop.                                 | Stratum |                                       |                |     |
| 0 - 29  | 3145000 | 3523  | 0.420                                | 0.0011  | 0.350                                 | 0.0004         |     |
| 30 - 59 | 3057000 | 10928 | 0.408                                | 0.0036  | 0.350                                 | 0.0013         |     |
| 60+     | 1294000 | 59104 | 0.173                                | 0.0457  | 0.300                                 | 0.0137         |     |
| Totals: | 7496000 | 73555 | Adjusted Cases: 115032.5             |         |                                       |                |     |
|         |         |       | Crude Rate: 0.0098                   |         |                                       |                |     |
|         |         |       | Adjusted Rate: 0.0153                |         |                                       |                |     |
|         |         |       | 95% Conf. Interval: [0.0152, 0.0155] |         |                                       |                |     |

Summary of Study Populations:

| nation | N       | Crude    | Adj_Rate | Confidence Interval |           |
|--------|---------|----------|----------|---------------------|-----------|
| Panama | 1075000 | 0.007238 | 0.016141 | [ 0.015645,         | 0.016637] |
| Sweden | 7496000 | 0.009813 | 0.015346 | [ 0.015235,         | 0.015457] |

The summary table above lets us make a quick inspection of the results within the study populations, and the detail tables give the behavior among the strata within the study populations.

➤ Example 2

We have individual-level data on persons in four cities over several years. Included in the data is a variable indicating whether the person has high blood pressure, together with information on the person’s age, sex, and race. We wish to obtain standardized high blood pressure rates for each city for 1990 and 1992, using, as the standard, the age, sex, and race distribution of the four cities and two years combined.



Our dataset contains

```
. use http://www.stata-press.com/data/r11/hbp
. describe
```

```
Contains data from http://www.stata-press.com/data/r11/hbp.dta
 obs: 1,130
 vars: 7
 size: 23,730 (99.9% of memory free)
21 Feb 2009 06:42
```

| variable name | storage<br>type | display<br>format | value<br>label | variable label               |
|---------------|-----------------|-------------------|----------------|------------------------------|
| id            | str10           | %10s              |                | Record identification number |
| city          | byte            | %8.0g             |                |                              |
| year          | int             | %8.0g             |                |                              |
| sex           | byte            | %8.0g             | sexfmt         |                              |
| age_group     | byte            | %8.0g             | agefmt         |                              |
| race          | byte            | %8.0g             | racefmt        |                              |
| hbp           | byte            | %8.0g             | yn             | high blood pressure          |

Sorted by:

The `dstdize` command is designed to work with aggregate data but will work with individual-level data only if we create a variable recording the population represented by each observation. For individual-level data, this is one:

```
. gen pop = 1
```

On the next page, we specify `print` to obtain a listing of the standard population and `level(90)` to request 90% rather than 95% confidence intervals. Typing `if year==1990 | year==1992` restricts the data to the two years for both summary tables and the standard population.

*(Continued on next page)*



-> city year= 5 1992

|         |          |        |    | ——Unadjusted—— |                                      |                 | Std.    |        |     |
|---------|----------|--------|----|----------------|--------------------------------------|-----------------|---------|--------|-----|
|         |          |        |    | Pop.           | Stratum                              | Pop.            |         |        |     |
| Stratum |          |        |    | Pop.           | Cases                                | Dist.           | Rate[s] | Dst[P] | s*P |
| 15 - 19 | Black    | Female | 6  | 0              | 0.087                                | 0.0000          | 0.077   | 0.0000 |     |
| 15 - 19 | Black    | Male   | 9  | 0              | 0.130                                | 0.0000          | 0.097   | 0.0000 |     |
| 15 - 19 | Hispanic | Female | 1  | 0              | 0.014                                | 0.0000          | 0.011   | 0.0000 |     |
| 15 - 19 | Hispanic | Male   | 2  | 0              | 0.029                                | 0.0000          | 0.022   | 0.0000 |     |
| 15 - 19 | White    | Female | 2  | 0              | 0.029                                | 0.0000          | 0.015   | 0.0000 |     |
| 15 - 19 | White    | Male   | 1  | 0              | 0.014                                | 0.0000          | 0.011   | 0.0000 |     |
| 20 - 24 | Black    | Female | 13 | 0              | 0.188                                | 0.0000          | 0.095   | 0.0000 |     |
| 20 - 24 | Black    | Male   | 10 | 0              | 0.145                                | 0.0000          | 0.147   | 0.0000 |     |
| 20 - 24 | Hispanic | Male   | 1  | 0              | 0.014                                | 0.0000          | 0.029   | 0.0000 |     |
| 20 - 24 | White    | Male   | 3  | 0              | 0.043                                | 0.0000          | 0.046   | 0.0000 |     |
| 25 - 29 | Black    | Female | 2  | 0              | 0.029                                | 0.0000          | 0.037   | 0.0000 |     |
| 25 - 29 | Black    | Male   | 2  | 0              | 0.029                                | 0.0000          | 0.097   | 0.0000 |     |
| 25 - 29 | Hispanic | Male   | 3  | 0              | 0.043                                | 0.0000          | 0.029   | 0.0000 |     |
| 25 - 29 | White    | Male   | 1  | 0              | 0.014                                | 0.0000          | 0.035   | 0.0000 |     |
| 30 - 34 | Black    | Female | 4  | 0              | 0.058                                | 0.0000          | 0.035   | 0.0000 |     |
| 30 - 34 | Black    | Male   | 5  | 0              | 0.072                                | 0.0000          | 0.070   | 0.0000 |     |
| 30 - 34 | Hispanic | Male   | 2  | 0              | 0.029                                | 0.0000          | 0.007   | 0.0000 |     |
| 30 - 34 | White    | Female | 1  | 0              | 0.014                                | 0.0000          | 0.011   | 0.0000 |     |
| 30 - 34 | White    | Male   | 1  | 1              | 0.014                                | 1.0000          | 0.051   | 0.0505 |     |
| Totals: |          |        |    | 69             | 1                                    | Adjusted Cases: |         | 3.5    |     |
|         |          |        |    |                |                                      | Crude Rate:     |         | 0.0145 |     |
|         |          |        |    |                |                                      | Adjusted Rate:  |         | 0.0505 |     |
|         |          |        |    |                | 90% Conf. Interval: [0.0505, 0.0505] |                 |         |        |     |

#### Summary of Study Populations:

| city<br>year | N  | Crude    | Adj_Rate | Confidence Interval |                     |
|--------------|----|----------|----------|---------------------|---------------------|
| 1            |    |          |          |                     |                     |
| 1990         | 47 | 0.063830 | 0.041758 | [                   | 0.007427, 0.076089] |
| 1            |    |          |          |                     |                     |
| 1992         | 56 | 0.017857 | 0.008791 | [                   | 0.000000, 0.022579] |
| 2            |    |          |          |                     |                     |
| 1990         | 64 | 0.046875 | 0.044898 | [                   | 0.009072, 0.080724] |
| 2            |    |          |          |                     |                     |
| 1992         | 67 | 0.029851 | 0.014286 | [                   | 0.002537, 0.026035] |
| 3            |    |          |          |                     |                     |
| 1990         | 69 | 0.159420 | 0.088453 | [                   | 0.050093, 0.126813] |
| 3            |    |          |          |                     |                     |
| 1992         | 37 | 0.189189 | 0.046319 | [                   | 0.025271, 0.067366] |
| 5            |    |          |          |                     |                     |
| 1990         | 46 | 0.043478 | 0.022344 | [                   | 0.002044, 0.042644] |
| 5            |    |          |          |                     |                     |
| 1992         | 69 | 0.014493 | 0.050549 | [                   | 0.050549, 0.050549] |

## Indirect standardization

Standardization of rates can be performed via the indirect method whenever the stratum-specific rates are either unknown or unreliable. If the stratum-specific rates are known, the direct standardization method is preferred.

- To apply the indirect method, you must have the following information:
- The observed number of cases in each population to be standardized,  $O$ . For example, if death rates in two states are being standardized using the U.S. death rate for the same period, you must know the total number of deaths in each state.
  - The distribution across the various strata for the population being studied,  $n_1, \dots, n_k$ . If you are standardizing the death rate in the two states, adjusting for age, you must know the number of individuals in each of the  $k$  age groups.
  - The stratum-specific rates for the standard population,  $p_1, \dots, p_k$ . For example, you must have the U.S. death rate for each stratum (age group).
  - The crude rate of the standard population,  $C$ . For example, you must have the U.S. mortality rate for the year.

The indirect adjusted rate is then

$$R_{\text{indirect}} = C \frac{O}{E}$$

where  $E$  is the expected number of cases (deaths) in each population. See [Methods and formulas](#) for a more detailed description of calculations.

### ► Example 3

This example is borrowed from [Kahn and Sempos \(1989, 95–105\)](#). We want to compare 1970 mortality rates in California and Maine, adjusting for age. Although we have age-specific population counts for the two states, we lack age-specific death rates. Direct standardization is not feasible here. We can use the U.S. population census data for the same year to produce indirectly standardized rates for these two states.

From the U.S. census, the standard population for this example was entered into Stata and saved in `popkahn.dta`.

```
. use http://www.stata-press.com/data/r11/popkahn, clear
. list age pop deaths rate, sep(4)
```

|    | age   | population | deaths  | rate   |
|----|-------|------------|---------|--------|
| 1. | <15   | 57,900,000 | 103,062 | .00178 |
| 2. | 15–24 | 35,441,000 | 45,261  | .00128 |
| 3. | 25–34 | 24,907,000 | 39,193  | .00157 |
| 4. | 35–44 | 23,088,000 | 72,617  | .00315 |
| 5. | 45–54 | 23,220,000 | 169,517 | .0073  |
| 6. | 55–64 | 18,590,000 | 308,373 | .01659 |
| 7. | 65–74 | 12,436,000 | 445,531 | .03583 |
| 8. | 75+   | 7,630,000  | 736,758 | .09656 |

The standard population contains for each age stratum the total number of individuals (`pop`) and both the age-specific mortality rate (`rate`) and the number of deaths. The standard population need not contain all three. If we have only the age-specific mortality rate, we can use the `rate(ratevarp, crudevarp)` or `rate(ratevarp, #)` option, where `crudevarp` refers to the variable containing the total population's crude death rate or `#` is the total population's crude death rate.

Now let's look at the states' data (study population):

```
. use http://www.stata-press.com/data/r11/kahn
. list, sep(4)
```

|     | state      | age   | populat-n | death   | st | death_~e |
|-----|------------|-------|-----------|---------|----|----------|
| 1.  | California | <15   | 5,524,000 | 166,285 | 1  | .0016    |
| 2.  | California | 15-24 | 3,558,000 | 166,285 | 1  | .0013    |
| 3.  | California | 25-34 | 2,677,000 | 166,285 | 1  | .0015    |
| 4.  | California | 35-44 | 2,359,000 | 166,285 | 1  | .0028    |
| 5.  | California | 45-54 | 2,330,000 | 166,285 | 1  | .0067    |
| 6.  | California | 55-64 | 1,704,000 | 166,285 | 1  | .0154    |
| 7.  | California | 65-74 | 1,105,000 | 166,285 | 1  | .0328    |
| 8.  | California | 75+   | 696,000   | 166,285 | 1  | .0917    |
| 9.  | Maine      | <15   | 286,000   | 11,051  | 2  | .0019    |
| 10. | Maine      | 15-24 | 168,000   | .       | 2  | .0011    |
| 11. | Maine      | 25-34 | 110,000   | .       | 2  | .0014    |
| 12. | Maine      | 35-44 | 109,000   | .       | 2  | .0029    |
| 13. | Maine      | 45-54 | 110,000   | .       | 2  | .0069    |
| 14. | Maine      | 55-64 | 94,000    | .       | 2  | .0173    |
| 15. | Maine      | 65-74 | 69,000    | .       | 2  | .039     |
| 16. | Maine      | 75+   | 46,000    | .       | 2  | .1041    |

For each state, the number of individuals in each stratum (age group) is contained in the `pop` variable. The `death` variable is the total number of deaths observed in the state during the year. It must have the same value for all observations in the group, as for California, or it could be missing in all but one observation per group, as for Maine.

To match these two datasets, the strata variables must have the same name in both datasets and ideally the same levels. If a level is missing from either dataset, that level will not be included in the standardization.

With `kahn.dta` in memory, we now execute the command. We will use the `print` option to obtain the standard population's summary table, and because we have both the standard population's age-specific count and deaths, we will specify the `popvars(casevarp popvarp)` option. Or, we could specify the `rate(rate 0.00945)` option because we know that 0.00945 is the U.S. crude death rate for 1970.

(Continued on next page)

```
. istdize death pop age using http://www.stata-press.com/data/r11/popkahn,
> by(state) pop(deaths pop) print
```

| Standard Population |         |
|---------------------|---------|
| Stratum             | Rate    |
| <15                 | 0.00178 |
| 15-24               | 0.00128 |
| 25-34               | 0.00157 |
| 35-44               | 0.00315 |
| 45-54               | 0.00730 |
| 55-64               | 0.01659 |
| 65-74               | 0.03583 |
| 75+                 | 0.09656 |

Standard population’s crude rate: 0.00945

```
-> state= California
```

| Indirect Standardization |                 |          |          |
|--------------------------|-----------------|----------|----------|
| Stratum                  | Standard        | Observed | Cases    |
|                          | Population Rate |          |          |
| <15                      | 0.0018          | 5524000  | 9832.72  |
| 15-24                    | 0.0013          | 3558000  | 4543.85  |
| 25-34                    | 0.0016          | 2677000  | 4212.46  |
| 35-44                    | 0.0031          | 2359000  | 7419.59  |
| 45-54                    | 0.0073          | 2330000  | 17010.10 |
| 55-64                    | 0.0166          | 1704000  | 28266.14 |
| 65-74                    | 0.0358          | 1105000  | 39587.63 |
| 75+                      | 0.0966          | 696000   | 67206.23 |

Totals: 19953000 178078.73

Observed Cases: 166285  
SMR (Obs/Exp): 0.93  
SMR exact 95% Conf. Interval: [0.9293, 0.9383]  
Crude Rate: 0.0083  
Adjusted Rate: 0.0088  
95% Conf. Interval: [0.0088, 0.0089]

```
-> state= Maine
```

| Stratum | Indirect Standardization       |                        |                   |
|---------|--------------------------------|------------------------|-------------------|
|         | Standard<br>Population<br>Rate | Observed<br>Population | Cases<br>Expected |
| <15     | 0.0018                         | 286000                 | 509.08            |
| 15-24   | 0.0013                         | 168000                 | 214.55            |
| 25-34   | 0.0016                         | 110000                 | 173.09            |
| 35-44   | 0.0031                         | 109000                 | 342.83            |
| 45-54   | 0.0073                         | 110000                 | 803.05            |
| 55-64   | 0.0166                         | 94000                  | 1559.28           |
| 65-74   | 0.0358                         | 69000                  | 2471.99           |
| 75+     | 0.0966                         | 46000                  | 4441.79           |
| Totals: |                                | 992000                 | 10515.67          |

```

Observed Cases: 11051
SMR (Obs/Exp): 1.05
SMR exact 95% Conf. Interval: [1.0314, 1.0707]
Crude Rate: 0.0111
Adjusted Rate: 0.0099
95% Conf. Interval: [0.0097, 0.0101]
```

Summary of Study Populations (Rates):

| state      | Cases    |          |          |                      |
|------------|----------|----------|----------|----------------------|
|            | Observed | Crude    | Adj_Rate | Confidence Interval  |
| California | 166285   | 0.008334 | 0.008824 | [0.008782, 0.008866] |
| Maine      | 11051    | 0.011140 | 0.009931 | [0.009747, 0.010118] |

Summary of Study Populations (SMR):

| state      | Cases    |           | SMR   | Exact                |  |
|------------|----------|-----------|-------|----------------------|--|
|            | Observed | Expected  |       | Confidence Interval  |  |
| California | 166285   | 178078.73 | 0.934 | [0.929290, 0.938271] |  |
| Maine      | 11051    | 10515.67  | 1.051 | [1.031405, 1.070687] |  |

◀

## Saved results

`dstdize` saves the following in `r()`:

Scalars

`r(k)`      number of populations

Macros

`r(by)`      variable names specified in `by()`

`r(c#)`      values of `r(by)` for `#th` group

`r(se)`      standard errors of adjusted rates

`r(ub)`      upper bounds of confidence intervals for adjusted rates

`r(lb)`      lower bounds of confidence intervals for adjusted rates

Matrices

`r(Nobs)`     $1 \times k$  vector of number of observations

`r(crude)`    $1 \times k$  vector of crude rates (\*)

`r(adj)`      $1 \times k$  vector of adjusted rates (\*)

(\*) If, in a group, the number of observations is 0, then 9 is stored for the corresponding crude and adjusted rates.

## Methods and formulas

`dstdize` and `istdize` are implemented as ado-files.

The directly standardized rate,  $S_R$ , is defined by

$$S_R = \frac{\sum_{i=1}^k w_i R_i}{\sum_{i=1}^k w_i}$$

(Rothman 1986, 44), where  $R_i$  is the stratum-specific rate in stratum  $i$  and  $w_i$  is the weight for stratum  $i$  derived from the standard population.

If  $n_i$  is the population of stratum  $i$ , the standard error,  $\text{se}(S_R)$ , in stratified sampling for proportions (ignoring the finite population correction) is

$$\text{se}(S_R) = \frac{1}{\sum w_i} \sqrt{\sum_{i=1}^k \frac{w_i^2 R_i (1 - R_i)}{n_i}}$$

(Cochran 1977, 108), from which the confidence intervals are calculated.

For indirect standardization, define  $O$  as the observed number of cases in each population to be standardized;  $n_1, \dots, n_k$  as the distribution across the various strata for the population being studied;  $R_1, \dots, R_k$  as the stratum-specific rates for the standard population; and  $C$  as the crude rate of the standard population. The expected number of cases (deaths),  $E$ , in each population is obtained by applying the standard population stratum-specific rates,  $R_1, \dots, R_k$ , to the study populations:

$$E = \sum_{i=1}^k n_i R_i$$

The indirectly adjusted rate is then

$$R_{\text{indirect}} = C \frac{O}{E}$$

and  $O/E$  is the study population's SMR if death is the event of interest or the SIR for studies of disease (or other) incidence.

The exact confidence interval is calculated for each estimated SMR by assuming a Poisson process as described in Breslow and Day (1987, 69–71). These intervals are obtained by first calculating the upper and lower bounds for the confidence interval of the Poisson-distributed observed events,  $O$ —say,  $L$  and  $U$ , respectively—and then computing  $\text{SMR}_L = L/E$  and  $\text{SMR}_U = U/E$ .

## Acknowledgments

We gratefully acknowledge the collaboration of Dr. Joel A. Harrison, CIGNA HealthCare of Texas; Dr. José Maria Pacheco from the Departamento de Epidemiologia, Faculdade de Saúde Pública/USP, Sao Paulo, Brazil; and Dr John L. Moran from The Queen Elizabeth Hospital, Woodville, Australia.



## References

- Breslow, N. E., and N. E. Day. 1987. *Statistical Methods in Cancer Research: Vol. 2—The Design and Analysis of Cohort Studies*. Lyon: IARC.
- Cleves, M. A. 1998. [sg80: Indirect standardization](#). *Stata Technical Bulletin* 42: 43–47. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 224–228. College Station, TX: Stata Press.
- Cochran, W. G. 1977. *Sampling Techniques*. 3rd ed. New York: Wiley.
- Fleiss, J. L., B. Levin, and M. C. Paik. 2003. *Statistical Methods for Rates and Proportions*. 3rd ed. New York: Wiley.
- Forthofer, R. N., and E. S. Lee. 1995. *Introduction to Biostatistics: A Guide to Design, Analysis, and Discovery*. New York: Academic Press.
- Juul, S. 2008. *An Introduction to Stata for Health Researchers*. 2nd ed. College Station, TX: Stata Press.
- Kahn, H. A., and C. T. Sempos. 1989. *Statistical Methods in Epidemiology*. New York: Oxford University Press.
- Kirkwood, B. R., and J. A. C. Sterne. 2003. *Essential Medical Statistics*. 2nd ed. Malden, MA: Blackwell.
- McGuire, T. J., and J. A. Harrison. 1994. [sbe11: Direct standardization](#). *Stata Technical Bulletin* 21: 5–9. Reprinted in *Stata Technical Bulletin Reprints*, vol. 4, pp. 88–94. College Station, TX: Stata Press.
- Pagano, M., and K. Gauvreau. 2000. *Principles of Biostatistics*. 2nd ed. Belmont, CA: Duxbury.
- Rothman, K. J. 1986. *Modern Epidemiology*. Boston: Little, Brown.
- van Belle, G., L. D. Fisher, P. J. Heagerty, and T. S. Lumley. 2004. *Biostatistics: A Methodology for the Health Sciences*. 2nd ed. New York: Wiley.
- Wang, D. 2000. [sbe40: Modeling mortality data using the Lee–Carter model](#). *Stata Technical Bulletin* 57: 15–17. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 118–121. College Station, TX: Stata Press.

## Also see

[\[ST\] epitab](#) — Tables for epidemiologists

[\[SVY\] direct standardization](#) — Direct standardization of means, proportions, and ratios

# Title

dydx — Calculate numeric derivatives and integrals

## Syntax

*Derivatives of numeric functions*

dydx yvar xvar [*if*] [*in*] , generate(*newvar*) [*dydx\_options*]

*Integrals of numeric functions*

integ yvar xvar [*if*] [*in*] [, *integ\_options*]

| <i>dydx_options</i> | description |
|---------------------|-------------|
|---------------------|-------------|

Main

|                            |                                     |
|----------------------------|-------------------------------------|
| *generate( <i>newvar</i> ) | create variable named <i>newvar</i> |
| replace                    | overwrite the existing variable     |

\*generate(*newvar*) is required.

| <i>integ_options</i> | description |
|----------------------|-------------|
|----------------------|-------------|

Main

|                           |                                                                     |
|---------------------------|---------------------------------------------------------------------|
| generate( <i>newvar</i> ) | create variable named <i>newvar</i>                                 |
| trapezoid                 | use trapezoidal rule to compute integrals; default is cubic splines |
| initial( <i>#</i> )       | initial value of integral; default is 0                             |
| replace                   | overwrite the existing variable                                     |

by is allowed with dydx and integ; see [\[D\]](#) **by**.

## Menu

### dydx

Data > Create or change data > Other variable-creation commands > Calculate numerical derivatives

### integ

Data > Create or change data > Other variable-creation commands > Calculate numeric integrals

## Description

dydx and integ calculate derivatives and integrals of numeric “functions”.

## Options

### Main

`generate(newvar)` specifies the name of the new variable to be created. It must be specified with `dydx`.

`trapezoid` requests that the trapezoidal rule [the sum of  $(x_i - x_{i-1})(y_i + y_{i-1})/2$ ] be used to compute integrals. The default is cubic splines, which give superior results for most smooth functions; for irregular functions, `trapezoid` may give better results.

`initial(#)` specifies the initial condition for calculating definite integrals; see [Methods and formulas](#) below. If not specified, the initial condition is taken as 0.

`replace` specifies that if an existing variable is specified for `generate()`, it should be overwritten.

## Remarks

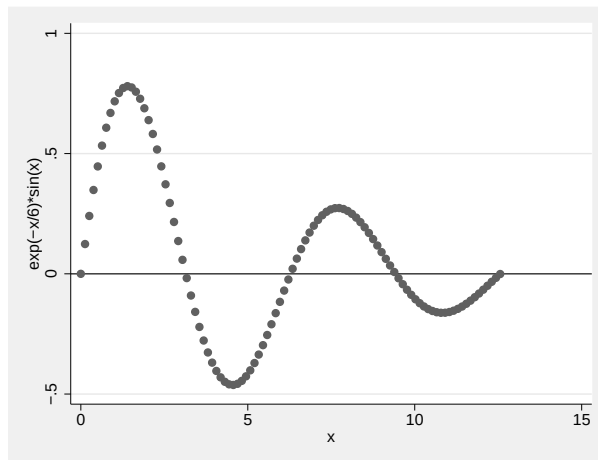
`dydx` and `integ` lets you extend Stata's graphics capabilities beyond data analysis and into mathematics. (See [Gould \[1993\]](#) for another command that draws functions.)

### ► Example 1

We graph  $y = e^{-x/6}\sin(x)$  over the interval  $[0, 12.56]$ :

```
. range x 0 12.56 100
obs was 0, now 100

. generate y = exp(-x/6)*sin(x)
. label variable y "exp(-x/6)*sin(x)"
. twoway connected y x, connect(i) yline(0)
```

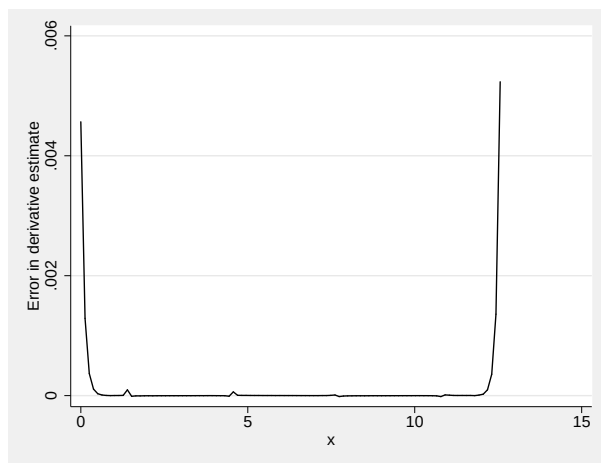


We estimate the derivative by using `dydx` and compute the relative difference between this estimate and the true derivative.

```
. dydx y x, gen(dy)
. generate dytrue = exp(-x/6)*(cos(x) - sin(x)/6)
. generate error = abs(dy - dytrue)/dytrue
```

The error is greatest at the endpoints, as we would expect. The error is approximately 0.5% at each endpoint, but the error quickly falls to less than 0.01%.

```
. label variable error "Error in derivative estimate"
. twoway line error x, ylabel(0(.002).006)
```

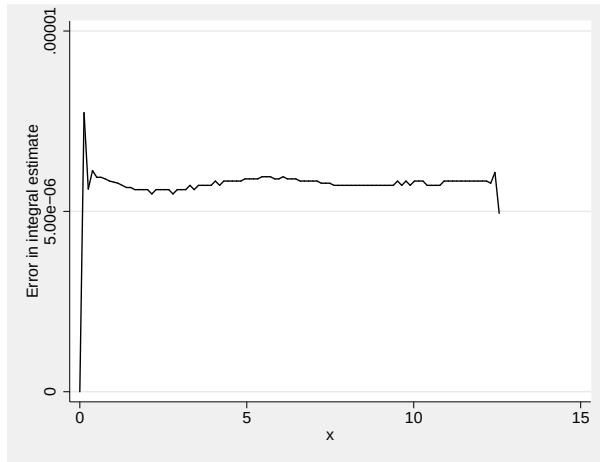


We now estimate the integral by using `integ`:

```
. integ y x, gen(iy)
number of points = 100
integral = .85316396
. generate iytrue = (36/37)*(1 - exp(-x/6)*(cos(x) + sin(x)/6))
. display iytrue[_N]
.85315901
. display abs(r(integral) - iytrue[_N])/iytrue[_N]
5.799e-06
. generate diff = iy - iytrue
```

The relative difference between the estimate [stored in `r(integral)`] and the true value of the integral is about  $6 \times 10^{-6}$ . A graph of the absolute difference (`diff`) is shown below. Here error is cumulative. Again most of the error is due to a relatively poorer fit near the endpoints.

```
. label variable diff "Error in integral estimate"
. twoway line diff x, ylabel(0(5.00e-06).00001)
```



4

## Saved results

`dydx` saves the following in `r()`:

Macros  
`r(y)` name of *yvar*

`integ` saves the following in `r()`:

Scalars  
`r(N_points)` number of unique *x* points  
`r(integral)` estimate of the integral

## Methods and formulas

`dydx` and `integ` are implemented as ado-files.

Consider a set of data points,  $(x_1, y_1), \dots, (x_n, y_n)$ , generated by a function  $y = f(x)$ . `dydx` and `integ` first fit these points with a cubic spline, which is then analytically differentiated (integrated) to give an approximation for the derivative (integral) of  $f$ .

The cubic spline (see, for example, Press et al. [2007]) consists of  $n - 1$  cubic polynomials  $P_i(x)$ , with the  $i$ th one defined on the interval  $[x_i, x_{i+1}]$ ,

$$P_i(x) = y_i a_i(x) + y_{i+1} b_i(x) + y_i'' c_i(x) + y_{i+1}'' d_i(x)$$

where

$$\begin{aligned} a_i(x) &= \frac{x_{i+1} - x}{x_{i+1} - x_i} & b_i(x) &= \frac{x - x_i}{x_{i+1} - x_i} \\ c_i(x) &= \frac{1}{6}(x_{i+1} - x_i)^2 a_i(x) [\{a_i(x)\}^2 - 1] & d_i(x) &= \frac{1}{6}(x_{i+1} - x_i)^2 b_i(x) [\{b_i(x)\}^2 - 1] \end{aligned}$$

and  $y_i''$  and  $y_{i+1}''$  are constants whose values will be determined as described below. The notation for these constants is justified because  $P_i''(x_i) = y_i''$  and  $P_i''(x_{i+1}) = y_{i+1}''$ .

Because  $a_i(x_i) = 1$ ,  $a_i(x_{i+1}) = 0$ ,  $b_i(x_i) = 0$ , and  $b_i(x_{i+1}) = 1$ . Therefore,  $P_i(x_i) = y_i$ , and  $P_i(x_{i+1}) = y_{i+1}$ . Thus the  $P_i$  jointly define a function that is continuous at the interval boundaries. The first derivative should be continuous at the interval boundaries; that is,

$$P_i'(x_{i+1}) = P_{i+1}'(x_{i+1})$$

The above  $n - 2$  equations (one equation for each point except the two endpoints) and the values of the first derivative at the endpoints,  $P_1'(x_1)$  and  $P_{n-1}'(x_n)$ , determine the  $n$  constants  $y_i''$ .

The value of the first derivative at an endpoint is set to the value of the derivative obtained by fitting a quadratic to the endpoint and the two adjacent points; namely, we use

$$P_1'(x_1) = \frac{y_1 - y_2}{x_1 - x_2} + \frac{y_1 - y_3}{x_1 - x_3} - \frac{y_2 - y_3}{x_2 - x_3}$$

and a similar formula for the upper endpoint.

`dydx` approximates  $f'(x_i)$  by using  $P_i'(x_i)$ .

`integ` approximates  $F(x_i) = F(x_1) + \int_{x_1}^{x_i} f(x) dx$  by using

$$I_0 + \sum_{k=1}^{i-1} \int_{x_k}^{x_{k+1}} P_k(x) dx$$

where  $I_0$  (an estimate of  $F(x_1)$ ) is the value specified by the `initial(#)` option. If the `trapezoid` option is specified, `integ` approximates the integral by using the trapezoidal rule:

$$I_0 + \sum_{k=1}^{i-1} \frac{1}{2} (x_{k+1} - x_k) (y_{k+1} + y_k)$$

If there are ties among the  $x_i$ , the mean of  $y_i$  is computed at each set of ties and the cubic spline is fit to these values.

## Acknowledgment

The present versions of `dydx` and `integ` were inspired by the `dydx2` command written by Patrick Royston of the MRC Clinical Trials Unit, London.

## References

- Gould, W. W. 1993. [ssi5.1: Graphing functions](#). *Stata Technical Bulletin* 16: 23–26. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 188–193. College Station, TX: Stata Press.
- . 1997. [crc46: Better numerical derivatives and integrals](#). *Stata Technical Bulletin* 35: 3–5. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 8–12. College Station, TX: Stata Press.
- Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 2007. *Numerical Recipes in C: The Art of Scientific Computing*. 3rd ed. Cambridge: Cambridge University Press.

## Also see

- [D] [obs](#) — Increase the number of observations in a dataset
- [D] [range](#) — Generate numerical range

# Title

*eform\_option* — Displaying exponentiated coefficients

## Description

An *eform\_option* causes the coefficient table to be displayed in exponentiated form: for each coefficient,  $e^b$  rather than  $b$  is displayed. Standard errors and confidence intervals (CIs) are also transformed. Display of the intercept, if any, is suppressed.

An *eform\_option* is one of the following:

| <i>eform_option</i>        | description                                                     |
|----------------------------|-----------------------------------------------------------------|
| <code>eform(string)</code> | use <i>string</i> for the column title                          |
| <code>eform</code>         | exponentiated coefficient, <i>string</i> is <code>exp(b)</code> |
| <code>hr</code>            | hazard ratio, <i>string</i> is <code>Haz. Ratio</code>          |
| <code>shr</code>           | subhazard ratio, <i>string</i> is <code>SHR</code>              |
| <code>irr</code>           | incidence-rate ratio, <i>string</i> is <code>IRR</code>         |
| <code>or</code>            | odds ratio, <i>string</i> is <code>Odds Ratio</code>            |
| <code>rrr</code>           | relative-risk ratio, <i>string</i> is <code>RRR</code>          |

## Remarks

### ► Example 1

Here is a simple example of the `or` option with `svy: logit`. The CI for the odds ratio is computed by transforming (by exponentiating) the endpoints of the CI for the corresponding coefficient.

```
. use http://www.stata-press.com/data/r11/nhanes2d
. svy, or: logit highbp female black
(running logit on estimation sample)
(output omitted)
```

| highbp | Linearized |           | t     | P> t  | [95% Conf. Interval] |          |
|--------|------------|-----------|-------|-------|----------------------|----------|
|        | Odds Ratio | Std. Err. |       |       |                      |          |
| female | .693628    | .048676   | -5.21 | 0.000 | .6011298             | .8003593 |
| black  | 1.509155   | .2089569  | 2.97  | 0.006 | 1.137872             | 2.001586 |

We also could have specified the following command and received the same results as above:

```
. svy: logit highbp female black, or
```



## Also see

[R] [ml](#) — Maximum likelihood estimation



# Title

eivreg — Errors-in-variables regression

## Syntax

```
eivreg depvar [indepvars] [if] [in] [weight]
[, reliab(indepvar # [indepvar # [...]]) level(#) display_options coeflegend]
```

† coeflegend does not appear in the dialog box.

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*bootstrap*, *by*, *jackknife*, *rolling*, and *statsby* are allowed; see [U] 11.1.10 Prefix commands.

Weights are not allowed with the *bootstrap* prefix.

*aweight*s are not allowed with the *jackknife* prefix.

*aweight*s and *fweight*s are allowed; see [U] 11.1.6 *weight*.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

## Menu

Statistics > Linear models and related > Errors-in-variables regression

## Description

**eivreg** fits errors-in-variables regression models.

## Options

### Model

**reliab**(*indepvar* # [*indepvar* # [...] ]) specifies the measurement reliability for each independent variable measured with error. Reliabilities are specified as pairs consisting of an independent variable name (a name that appears in *indepvars*) and the corresponding reliability  $r$ ,  $0 < r \leq 1$ . Independent variables for which no reliability is specified are assumed to have reliability 1. If the option is not specified, all variables are assumed to have reliability 1, and the result is thus the same as that produced by **regress** (the ordinary least-squares results).

### Reporting

**level**(#); see [R] estimation options.

**display\_options**: noomitted, vsquish, noemptycells, baselevels, allbaselevels; see [R] estimation options.

The following option is available with **eivreg** but is not shown in the dialog box:

**coeflegend**; see [R] estimation options.

## Remarks

For an introduction to errors-in-variables regression, see [Draper and Smith \(1998, 89–91\)](#) or [Kmenta \(1997, 352–357\)](#). [Treiman \(2009, 258–261\)](#) compares the results of errors-in-variables regression with conventional regression.

Errors-in-variables regression models are useful when one or more of the independent variables are measured with additive noise. Standard regression (as performed by `regress`) would underestimate the effect of the variable, and the other coefficients in the model can be biased to the extent that they are correlated with the poorly measured variable. You can adjust for the biases if you know the reliability:

$$r = 1 - \frac{\text{noise variance}}{\text{total variance}}$$

That is, given the model  $y = X\beta + u$ , for some variable  $x_i$  in  $X$ , the  $x_i$  is observed with error,  $x_i = x_i^* + e$ , and the noise variance is the variance of  $e$ . The total variance is the variance of  $x_i$ .

### ► Example 1

Say that in our automobile data, the weight of cars was measured with error, and the reliability of our measured weight is 0.85. The result of this would be to underestimate the effect of `weight` in a regression of, say, `price` on `weight` and `foreign`, and it would also bias the estimate of the coefficient on `foreign` (because being of foreign manufacture is correlated with the weight of cars). We would ignore all of this if we fit the model with `regress`:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. regress price weight foreign
```

| Source   | SS        | df | MS         | Number of obs = 74     |  |  |
|----------|-----------|----|------------|------------------------|--|--|
| Model    | 316859273 | 2  | 158429637  | F( 2, 71) = 35.35      |  |  |
| Residual | 318206123 | 71 | 4481776.38 | Prob > F = 0.0000      |  |  |
|          |           |    |            | R-squared = 0.4989     |  |  |
|          |           |    |            | Adj R-squared = 0.4848 |  |  |
| Total    | 635065396 | 73 | 8699525.97 | Root MSE = 2117        |  |  |

| price   | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|---------|-----------|-----------|-------|-------|----------------------|-----------|
| weight  | 3.320737  | .3958784  | 8.39  | 0.000 | 2.531378             | 4.110096  |
| foreign | 3637.001  | 668.583   | 5.44  | 0.000 | 2303.885             | 4970.118  |
| _cons   | -4942.844 | 1345.591  | -3.67 | 0.000 | -7625.876            | -2259.812 |

With `eivreg`, we can account for our measurement error:

```
. eivreg price weight foreign, r(weight .85)
```

| variable | assumed<br>reliability | Errors-in-variables regression |  |  |  |  |
|----------|------------------------|--------------------------------|--|--|--|--|
| weight   | 0.8500                 | Number of obs = 74             |  |  |  |  |
| *        | 1.0000                 | F( 2, 71) = 50.37              |  |  |  |  |
|          |                        | Prob > F = 0.0000              |  |  |  |  |
|          |                        | R-squared = 0.6483             |  |  |  |  |
|          |                        | Root MSE = 1773.54             |  |  |  |  |

| price   | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|---------|-----------|-----------|-------|-------|----------------------|-----------|
| weight  | 4.31985   | .431431   | 10.01 | 0.000 | 3.459601             | 5.180099  |
| foreign | 4637.32   | 624.5362  | 7.43  | 0.000 | 3392.03              | 5882.609  |
| _cons   | -8257.017 | 1452.086  | -5.69 | 0.000 | -11152.39            | -5361.639 |

The effect of weight is increased—as we knew it would be—and here the effect of foreign manufacture is also increased. A priori, we knew only that the estimate of `foreign` might be biased; we did not know the direction.



## □ Technical note

Swept under the rug in our example is how we would determine the reliability,  $r$ . We can easily see that a variable is measured with error, but we may not know the reliability because the ingredients for calculating  $r$  depend on the unobserved noise.

For our example, we made up a value for  $r$ , and in fact we do not believe that weight is measured with error at all, so the reported `eivreg` results have no validity. The `regress` results were the statistically correct results here.

But let's say that we do suspect that weight is measured with error and that we do not know  $r$ . We could then experiment with various values of  $r$  to describe the sensitivity of our estimates to possible error levels. We may not know  $r$ , but  $r$  does have a simple interpretation, and we could probably produce a sensible range for  $r$  by thinking about how the data were collected.

If the reliability,  $r$ , is less than the  $R^2$  from a regression of the poorly measured variable on all the other variables, including the dependent variable, the information might as well not have been collected; no adjustment to the final results is possible. For our automobile data, running a regression of `weight` on `foreign` and `price` would result in an  $R^2$  of 0.6743. Thus the reliability must be at least 0.6743 here. If we specify a reliability that is too small, `eivreg` will inform us and refuse to fit the model:

```
. eivreg price weight foreign, r(weight .6742)
reliability r() too small
r(399);
```

Returning to our problem of how to estimate  $r$ , too small or not, if the measurements are summaries of scaled items, the reliability may be estimated using the `alpha` command; see [\[R\] alpha](#). If the score is computed from factor analysis and the data are scored using `predict`'s default options (see [\[MV\] factor postestimation](#)), the square of the standard deviation of the score is an estimate of the reliability.



## □ Technical note

Consider a model with more than one variable measured with error. For instance, say that our model is that `price` is a function of `weight`, `foreign`, and `mpg` and that both `weight` and `mpg` are measured with error.

(Continued on next page)

```
. eivreg price weight foreign mpg, r(weight .85 mpg .9)
```

| variable | assumed<br>reliability | Errors-in-variables regression |
|----------|------------------------|--------------------------------|
| weight   | 0.8500                 | Number of obs = 74             |
| mpg      | 0.9000                 | F( 3, 70) = 429.14             |
| *        | 1.0000                 | Prob > F = 0.0000              |
|          |                        | R-squared = 0.9728             |
|          |                        | Root MSE = 496.41              |

| price   | Coef.     | Std. Err. | t      | P> t  | [95% Conf. Interval] |          |
|---------|-----------|-----------|--------|-------|----------------------|----------|
| weight  | 12.88302  | .6820532  | 18.89  | 0.000 | 11.52271             | 14.24333 |
| foreign | 8268.951  | 352.8719  | 23.43  | 0.000 | 7565.17              | 8972.732 |
| mpg     | 999.2043  | 73.60037  | 13.58  | 0.000 | 852.413              | 1145.996 |
| _cons   | -56473.19 | 3710.015  | -15.22 | 0.000 | -63872.58            | -49073.8 |



# Saved results

eivreg saves the following in e():

- Scalars
- e(N)

number of observations
- e(df\_m)

model degrees of freedom
- e(df\_r)

residual degrees of freedom
- e(r2)

R-squared
- e(F)

F statistic
- e(rmse)

root mean squared error
- e(rank)

rank of e(V)
- Macros
- e(cmd)

eivreg
- e(cmdline)

command as typed
- e(depvar)

name of dependent variable
- e(rellist)

indepvars and associated reliabilities
- e(wtype)

weight type
- e(wexp)

weight expression
- e(properties)

b V
- e(predict)

program used to implement predict
- e(asbalanced)

factor variables fvset as asbalanced
- e(asobserved)

factor variables fvset as asobserved
- Matrices
- e(b)

coefficient vector
- e(V)

variance-covariance matrix of the estimators
- Functions
- e(sample)

marks estimation sample

## Methods and formulas

`eivreg` is implemented as an ado-file.

Let the model to be fit be

$$\begin{aligned} y &= \mathbf{X}^* \beta + e \\ \mathbf{X} &= \mathbf{X}^* + \mathbf{U} \end{aligned}$$

where  $\mathbf{X}^*$  are the true values and  $\mathbf{X}$  are the observed values. Let  $\mathbf{W}$  be the user-specified weights. If no weights are specified,  $\mathbf{W} = \mathbf{I}$ . If weights are specified, let  $\mathbf{v}$  be the specified weights. If `fweight` frequency weights are specified, then  $\mathbf{W} = \text{diag}(\mathbf{v})$ . If `aweight` analytic weights are specified, then  $\mathbf{W} = \text{diag}\{\mathbf{v}/(\mathbf{1}'\mathbf{v})(\mathbf{1}'\mathbf{1})\}$ , meaning that the weights are normalized to sum to the number of observations.

The estimates  $\mathbf{b}$  of  $\beta$  are obtained as  $\mathbf{A}^{-1}\mathbf{X}'\mathbf{W}\mathbf{y}$ , where  $\mathbf{A} = \mathbf{X}'\mathbf{W}\mathbf{X} - \mathbf{S}$ .  $\mathbf{S}$  is a diagonal matrix with elements  $N(1 - r_i)s_i^2$ .  $N$  is the number of observations,  $r_i$  is the user-specified reliability coefficient for the  $i$ th explanatory variable or 1 if not specified, and  $s_i^2$  is the (appropriately weighted) variance of the variable.

The variance–covariance matrix of the estimators is obtained as  $s^2\mathbf{A}^{-1}\mathbf{X}'\mathbf{W}\mathbf{X}\mathbf{A}^{-1}$ , where the root mean squared error  $s^2 = (\mathbf{y}'\mathbf{W}\mathbf{y} - \mathbf{b}\mathbf{A}\mathbf{b}')/(N - p)$ , where  $p$  is the number of estimated parameters.

## References

- Draper, N., and H. Smith. 1998. *Applied Regression Analysis*. 3rd ed. New York: Wiley.
- Kmenta, J. 1997. *Elements of Econometrics*. 2nd ed. Ann Arbor: University of Michigan Press.
- Treiman, D. J. 2009. *Quantitative Data Analysis: Doing Social Research to Test Ideas*. San Francisco, CA: Jossey-Bass.

## Also see

- [R] [eivreg postestimation](#) — Postestimation tools for `eivreg`
- [R] [regress](#) — Linear regression
- [U] [20 Estimation and postestimation commands](#)

## Description

The following postestimation commands are available for `eivreg`:

| command                | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>     | VCE and estimation sample summary                                                                   |
| <code>estimates</code> | cataloging estimation results                                                                       |
| <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>  | link test for model specification                                                                   |
| <code>margins</code>   | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>   | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code> | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details.

## Syntax for predict

```
predict [type] newvar [if] [in] [, statistic]
```

| <i>statistic</i>                      | description                                 |
|---------------------------------------|---------------------------------------------|
| Main                                  |                                             |
| <code>xb</code>                       | linear prediction; the default              |
| <code>residuals</code>                | residuals                                   |
| <code>stdp</code>                     | standard error of the prediction            |
| <code>stdf</code>                     | standard error of the forecast              |
| <code>pr(<i>a</i>,<i>b</i>)</code>    | $\Pr(a < y_j < b)$                          |
| <code>e(<i>a</i>,<i>b</i>)</code>     | $E(y_j \mid a < y_j < b)$                   |
| <code>ystar(<i>a</i>,<i>b</i>)</code> | $E(y_j^*), y_j^* = \max\{a, \min(y_j, b)\}$ |

These statistics are available both in and out of sample; type `predict ... if e(sample)` ... if wanted only for the estimation sample.

where *a* and *b* may be numbers or variables; *a* missing (*a* ≥ .) means  $-\infty$ , and *b* missing (*b* ≥ .) means  $+\infty$ ; see [U] [12.2.1 Missing values](#).

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

**xb**, the default, calculates the linear prediction.

**residuals** calculates the residuals, that is,  $y_j - \mathbf{x}_j\mathbf{b}$ .

**stdp** calculates the standard error of the prediction, which can be thought of as the standard error of the predicted expected value or mean for the observation's covariate pattern. The standard error of the prediction is also referred to as the standard error of the fitted value.

**stdf** calculates the standard error of the forecast, which is the standard error of the point prediction for 1 observation and is commonly referred to as the standard error of the future or forecast value. By construction, the standard errors produced by **stdf** are always larger than those produced by **stdp**; see *Methods and formulas* in [R] **regress**.

**pr(*a*,*b*)** calculates  $\Pr(a < \mathbf{x}_j\mathbf{b} + u_j < b)$ , the probability that  $y_j|\mathbf{x}_j$  would be observed in the interval  $(a, b)$ .

*a* and *b* may be specified as numbers or variable names; *lb* and *ub* are variable names;

**pr(20,30)** calculates  $\Pr(20 < \mathbf{x}_j\mathbf{b} + u_j < 30)$ ;

**pr(*lb*,*ub*)** calculates  $\Pr(lb < \mathbf{x}_j\mathbf{b} + u_j < ub)$ ; and

**pr(20,*ub*)** calculates  $\Pr(20 < \mathbf{x}_j\mathbf{b} + u_j < ub)$ .

*a* missing (*a* ≥ .) means  $-\infty$ ; **pr(.,30)** calculates  $\Pr(-\infty < \mathbf{x}_j\mathbf{b} + u_j < 30)$ ;

**pr(*lb*,30)** calculates  $\Pr(-\infty < \mathbf{x}_j\mathbf{b} + u_j < 30)$  in observations for which *lb* ≥ . and calculates  $\Pr(lb < \mathbf{x}_j\mathbf{b} + u_j < 30)$  elsewhere.

*b* missing (*b* ≥ .) means  $+\infty$ ; **pr(20,.)** calculates  $\Pr(+\infty > \mathbf{x}_j\mathbf{b} + u_j > 20)$ ;

**pr(20,*ub*)** calculates  $\Pr(+\infty > \mathbf{x}_j\mathbf{b} + u_j > 20)$  in observations for which *ub* ≥ . and calculates  $\Pr(20 < \mathbf{x}_j\mathbf{b} + u_j < ub)$  elsewhere.

**e(*a*,*b*)** calculates  $E(\mathbf{x}_j\mathbf{b} + u_j | a < \mathbf{x}_j\mathbf{b} + u_j < b)$ , the expected value of  $y_j|\mathbf{x}_j$  conditional on  $y_j|\mathbf{x}_j$  being in the interval  $(a, b)$ , meaning that  $y_j|\mathbf{x}_j$  is censored. *a* and *b* are specified as they are for **pr()**.

**ystar(*a*,*b*)** calculates  $E(y_j^*)$ , where  $y_j^* = a$  if  $\mathbf{x}_j\mathbf{b} + u_j \leq a$ ,  $y_j^* = b$  if  $\mathbf{x}_j\mathbf{b} + u_j \geq b$ , and  $y_j^* = \mathbf{x}_j\mathbf{b} + u_j$  otherwise, meaning that  $y_j^*$  is truncated. *a* and *b* are specified as they are for **pr()**.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] **eivreg** — Errors-in-variables regression

[U] **20 Estimation and postestimation commands**

## Description

Whenever Stata detects that something is wrong—that what you typed is uninterpretable, that you are trying to do something you should not be trying to do, or that you requested the impossible—Stata responds by typing a message describing the problem, together with a *return code*. For instance,

```
. lsit
unrecognized command: lsit
r(199);

. list myvar
variable myvar not found
r(111);

. test a=b
last estimates not found
r(301);
```

In each case, the message is probably sufficient to guide you to a solution. When we typed `lsit`, Stata responded with “unrecognized command”. We meant to type `list`. When we typed `list myvar`, Stata responded with “variable myvar not found”. There is no variable named `myvar` in our data. When we typed `test a=b`, Stata responded with “last estimates not found”. `test` tests hypotheses about previously fit models, and we have not yet fit a model.

The numbers in parentheses in the `r(199)`, `r(111)`, and `r(301)` messages are called the *return codes*. To find out more about these messages, type `search rc #`, where `#` is the number returned in the parentheses.

### ► Example 1

```
. search rc 301
[P] error messages Return code 301
 last estimates not found;
 You typed an estimation command such as regress without arguments
 or attempted to perform a test or typed predict, but there were no
 previous estimation results.
```



Programmers should see [\[P\] error](#) for details on programming error messages.

## Also see

[\[R\] search](#) — Search Stata documentation



Syntax

Common subcommands

Obtain information criteria

```
estat ic [, n(#)]
```

Summarize estimation sample

```
estat summarize [eqlist] [, estat_summ_options]
```

Display covariance matrix estimates

```
estat vce [, estat_vce_options]
```

Command-specific subcommands

```
estat subcommand1 [, options1]
```

```
estat subcommand2 [, options2]
```

...

| <i>estat_summ_options</i>        | description                                                               |
|----------------------------------|---------------------------------------------------------------------------|
| <u>e</u> quation                 | display summary by equation                                               |
| <u>l</u> abels                   | display variable labels                                                   |
| <u>n</u> oheader                 | suppress the header                                                       |
| <u>n</u> oweight                 | ignore weights                                                            |
| <i>d</i> isplay_ <i>o</i> ptions | control spacing and display of omitted variables and base and empty cells |

*eqlist* is rarely used and specifies the variables, with optional equation name, to be summarized. *eqlist* may be *varlist* or (*eqname*<sub>1</sub>: *varlist*) (*eqname*<sub>2</sub>: *varlist*) ... *varlist* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

| <i>estat_vce_options</i>         | description                                                   |
|----------------------------------|---------------------------------------------------------------|
| <u>c</u> ovariance               | display as covariance matrix; the default                     |
| <u>c</u> orrelation              | display as correlation matrix                                 |
| equation( <i>spec</i> )          | display only specified equations                              |
| <u>b</u> lock                    | display submatrices by equation                               |
| <u>d</u> iag                     | display submatrices by equation; diagonal blocks only         |
| <u>f</u> ormat( <i>%fmt</i> )    | display format for covariances and correlations               |
| <u>n</u> olines                  | suppress lines between equations                              |
| <i>d</i> isplay_ <i>o</i> ptions | control display of omitted variables and base and empty cells |

## Menu

Statistics > Postestimation > Reports and statistics

## Description

`estat` displays scalar- and matrix-valued statistics after estimation; it complements `predict`, which calculates variables after estimation. Exactly what statistics `estat` can calculate depends on the previous estimation command.

Three sets of statistics are so commonly used that they are available after all estimation commands that store the model log likelihood. `estat ic` displays Akaike's and Schwarz's Bayesian information criteria. `estat summarize` summarizes the variables used by the command and automatically restricts the sample to `e(sample)`; it also summarizes the weight variable and cluster structure, if specified. `estat vce` displays the covariance or correlation matrix of the parameter estimates of the previous model.

## Option for estat ic

`n(#)` specifies the  $N$  to be used in calculating BIC; see [\[R\] BIC note](#).

## Options for estat summarize

`equation` requests that the dependent variables and the independent variables in the equations be displayed in the equation-style format of estimation commands, repeating the summary information about variables entered in more than one equation.

`labels` displays variable labels.

`noheader` suppresses the header.

`nweights` ignores the weights, if any, from the previous estimation command. The default when weights are present is to perform a weighted `summarize` on all variables except the weight variable itself. An unweighted `summarize` is performed on the weight variable.

*display\_options*: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [\[R\] estimation options](#).

## Options for estat vce

`covariance` displays the matrix as a variance–covariance matrix; this is the default.

`correlation` displays the matrix as a correlation matrix rather than a variance–covariance matrix. `rho` is a synonym.

`equation(spec)` selects part of the VCE to be displayed. If *spec* is *eqlist*, the VCE for the listed equations is displayed. If *spec* is *eqlist1* \ *eqlist2*, the part of the VCE associated with the equations in *eqlist1* (rowwise) and *eqlist2* (columnwise) is displayed. If *spec* is `*`, all equations are displayed. `equation()` implies `block` if `diag` is not specified.

`block` displays the submatrices pertaining to distinct equations separately.

`diag` displays the diagonal submatrices pertaining to distinct equations separately.

`format(%fmt)` specifies the number format for displaying the elements of the matrix. The default is `format(%10.0g)` for covariances and `format(%8.4f)` for correlations. See [U] [12.5 Formats: Controlling how data are displayed](#) for more information.

`nolines` suppresses lines between equations.

*display\_options*: `noomitted`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

## Remarks

`estat` displays a variety of scalar- and matrix-valued statistics after you have estimated the parameters of a model. Exactly what statistics `estat` can calculate depends on the estimation command used, and command-specific statistics are detailed in that command’s postestimation manual entry. The rest of this entry discusses three sets of statistics that are available after all estimation commands.

Remarks are presented under the following headings:

[estat ic](#)  
[estat summarize](#)  
[estat vce](#)

### estat ic

`estat ic` calculates two information criteria used to compare models. Unlike likelihood-ratio, Wald, and similar testing procedures, the models need not be nested to compare the information criteria. Because they are based on the log-likelihood function, information criteria are available only after commands that report the log likelihood.

In general, “smaller is better”: given two models, the one with the smaller AIC fits the data better than the one with the larger AIC. As with the AIC, a smaller BIC indicates a better-fitting model. For AIC and BIC formulas, see [Methods and formulas](#).

#### ► Example 1

In [R] [mlogit](#), we fit a model explaining the type of insurance a person has on the basis of age, gender, race, and site of study. Here we refit the model with and without the site dummies and compare the models.

```
. use http://www.stata-press.com/data/r11/sysdsn1
(Health insurance data)
. mlogit insure age male nonwhite
(output omitted)
. estat ic
```

| Model | Obs | ll(null)  | ll(model) | df | AIC      | BIC     |
|-------|-----|-----------|-----------|----|----------|---------|
| .     | 615 | -555.8545 | -545.5833 | 8  | 1107.167 | 1142.54 |

Note: N=Obs used in calculating BIC; see [R] [BIC note](#)

```
. mlogit insure age male nonwhite i.site
(output omitted)
```

```
. estat ic
```

| Model | Obs | ll(null)  | ll(model) | df | AIC      | BIC      |
|-------|-----|-----------|-----------|----|----------|----------|
| .     | 615 | -555.8545 | -534.3616 | 12 | 1092.723 | 1145.783 |

Note: N=Obs used in calculating BIC; see **[R] BIC note**

The AIC indicates that the model including the site dummies fits the data better, whereas the BIC indicates the opposite. As is often the case, different model-selection criteria have led to conflicting conclusions.

❑ Technical note

`glm` and `binreg`, `ml` report a slightly different version of AIC and BIC; see **[R] glm** for the formulas used. That version is commonly used within the GLM literature; see, for example, [Hardin and Hilbe \(2007\)](#). The literature on information criteria is vast; see, among others, [Akaike \(1973\)](#), [Sawa \(1978\)](#), and [Raftery \(1995\)](#). Judge et al. (1985) contains a discussion of using information criteria in econometrics. [Royston and Sauerbrei \(2008, chap. 2\)](#) examine the use of information criteria as an alternative to stepwise procedures for selecting model variables.

estat summarize

Often when fitting a model, you will also be interested in obtaining summary statistics, such as the sample means and standard deviations of the variables in the model. `estat summarize` makes this process simple. The output displayed is similar to that obtained by typing

```
. summarize varlist if e(sample)
```

without the need to type the *varlist* containing the dependent and independent variables.

➤ Example 2

Continuing with the previous multinomial logit model, here we summarize the variables by using `estat summarize`.

```
. estat summarize
```

| Estimation sample mlogit |          |           | Number of obs = | 615     |
|--------------------------|----------|-----------|-----------------|---------|
| Variable                 | Mean     | Std. Dev. | Min             | Max     |
| insure                   | 1.596748 | .6225846  | 1               | 3       |
| age                      | 44.46832 | 14.18523  | 18.1109         | 86.0725 |
| male                     | .2504065 | .4335998  | 0               | 1       |
| nonwhite                 | .196748  | .3978638  | 0               | 1       |
| site                     |          |           |                 |         |
| 2                        | .3707317 | .4833939  | 0               | 1       |
| 3                        | .3138211 | .4644224  | 0               | 1       |

The output in the previous example contains all the variables in one table, though `mlogit` presents its results in a multiple-equation format. For models in which the same variables appear in all equations, that is fine; but for other multiple-equation models, we may prefer to have the variables separated by the equation in which they appear. The `equation` option makes this possible.

► Example 3

Systems of simultaneous equations typically have different variables in each equation, and the `equation` option of `estat summarize` is helpful in such situations. In [example 2](#) of [\[R\] reg3](#), we have a model of supply and demand. We first refit the model and then call `estat summarize`.

```
. use http://www.stata-press.com/data/r11/supDem
. reg3 (Demand:quantity price pcompete income) (Supply:quantity price praw),
> endog(price)
(output omitted)
```

| . estat summarize, equation |          |           | Number of obs = |         | 49 |
|-----------------------------|----------|-----------|-----------------|---------|----|
| Estimation sample reg3      |          |           |                 |         |    |
| Variable                    | Mean     | Std. Dev. | Min             | Max     |    |
| depvar                      |          |           |                 |         |    |
| quantity                    | 12.61818 | 2.774952  | 7.71069         | 20.0477 |    |
| quantity                    | 12.61818 | 2.774952  | 7.71069         | 20.0477 |    |
| Demand                      |          |           |                 |         |    |
| price                       | 32.70944 | 2.882684  | 26.3819         | 38.4769 |    |
| pcompete                    | 5.929975 | 3.508264  | .207647         | 11.5549 |    |
| income                      | 7.811735 | 4.18859   | .570417         | 14.0077 |    |
| Supply                      |          |           |                 |         |    |
| price                       | 32.70944 | 2.882684  | 26.3819         | 38.4769 |    |
| praw                        | 4.740891 | 2.962565  | .151028         | 9.79881 |    |

The first block of the table contains statistics on the dependent (or, more accurately, left-hand-side) variables, and because we specified quantity as the left-hand-side variable in both equations, it is listed twice. The second block refers to the variables in the first equation we specified, which we labeled “Demand” in our call to `reg3`; and the final block refers to the supply equation.



estat vce

`estat vce` allows you to display the VCE of the parameters of the previously fit model, as either a covariance matrix or a correlation matrix.

(Continued on next page)

➤ Example 4

Returning to the mlogit example, we type

```
. use http://www.stata-press.com/data/r11/sysdsn1
(Health insurance data)

. mlogit insure age male nonwhite, nolog

Multinomial logistic regression Number of obs = 615
 LR chi2(6) = 20.54
 Prob > chi2 = 0.0022
Log likelihood = -545.58328 Pseudo R2 = 0.0185
```

| insure    | Coef.          | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-----------|----------------|-----------|-------|-------|----------------------|-----------|
| Indemnity | (base outcome) |           |       |       |                      |           |
| Prepaid   |                |           |       |       |                      |           |
| age       | -.0111915      | .0060915  | -1.84 | 0.066 | -.0231305            | .0007475  |
| male      | .5739825       | .2005221  | 2.86  | 0.004 | .1809665             | .9669985  |
| nonwhite  | .7312659       | .218978   | 3.34  | 0.001 | .302077              | 1.160455  |
| _cons     | .1567003       | .2828509  | 0.55  | 0.580 | -.3976773            | .7110778  |
| Uninsure  |                |           |       |       |                      |           |
| age       | -.0058414      | .0114114  | -0.51 | 0.609 | -.0282073            | .0165245  |
| male      | .5102237       | .3639793  | 1.40  | 0.161 | -.2031626            | 1.22361   |
| nonwhite  | .4333141       | .4106255  | 1.06  | 0.291 | -.371497             | 1.238125  |
| _cons     | -1.811165      | .5348606  | -3.39 | 0.001 | -2.859473            | -.7628578 |

```
. estat vce, block
```

Covariance matrix of coefficients of mlogit model

|                                                                       |            |            |            |           |  |
|-----------------------------------------------------------------------|------------|------------|------------|-----------|--|
| covariances of equation Indemnity                                     |            |            |            |           |  |
|                                                                       | age        | male       | nonwhite   | _cons     |  |
| age                                                                   | 0          |            |            |           |  |
| male                                                                  | 0          | 0          |            |           |  |
| nonwhite                                                              | 0          | 0          | 0          |           |  |
| _cons                                                                 | 0          | 0          | 0          | 0         |  |
| covariances of equation Prepaid (row) by equation Indemnity (column)  |            |            |            |           |  |
|                                                                       | age        | male       | nonwhite   | _cons     |  |
| age                                                                   | 0          |            |            |           |  |
| male                                                                  | 0          | 0          |            |           |  |
| nonwhite                                                              | 0          | 0          | 0          |           |  |
| _cons                                                                 | 0          | 0          | 0          | 0         |  |
| covariances of equation Prepaid                                       |            |            |            |           |  |
|                                                                       | age        | male       | nonwhite   | _cons     |  |
| age                                                                   | .00003711  |            |            |           |  |
| male                                                                  | -.00015303 | .0402091   |            |           |  |
| nonwhite                                                              | -.00008948 | .00470608  | .04795135  |           |  |
| _cons                                                                 | -.00159095 | -.00398961 | -.00628886 | .08000462 |  |
| covariances of equation Uninsure (row) by equation Indemnity (column) |            |            |            |           |  |
|                                                                       | age        | male       | nonwhite   | _cons     |  |
| age                                                                   | 0          |            |            |           |  |
| male                                                                  | 0          | 0          |            |           |  |
| nonwhite                                                              | 0          | 0          | 0          |           |  |
| _cons                                                                 | 0          | 0          | 0          | 0         |  |

covariances of equation Uninsure (row) by equation Prepaid (column)

|          | age        | male       | nonwhite   | _cons      |
|----------|------------|------------|------------|------------|
| age      | .00001753  | -.00007926 | -.00004564 | -.00076886 |
| male     | -.00007544 | .02188398  | .0023186   | -.00145923 |
| nonwhite | -.00004577 | .00250588  | .02813553  | -.00263872 |
| _cons    | -.00077045 | -.00130535 | -.00257593 | .03888032  |

covariances of equation Uninsure

|          | age        | male       | nonwhite   | _cons     |
|----------|------------|------------|------------|-----------|
| age      | .00013022  |            |            |           |
| male     | -.00050406 | .13248095  |            |           |
| nonwhite | -.00026145 | .01505449  | .16861327  |           |
| _cons    | -.00562159 | -.01686629 | -.02474852 | .28607591 |

The `block` option is particularly useful for multiple-equation estimators. The first block of output here corresponds to the VCE of the estimated parameters for the first equation—the square roots of the diagonal elements of this matrix are equal to the standard errors of the first equation’s parameters. Similarly, the final block corresponds to the VCE of the parameters for the second equation. The middle block shows the covariances between the estimated parameters of the first and second equations.

◀

Saved results

`estat ic` saves the following in `r()`:

Matrices

`r(S)` 1 × 6 matrix of results:

1. sample size

4. degrees of freedom

2. log likelihood of null model

5. AIC

3. log likelihood of full model

6. BIC

`estat summarize` saves the following in `r()`:

Matrices

`r(stats)`  $k \times 4$  matrix of means, standard deviations, minimums, and maximums

`estat vce` saves the following in `r()`:

Matrices

`r(V)` VCE or correlation matrix

Methods and formulas

`estat` is implemented as an ado-file.

Akaike’s (1974) information criterion is defined as

$$AIC = -2 \ln L + 2k$$

where  $\ln L$  is the maximized log-likelihood of the model and  $k$  is the number of parameters estimated. Some authors define the AIC as the expression above divided by the sample size.

Schwarz's (1978) Bayesian information criterion is another measure of fit defined as

$$\text{BIC} = -2 \ln L + k \ln N$$

where  $N$  is the sample size. See [R] **BIC note** for additional information on calculating and interpreting BIC.

Hirotsugu Akaike (1927– ) was born in Fujinomiya City, Shizuoka Prefecture, Japan. He gained BA and PhD degrees from the University of Tokyo. Akaike's career from 1952 at the Institute of Statistical Mathematics in Japan culminated in service as Director General; since 1994, he has been Professor Emeritus. His best known work in a prolific career is on what is now known as the Akaike information criterion (AIC), which was formulated to help selection of the most appropriate model from a number of candidates.

Gideon E. Schwarz (1933– ) is Professor Emeritus of Statistics at the Hebrew University, Jerusalem. He was born in Salzburg, Austria, and obtained an MSc in 1956 from the Hebrew University and a PhD in 1961 from Columbia University. He is interested in stochastic processes, sequential analysis, probability, and geometry, and is known for the Bayesian information criterion (BIC).

## References

- Akaike, H. 1973. Information theory and an extension of the maximum likelihood principle. In *Second International Symposium on Information Theory*, ed. B. N. Petrov and F. Csaki, 267–281. Budapest: Akaikeonai-Kiudo.
- . 1974. A new look at the statistical model identification. *IEEE Transactions on Automatic Control* 19: 716–723.
- Belsley, D. A., E. Kuh, and R. E. Welsch. 1980. *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. New York: Wiley.
- Findley, D. F., and E. Parzen. 1995. A conversation with Hirotsugu Akaike. *Statistical Science* 10: 104–117.
- Hardin, J. W., and J. M. Hilbe. 2007. *Generalized Linear Models and Extensions*. 2nd ed. College Station, TX: Stata Press.
- Judge, G. G., W. E. Griffiths, R. C. Hill, H. Lütkepohl, and T.-C. Lee. 1985. *The Theory and Practice of Econometrics*. 2nd ed. New York: Wiley.
- Raftery, A. 1995. Bayesian model selection in social research. In Vol. 25 of *Sociological Methodology*, ed. P. V. Marsden, 111–163. Oxford: Blackwell.
- Royston, P., and W. Sauerbrei. 2008. *Multivariable Model-building: A Pragmatic Approach to Regression Analysis Based on Fractional Polynomials for Modelling Continuous Variables*. Chichester, UK: Wiley.
- Sawa, T. 1978. Information criteria for discriminating among alternative regression models. *Econometrica* 46: 1273–1291.
- Schwarz, G. 1978. Estimating the dimension of a model. *Annals of Statistics* 6: 461–464.

## Also see

- [R] **estimates** — Save and manipulate estimation results
- [R] **summarize** — Summary statistics
- [P] **estat programming** — Controlling estat after user-written commands
- [U] **20 Estimation and postestimation commands**



Syntax

| command                                               | reference                              |
|-------------------------------------------------------|----------------------------------------|
| <i>Save and use results from disk</i>                 |                                        |
| <code>estimates save <i>filename</i></code>           | [R] <a href="#">estimates save</a>     |
| <code>estimates use <i>filename</i></code>            | [R] <a href="#">estimates save</a>     |
| <code>estimates describe using <i>filename</i></code> | [R] <a href="#">estimates describe</a> |
| <code>estimates esample: ...</code>                   | [R] <a href="#">estimates save</a>     |
| <i>Store and restore estimates in memory</i>          |                                        |
| <code>estimates store <i>name</i></code>              | [R] <a href="#">estimates store</a>    |
| <code>estimates restore <i>name</i></code>            | [R] <a href="#">estimates store</a>    |
| <code>estimates query</code>                          | [R] <a href="#">estimates store</a>    |
| <code>estimates dir</code>                            | [R] <a href="#">estimates store</a>    |
| <code>estimates drop <i>namelist</i></code>           | [R] <a href="#">estimates store</a>    |
| <code>estimates clear</code>                          | [R] <a href="#">estimates store</a>    |
| <i>Set titles and notes</i>                           |                                        |
| <code>estimates title: <i>text</i></code>             | [R] <a href="#">estimates title</a>    |
| <code>estimates title</code>                          | [R] <a href="#">estimates title</a>    |
| <code>estimates notes: <i>text</i></code>             | [R] <a href="#">estimates notes</a>    |
| <code>estimates notes</code>                          | [R] <a href="#">estimates notes</a>    |
| <code>estimates notes list ...</code>                 | [R] <a href="#">estimates notes</a>    |
| <code>estimates notes drop ...</code>                 | [R] <a href="#">estimates notes</a>    |
| <i>Report</i>                                         |                                        |
| <code>estimates describe [<i>name</i>]</code>         | [R] <a href="#">estimates describe</a> |
| <code>estimates replay [<i>namelist</i>]</code>       | [R] <a href="#">estimates replay</a>   |
| <i>Tables and statistics</i>                          |                                        |
| <code>estimates table [<i>namelist</i>]</code>        | [R] <a href="#">estimates table</a>    |
| <code>estimates stats [<i>namelist</i>]</code>        | [R] <a href="#">estimates stats</a>    |
| <code>estimates for <i>namelist</i>: ...</code>       | [R] <a href="#">estimates for</a>      |

## Description

`estimates` allows you to store and manipulate estimation results:

- You can save estimation results in a file for use in later sessions.
- You can store estimation results in memory so that you can
  - a. switch among separate estimation results and
  - b. form tables combining separate estimation results.

## Remarks

`estimates` is for use after you have fit a model, be it with `regress`, `logistic`, etc. You can use `estimates` after any estimation command, whether it be an official estimation command of Stata or a user-written one.

`estimates` has three separate but related capabilities:

1. You can save estimation results in a file on disk so that you can use them later, even in a different Stata session.
2. You can store up to 300 estimation results in memory so that they are at your fingertips.
3. You can make tables comparing any results you have stored in memory.

Remarks are presented under the following headings:

[Saving and using estimation results](#)  
[Storing and restoring estimation results](#)  
[Comparing estimation results](#)  
[Jargon](#)

## Saving and using estimation results

After you have fit a model, say, with `regress`, type

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. regress mpg weight displ foreign
(output omitted)
```

You can save the results in a file:

```
. estimates save basemodel
(file basemodel.ster saved)
```

Later, say, in a different session, you can reload those results:

```
. estimates use basemodel
```

The situation is now nearly identical to what it was immediately after you fit the model. You can replay estimation results:

```
. regress
(output omitted)
```

You can perform tests:

```
. test foreign==0
(output omitted)
```

And you can use any postestimation command or postestimation capability of Stata. The only difference is that Stata no longer knows what the estimation sample, `e(sample)` in Stata jargon, was. When you reload the estimation results, you might not even have the original data in memory. That is okay. Stata will know to refuse to calculate anything that can be calculated only on the original estimation sample.

If it is important that you use a postestimation command that can be used only on the original estimation sample, there is a way you can do that. You use the original data and then use `estimates esample` to tell Stata what the original sample was.

See [\[R\] estimates save](#) for details.

## Storing and restoring estimation results

Storing and restoring estimation results in memory is much like saving them to disk. You type

```
. estimates store base
```

to save the current estimation results under the name `base`, and you type

```
. estimates restore base
```

to get them back later. You can find out what you have stored by typing

```
. estimates dir
```

Saving estimation results to disk is more permanent than storing them in memory, so why would you want merely to store them? The answer is that, once they are stored, you can use other `estimates` commands to produce tables and reports from them.

See [\[R\] estimates store](#) for details about the `estimates store` and `restore` commands.

## Comparing estimation results

Let's say that you have done the following:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. regress mpg weight displ
(output omitted)
. estimates store base
. regress mpg weight displ foreign
(output omitted)
. estimates store alt
```

You can now get a table comparing the coefficients:

```
. estimates table base alt
```

| Variable     | base       | alt        |
|--------------|------------|------------|
| weight       | -.00656711 | -.00677449 |
| displacement | .00528078  | .00192865  |
| foreign      |            | -1.6006312 |
| _cons        | 40.084522  | 41.847949  |

`estimates table` can do much more; see [\[R\] estimates table](#). Also see [\[R\] estimates stats](#). It works similarly to `table` but produces model comparisons in terms of BIC and AIC.

## Jargon

You know that if you fit a model, say, by typing

```
. regress mpg weight displacement
```

then you can later replay the results by typing

```
. regress
```

and you can do tests and calculate other postestimation statistics by typing

```
. test displacement==0
. estat vif
. predict mpghat
```

As a result, we often refer to the *estimation results* or the *current estimation results* or the *most recent estimation results* or the *last estimation results* or the *estimation results in memory*.

With `estimates store` and `estimates restore`, you can have many estimation results in memory. One set of those, the set most recently estimated, or the set most recently restored, are the *current* or *active* estimation results, which you can replay, which you can test, or from which you can calculate postestimation statistics.

*Current* and *active* are the two words we will use interchangeably from now on.

## Also see

[R] [estimates save](#) — Save and use estimation results

[R] [estimates describe](#) — Describe estimation results

[R] [estimates store](#) — Store and restore estimation results

[R] [estimates title](#) — Set title for estimation results

[R] [estimates notes](#) — Add notes to estimation results

[R] [estimates replay](#) — Redisplay estimation results

[R] [estimates table](#) — Compare estimation results

[R] [estimates stats](#) — Model statistics

[R] [estimates for](#) — Repeat postestimation command across models

# Title

**estimates describe** — Describe estimation results

## Syntax

```
estimates describe
estimates describe name
estimates describe using filename [, number(#)]
```

## Menu

Statistics > Postestimation > Manage estimation results > Describe results

## Description

`estimates describe` describes the current (active) estimates. Reported are the command line that produced the estimates, any title that was set by `estimates title` (see [R] [estimates title](#)), and any notes that were added by `estimates notes` (see [R] [estimates notes](#)).

`estimates describe name` does the same but reports results for estimates stored by `estimates store` (see [R] [estimates store](#)).

`estimates describe using filename` does the same but reports results for estimates saved by `estimates save` (see [R] [estimates save](#)). If *filename* contains multiple sets of estimates (saved in it by `estimates save`, `append`), the number of sets of estimates is also reported. If *filename* is specified without an extension, `.ster` is assumed.

## Option

`number(#)` specifies that the *#*th set of estimation results from *filename* be described. This assumes that multiple sets of estimation results have been saved in *filename* by `estimates save`, `append`. The default is `number(1)`.

## Remarks

`estimates describe` can be used to describe the estimation results currently in memory,

```
. estimates describe
Estimation results produced by
 . regress mpg weight displ if foreign
```

or to describe results saved by `estimates save` in a `.ster` file:

```
. estimates describe using final
Estimation results "Final results" saved on 12apr2009 14:20, produced by
 . logistic myopic age sex drug1 drug2 if complete==1
Notes:
1. Used file patient.dta
2. "datasignature myopic age sex drug1 drug2 if complete==1"
 reports 148:5(58763):2252897466:3722318443
3. must be reviewed by rgg
```

## ▷ Example 1

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. regress mpg weight displ if foreign
(output omitted)

. estimates notes: file 'c(filename)'

. datasignature
74:12(71728):3831085005:1395876116

. estimates notes: datasignature report 'r(datasignature)'

. estimates save foreign
file foreign.ster saved

. regress mpg weight displ if !foreign
(output omitted)

. estimates describe using foreign
Estimation results saved on 02may2009 10:33, produced by
. regress mpg weight displ if foreign

Notes:
1. file http://www.stata-press.com/data/r11/auto.dta
2. datasignature report 74:12(71728):3831085005:1395876116
```

◀

## Saved results

`estimates describe` and `estimates describe name` save the following in `r()`:

### Macros

|                         |                       |
|-------------------------|-----------------------|
| <code>r(title)</code>   | title                 |
| <code>r(cmdline)</code> | original command line |

`estimates describe using filename` saves the above and the following in `r()`:

### Scalars

|                             |                                              |
|-----------------------------|----------------------------------------------|
| <code>r(datetime)</code>    | %tc value of date/time file saved            |
| <code>r(nestresults)</code> | number of sets of estimation results in file |

## Methods and formulas

`estimates describe` is implemented as an ado-file.

## Also see

[R] [estimates](#) — Save and manipulate estimation results

# Title

estimates for — Repeat postestimation command across models

## Syntax

`estimates for namelist [, options]: postestimation_command`

where *namelist* is a name, a list of names, `_all`, or `*`. A name may be `.`, meaning the current (active) estimates. `_all` and `*` mean the same thing.

| <i>options</i>        | description                  |
|-----------------------|------------------------------|
| <code>noheader</code> | do not display title         |
| <code>nostop</code>   | do not stop if command fails |

## Description

`estimates for` performs *postestimation\_command* on each estimation result specified.

## Options

- `noheader` suppresses the display of the header as *postestimation\_command* is executed each time.
- `nostop` specifies that execution of *postestimation\_command* is to be performed on the remaining models even if it fails on some.

## Remarks

In the example that follows, we fit a model two different ways, store the results, and then use `estimates for` to perform the same test on both of them:

### ► Example 1

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. gen fwgt = foreign*weight
. gen dwgt = !foreign*weight
. gen gpm = 1/mpg
. regress gpm fwgt dwgt displ foreign
(output omitted)
. estimates store reg
. qreg gpm fwgt dwgt displ foreign
(output omitted)
. estimates store qreg
```

```
. estimates for reg qreg: test fwgt==dwgt
```

|                   |                 |        |  |
|-------------------|-----------------|--------|--|
| Model <b>reg</b>  |                 |        |  |
| <hr/>             |                 |        |  |
| ( 1)              | fwgt - dwgt = 0 |        |  |
|                   | F( 1, 69) =     | 4.87   |  |
|                   | Prob > F =      | 0.0307 |  |
| <hr/>             |                 |        |  |
| Model <b>qreg</b> |                 |        |  |
| <hr/>             |                 |        |  |
| ( 1)              | fwgt - dwgt = 0 |        |  |
|                   | F( 1, 69) =     | 0.07   |  |
|                   | Prob > F =      | 0.7937 |  |



## Methods and formulas

`estimates for` is implemented as an ado-file.

## Also see

[\[R\] estimates](#) — Save and manipulate estimation results



# Title

**estimates notes** — Add notes to estimation results

## Syntax

```
estimates notes: text
```

```
estimates notes
```

```
estimates notes list [in noterange]
```

```
estimates notes drop in noterange
```

where *noterange* is # or #/# and where # may be a number, the letter f (meaning first), or the letter l (meaning last).

## Description

`estimates notes: text` adds a note to the current (active) estimation results.

`estimates notes` and `estimates notes list` list the current notes.

`estimates notes drop in noterange` eliminates the specified notes.

## Remarks

After adding or removing notes, if estimates have been stored, do not forget to store them again. If estimates have been saved, do not forget to save them again.

Notes are most useful when you intend to save estimation results in a file; see [R] [estimates save](#). For instance, after fitting a model, you might type

```
. estimates note: I think these are final
. estimates save lock2
```

and then, later when going through your files, you could type

```
. estimates use lock2
. estimates notes
1. I think these are final
```

Up to 9,999 notes can be attached to estimation results. If estimation results are important, we recommend that you add a note identifying the `.dta` dataset you used. The best way to do that is to type

```
. estimates notes: file 'c(filename)'
```

because `'c(filename)'` will expand to include not just the name of the file but also its full path; see [P] [creturn](#).

If estimation results took a long time to estimate—say, they were produced by `asmpbit` or `gllamm` (see [R] [asmpbit](#) and <http://www.gllamm.org>)—it is also a good idea to add a data signature. A data signature takes less time to compute than reestimation when you need proof that you really have the right dataset. The easy way to do that is to type

```
. datasignature
74:12(71728):3831085005:1395876116
. estimates notes: datasignature reports 'r(datasignature)'
```

Now when you ask to see the notes, you will see

```
. estimates notes
1. I think these are final
2. file C:\project\one\pat4.dta
3. datasignature reports 74:12(71728):3831085005:1395876116
```

See [D] [datasignature](#).

Notes need not be positive. You might set a note to be, “I need to check that age is defined correctly.”

## ► Example 1

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. regress mpg weight displ if foreign
(output omitted)
. estimates notes: file 'c(filename)'.
. datasignature
74:12(71728):3831085005:1395876116
. estimates notes: datasignature report 'r(datasignature)'.
. estimates save foreign
file foreign.ster saved
. estimates notes list in 1/2
1. file http://www.stata-press.com/data/r11/auto.dta
2. datasignature report 74:12(71728):3831085005:1395876116
. estimates notes drop in 2
(1 note dropped)
. estimates notes
1. file http://www.stata-press.com/data/r11/auto.dta
```

◀

## Methods and formulas

`estimates notes` is implemented as an ado-file.

## Also see

[R] [estimates](#) — Save and manipulate estimation results

# Title

estimates replay — Redisplay estimation results

## Syntax

`estimates replay`

`estimates replay namelist`

where *namelist* is a name, a list of names, `_all`, or `*`. A name may be `.`, meaning the current (active) estimates. `_all` and `*` mean the same thing.

## Menu

Statistics > Postestimation > Manage estimation results > Redisplay estimation output

## Description

`estimates replay` redisplays the current (active) estimation results, just as typing the name of the estimation command would do.

`estimates replay namelist` redisplays each specified estimation result. The active estimation results are left unchanged.

## Remarks

In the example that follows, we fit a model two different ways, store the results, use `estimates for` to perform the same test on both of them, and then replay the results:

### ► Example 1

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. gen fwgt = foreign*weight
. gen dwgt = !foreign*weight
. gen gpm = 1/mpg
. regress gpm fwgt dwgt displ foreign
(output omitted)
. estimates store reg
. qreg gpm fwgt dwgt displ foreign
(output omitted)
. estimates store qreg
. estimates for reg qreg: test fwgt==dwgt
```

---

Model **reg**

---

```
(1) fwgt - dwgt = 0
 F(1, 69) = 4.87
 Prob > F = 0.0307
```

```
Model qreg

(1) fwgt - dwgt = 0
 F(1, 69) = 0.07
 Prob > F = 0.7937

. estimates replay
```

```
Model qreg

Median regression Number of obs = 74
Raw sum of deviations .7555689 (about .05)
Min sum of deviations .3201479 Pseudo R2 = 0.5763
```

| gpm          | Coef.    | Std. Err. | t    | P> t  | [95% Conf. Interval] |          |
|--------------|----------|-----------|------|-------|----------------------|----------|
| fwgt         | .0000155 | 2.87e-06  | 5.40 | 0.000 | 9.76e-06             | .0000212 |
| dwgt         | .0000147 | 1.88e-06  | 7.81 | 0.000 | .0000109             | .0000184 |
| displacement | .0000179 | .0000147  | 1.22 | 0.226 | -.0000113            | .0000471 |
| foreign      | .0065352 | .0078098  | 0.84 | 0.406 | -.009045             | .0221153 |
| _cons        | .0003134 | .0042851  | 0.07 | 0.942 | -.0082351            | .0088618 |

```
. estimates replay reg
```

| Source   | SS         | df | MS         | Number of obs = 74 |        |  |
|----------|------------|----|------------|--------------------|--------|--|
| Model    | .009342436 | 4  | .002335609 | F( 4, 69) =        | 61.62  |  |
| Residual | .002615192 | 69 | .000037901 | Prob > F =         | 0.0000 |  |
| Total    | .011957628 | 73 | .000163803 | R-squared =        | 0.7813 |  |
|          |            |    |            | Adj R-squared =    | 0.7686 |  |
|          |            |    |            | Root MSE =         | .00616 |  |

| gpm          | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|--------------|-----------|-----------|-------|-------|----------------------|----------|
| fwgt         | .00002    | 3.27e-06  | 6.12  | 0.000 | .0000135             | .0000265 |
| dwgt         | .0000123  | 2.30e-06  | 5.36  | 0.000 | 7.75e-06             | .0000169 |
| displacement | .0000296  | .0000187  | 1.58  | 0.119 | -7.81e-06            | .000067  |
| foreign      | -.0117756 | .0086088  | -1.37 | 0.176 | -.0289497            | .0053986 |
| _cons        | .0053352  | .0046748  | 1.14  | 0.258 | -.0039909            | .0146612 |



## Methods and formulas

estimates replay is implemented as an ado-file.

## Also see

[R] estimates — Save and manipulate estimation results

# Title

estimates save — Save and use estimation results

## Syntax

```
estimates save filename [, append replace]
```

```
estimates use filename [, number(#)]
```

```
estimates esample: [varlist] [if] [in] [weight]
[, replace stringvars(varlist) zeroweight]
```

```
estimates esample
```

## Menu

### estimates save

Statistics > Postestimation > Manage estimation results > Save to disk

### estimates use

Statistics > Postestimation > Manage estimation results > Load from disk

## Description

`estimates save filename` saves the current (active) estimation results in *filename*.

`estimates use filename` loads the results saved in *filename* into the current (active) estimation results.

In both cases, if *filename* is specified without an extension, `.ster` is assumed.

`estimates esample:` (note the colon) resets `e(sample)`. After `estimates use filename`, `e(sample)` is set to contain 0, meaning that none of the observations currently in memory was used in obtaining the estimates.

`estimates esample` (without a colon) displays how `e(sample)` is currently set.

## Options

`append`, used with `estimates save`, specifies that results be appended to an existing file. If the file does not already exist, a new file is created.

`replace`, used with `estimates save`, specifies that *filename* can be replaced if it already exists.

`number( # )`, used with `estimates use`, specifies that the #th set of estimation results from *filename* be loaded. This assumes that multiple sets of estimation results have been saved in *filename* by `estimates save, append`. The default is `number(1)`.

`replace`, used with `estimates esample:`, specifies that `e(sample)` can be replaced even if it is already set.

`stringvars(varlist)`, used with `estimates esample:`, specifies string variables. Observations containing variables that contain "" will be omitted from `e(sample)`.

`zeroweight`, used with `estimates esample:`, specifies that observations with zero weights are to be included in `e(sample)`.

Remarks

See [R] [estimates](#) for an overview of the `estimates` commands.

For a description of `estimates save` and `estimates use`, see *Saving and using estimation results* in [R] [estimates](#).

The rest of this entry concerns `e(sample)`.

Remarks are presented under the following headings:

- [Setting e\(sample\)](#)
- [Resetting e\(sample\)](#)
- [Determining who set e\(sample\)](#)

Setting e(sample)

After `estimates use filename`, the situation is nearly identical to what it was immediately after you fit the model. The one difference is that `e(sample)` is set to 0.

`e(sample)` is Stata's function to mark which observations among those currently in memory were used in producing the estimates. For instance, you might type

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. regress mpg weight displ if foreign
(output omitted)
. summarize mpg if e(sample)
(output omitted)
```

and `summarize` would report the summary statistics for the observations `regress` in fact used, which would exclude not only observations for which `foreign` = 0 but also any observations for which `mpg`, `weight`, or `displ` was missing.

If you saved the above estimation results and then reloaded them, however, `summarize mpg if e(sample)` would produce

```
. summarize mpg if e(sample)
```

| Variable | Obs | Mean | Std. Dev. | Min | Max |
|----------|-----|------|-----------|-----|-----|
| mpg      | 0   |      |           |     |     |

Stata thinks that none of these observations was used in producing the estimates currently loaded.

What else could Stata think? When you `estimates use filename`, you do not have to have the original data in memory. Even if you do have data in memory that look like the original data, it might not be. Setting `e(sample)` to 0 is the safe thing to do. There are some postestimation statistics, for instance, that are appropriate only when calculated on the estimation sample. Setting `e(sample)` to 0 ensures that, should you ask for one of them, you will get back a null result.

We recommend that you leave `e(sample)` set to 0. But what if you really need to calculate that postestimation statistic? Well, you can get it, but you are going to take responsibility for setting `e(sample)` correctly. Here we just happen to know that all the foreign observations were used, so we can type

```
. estimates esample: if foreign
```

If all the observations had been used, we could simply type

```
. estimates esample:
```

The safe thing to do, however, is to look at the estimation command—`estimates describe` will show it to you—and then type

```
. estimates esample: mpg weight displ if foreign
```

## Resetting `e(sample)`

`estimates esample:` will allow you to not only set but also reset `e(sample)`. If `e(sample)` has already been set (say that you just fit the model) and you try to set it, you will see

```
. estimates esample: mpg weight displ if foreign
no; e(sample) already set
r(322);
```

Here you can specify the `replace` option:

```
. estimates esample: mpg weight displ if foreign, replace
```

We do not recommend resetting `e(sample)`, but the situation can arise where you need to. Imagine that you `estimates use filename`, you set `e(sample)`, and then you realize that you set it wrong. Here you would want to reset it.

## Determining who set `e(sample)`

`estimates esample` without a colon will report whether and how `e(sample)` was set. You might see

```
. estimates esample
e(sample) set by estimation command
```

or

```
. estimates esample
e(sample) set by user
```

or

```
. estimates esample
e(sample) not set (0 assumed)
```

## Saved results

`estimates esample` without the colon saves macro `r(who)`, which will contain `cmd`, `user`, or `zero'd`.

## Methods and formulas

`estimates save`, `estimates use`, `estimates esample:`, and `estimates esample` are implemented as ado-files.

## Also see

[\[R\] estimates](#) — Save and manipulate estimation results



Syntax

```
estimates stats [namelist] [, n(#)]
```

where *namelist* is a name, a list of names, `_all`, or `*`. A name may be `.`, meaning the current (active) estimates. `_all` and `*` mean the same thing.

Menu

Statistics > Postestimation > Manage estimation results > Table of fit statistics

Description

`estimates stats` reports model-selection statistics, including the Akaike information criterion (AIC) and the Bayesian information criterion (BIC). These measures are appropriate for maximum likelihood models.

If `estimates stats` is used for a non-likelihood-based model, such as `qreg`, missing values are reported.

Option

`n( # )` specifies the *N* to be used in calculating BIC; see [\[R\] BIC note](#).

Remarks

If you type `estimates stats` without arguments, a table for the most recent estimation results will be shown:

```
. logistic foreign mpg weight displ
(output omitted)
. estimates stats
```

| Model | Obs | ll(null)  | ll(model) | df | AIC      | BIC      |
|-------|-----|-----------|-----------|----|----------|----------|
| .     | 74  | -45.03321 | -20.59083 | 4  | 49.18167 | 58.39793 |

Note: N=Obs used in calculating BIC; see [\[R\] BIC note](#)

Regarding the note at the bottom of the table, *N* is an ingredient in the calculation of BIC; see [\[R\] BIC note](#). The note changes if you specify the `n()` option, which tells `estimates stats` what *N* to use. *N* = Obs is the default.

Regarding the table itself, `ll(null)` is the log likelihood for the constant-only model, `ll(model)` is the log likelihood for the model, `df` is the number of degrees of freedom, and AIC and BIC are the Akaike and Bayesian information criteria.

Models with smaller values of an information criterion are considered preferable.

estimates stats can compare estimation results:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. logistic foreign mpg weight displ
(output omitted)

. estimates store full

. logistic foreign mpg weight
(output omitted)

. estimates store sub

. estimates stats full sub
```

| Model | Obs | ll(null)  | ll(model) | df | AIC      | BIC      |
|-------|-----|-----------|-----------|----|----------|----------|
| full  | 74  | -45.03321 | -20.59083 | 4  | 49.18167 | 58.39793 |
| sub   | 74  | -45.03321 | -27.17516 | 3  | 60.35031 | 67.26251 |

Note: N=Obs used in calculating BIC; see [\[R\] BIC note](#)

## Saved results

estimates stats saves the following in `r()`:

Matrices  
`r(S)` matrix with 6 columns (N, ll0, ll, df, AIC, and BIC) and rows corresponding to models in table

## Methods and formulas

estimates stats is implemented as an ado-file.  
See [\[R\] BIC note](#).

## Also see

[\[R\] estimates](#) — Save and manipulate estimation results

# Title

estimates store — Store and restore estimation results

## Syntax

```
estimates store name [, nocopy]
```

```
estimates restore name
```

```
estimates query
```

```
estimates dir [namelist]
```

```
estimates drop namelist
```

```
estimates clear
```

where *namelist* is a name, a list of names, `_all`, or `*`. `_all` and `*` mean the same thing.

## Menu

### estimates store

Statistics > Postestimation > Manage estimation results > Store in memory

### estimates restore

Statistics > Postestimation > Manage estimation results > Restore from memory

### estimates dir

Statistics > Postestimation > Manage estimation results > List results stored in memory

### estimates drop

Statistics > Postestimation > Manage estimation results > Drop from memory

## Description

`estimates store name` saves the current (active) estimation results under the name *name*.

`estimates restore name` loads the results saved under *name* into the current (active) estimation results.

`estimates query` tells you whether the current (active) estimates have been stored and, if so, the name.

`estimates dir` displays a list of the stored estimates.

`estimates drop namelist` drops the specified stored estimation results.

`estimates clear` drops all stored estimation results.

`estimates clear`, `estimates drop _all`, and `estimates drop *` do the same thing. `estimates drop` and `estimates clear` do not eliminate the current (active) estimation results.

## Option

`nocopy`, used with `estimates store`, specifies that the current (active) estimation results are to be moved into *name* rather than copied. Typing

```
. estimates store hold, nocopy
```

is the same as typing

```
. estimates store hold
. ereturn clear
```

except that the former is faster. The `nocopy` option is sometimes used by programmers.

## Remarks

`estimates store` stores estimation results in memory so that you can access them later.

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. regress mpg weight displ
(output omitted)
. estimates store myreg
. ... you do other things, including fitting other models ...
. estimates restore myreg
. regress
(same output shown again)
```

After `estimates restore myreg`, things are once again just as they were, estimationwise, just after you typed `regress mpg weight displ`.

`estimates store` stores results in memory. When you exit Stata, those stored results vanish. If you wish to make a permanent copy of your estimation results, see [\[R\] estimates save](#).

The purpose of making copies in memory is 1) so that you can quickly switch between them and 2) so that you can make tables comparing estimation results. Concerning the latter, see [\[R\] estimates table](#) and [\[R\] estimates stats](#).

## Saved results

`estimates dir` saves the following in `r()`:

```
Macros
 r(names) names of stored results
```

## Methods and formulas

`estimates store`, `estimates restore`, `estimates query`, `estimates dir`, `estimates drop`, and `estimates clear` are implemented as ado-files.

## References

- Jann, B. 2005. [Making regression tables from stored estimates](#). *Stata Journal* 5: 288–308.
- . 2007. [Making regression tables simplified](#). *Stata Journal* 7: 227–244.

## Also see

[\[R\] estimates](#) — Save and manipulate estimation results

Syntax

```
estimates table [namelist] [, options]
```

where *namelist* is a name, a list of names, `_all`, or `*`. A name may be `.`, meaning the current (active) estimates. `_all` and `*` mean the same thing.

| <i>options</i>                                 | description                                                               |
|------------------------------------------------|---------------------------------------------------------------------------|
| Main                                           |                                                                           |
| <code>stats(<i>scalarlist</i>)</code>          | report <i>scalarlist</i> in table                                         |
| <code>star [ (<i>#1 #2 #3</i>) ]</code>        | use stars to denote significance levels                                   |
| Options                                        |                                                                           |
| <code>keep(<i>coeflist</i>)</code>             | report coefficients in order specified                                    |
| <code>drop(<i>coeflist</i>)</code>             | omit specified coefficients from table                                    |
| <code>equations(<i>matchlist</i>)</code>       | match equations of models as specified                                    |
| Numerical formats                              |                                                                           |
| <code>b [ (<i>%fmt</i>) ]</code>               | how to format coefficients, which are always reported                     |
| <code>se [ (<i>%fmt</i>) ]</code>              | report standard errors and use optional format                            |
| <code>t [ (<i>%fmt</i>) ]</code>               | report <i>t</i> or <i>z</i> and use optional format                       |
| <code>p [ (<i>%fmt</i>) ]</code>               | report <i>p</i> -values and use optional format                           |
| <code>stfmt(<i>%fmt</i>)</code>                | how to format scalar statistics                                           |
| General format                                 |                                                                           |
| <code>varwidth(<i>#</i>)</code>                | use <i>#</i> characters to display variable names and statistics          |
| <code>modelwidth(<i>#</i>)</code>              | use <i>#</i> characters to display model names                            |
| <code>eform</code>                             | display coefficients in exponentiated form                                |
| <code>label</code>                             | display variable labels rather than variable names                        |
| <code>newpanel</code>                          | display statistics in separate table from coefficients                    |
| <code>style(online)</code>                     | put vertical line after variable names; the default                       |
| <code>style(columns)</code>                    | put vertical line separating every column                                 |
| <code>style(noline)</code>                     | suppress all vertical lines                                               |
| <code>coded</code>                             | display compact table                                                     |
| Reporting                                      |                                                                           |
| <code>display_options</code>                   | control spacing and display of omitted variables and base and empty cells |
| <sup>†</sup> <code>title(<i>string</i>)</code> | title for table                                                           |

<sup>†</sup>`title()` does not appear in the dialog box.

where

- A *scalarlist* is a list of any or all of the names of scalars stored in `e()`, plus `aic`, `bic`, and `rank`.
- `#1 #2 #3` are three numbers such as `.05 .01 .001`.
- A *coeflist* is a list of coefficient names, each name of which may be simple (e.g., `price`), an equation name followed by a colon (e.g., `mean:`), or a full name (e.g., `mean:price`). Names are separated by blanks.
- A *matchlist* specifies how equations from different estimation results are to be matched. If you need to specify a *matchlist*, the solution is usually 1, as in `equations(1)`. The full syntax is

$$\begin{aligned} \text{matchlist} &:= \text{term} \left[ , \text{term} \dots \right] \\ \text{term} &:= \left[ \text{eqname} = \right] \# : \# \dots : \# \\ &\quad \left[ \text{eqname} = \right] \# \end{aligned}$$

See `equations()` under *Options* below.

- `%fmt` is any valid Stata numerical display format.

## Menu

Statistics > Postestimation > Manage estimation results > Table of fit statistics

## Description

`estimates table` displays a table of coefficients and statistics for one or more sets of estimation results.

## Options

Main

`stats(scalarlist)` specifies one or more scalar statistics to be displayed in the table. *scalarlist* may contain

|                   |                                                           |
|-------------------|-----------------------------------------------------------|
| <code>aic</code>  | Akaike's information criterion                            |
| <code>bic</code>  | Schwarz's Bayesian information criterion                  |
| <code>rank</code> | rank of <code>e(V)</code> (# of free parameters in model) |

along with the names of any scalars stored in `e()`. The specified statistics do not have to be available for all estimation results being displayed.

For example, `stats(N ll chi2 aic)` specifies that `e(N)`, `e(ll)`, `e(chi2)`, and `AIC` be included. In Stata, `e(N)` records the number of observations; `e(ll)`, the log likelihood; and `e(chi2)`, the chi-squared test that all coefficients in the first equation of the model are equal to zero.

`star` and `star(#1 #2 #3)` specify that stars (asterisks) are to be used to mark significance. The second syntax specifies the significance levels for one, two, and three stars. If you specify simply `star`, that is equivalent to specifying `star(.05 .01 .001)`, which means one star (\*) if  $p < 0.05$ , two stars (\*\*) if  $p < 0.01$ , and three stars (\*\*\*) if  $p < 0.001$ .

The `star` and `star()` options may not be combined with `se`, `t`, or `p` option.

### Options

`keep(coeflist)` and `drop(coeflist)` are alternatives; they specify coefficients to be included or omitted from the table. The default is to display all coefficients.

If `keep()` is specified, it specifies not only the coefficients to be included but also the order in which they appear.

A *coeflist* is a list of coefficient names, each name of which may be simple (e.g., `price`), an equation name followed by a colon (e.g., `mean:`), or a full name (e.g., `mean:price`). Names are separated from each other by blanks.

When full names are not specified, all coefficients that match the partial specification are included. For instance, `drop(_cons)` would omit `_cons` for all equations.

`equations(matchlist)` specifies how the equations of the models in *namelist* are to be matched. The default is to match equations by name. Matching by name usually works well when all results were fit by the same estimation command. When you are comparing results from different estimation commands, however, specifying `equations()` may be necessary.

The most common usage is `equations(1)`, which indicates that all first equations are to be matched into one equation named `#1`.

*matchlist* has the syntax

```
term [, term ...]
```

where *term* is

```
[eqname =] #:#...# (syntax 1)
```

```
[eqname =] # (syntax 2)
```

In syntax 1, each `#` is a number or a period (`.`). If a number, it specifies the position of the equation in the corresponding model; `1:3:1` would indicate that equation 1 in the first model matches equation 3 in the second, which matches equation 1 in the third. A period indicates that there is no corresponding equation in the model; `1::1` indicates that equation 1 in the first matches equation 1 in the third.

In syntax 2, you specify just one number, say, 1 or 2, and that is shorthand for `1:1...:1` or `2:2...:2`, meaning that equation 1 matches across all models specified or that equation 2 matches across all models specified.

Now that you can specify a *term*, you can put that together into a *matchlist* by separating one term from the other by commas. In what follows, we will assume that three names were specified,

```
. estimates table alpha beta gamma, ...
```

`equations(1)` is equivalent to `equations(1:1:1)`; we would be saying that the first equations match across the board.

`equations(1::1)` would specify that equation 1 matches in models `alpha` and `gamma` but that there is nothing corresponding in model `beta`.



`equations(1,2)` is equivalent to `equations(1:1:1, 2:2:2)`. We would be saying that the first equations match across the board and so do the second equations.

`equations(1, 2::2)` would specify that the first equations match across the board, that the second equations match for models `alpha` and `gamma`, and that there is nothing equivalent to equation 2 in model `beta`.

If `equations()` is specified, equations not matched by position are matched by name.

#### Numerical formats

`b(%fmt)` specifies how the coefficients are to be displayed. You might specify `b(%9.2f)` to make decimal points line up. There is also a `b` option, which specifies that coefficients are to be displayed, but that is just included for consistency with the `se`, `t`, and `p` options. Coefficients are always displayed.

`se`, `t`, and `p` specify that standard errors,  $t$  or  $z$  statistics, and significance levels are to be displayed. The default is not to display them. `se(%fmt)`, `t(%fmt)`, and `p(%fmt)` specify that each is to be displayed and specifies the display format to be used to format them.

`stfmt(%fmt)` specifies the format for displaying the scalar statistics included by the `stats()` option.

#### General format

`varwidth(#)` specifies the number of character positions used to display the names of the variables and statistics. The default is 12.

`modelwidth(#)` specifies the number of character positions used to display the names of the models. The default is 12.

`eform` displays coefficients in exponentiated form. For each coefficient,  $\exp(\beta)$  rather than  $\beta$  is displayed, and standard errors are transformed appropriately. Display of the intercept, if any, is suppressed.

`label` specifies that variable labels be displayed instead of variable names.

`newpanel` specifies that the statistics be displayed in a table separated by a blank line from the table with coefficients rather than in the style of another equation in the table of coefficients.

`style(stylespec)` specifies the style of the coefficient table.

`style(online)` specifies that a vertical line be displayed after the variables but not between the models. This is the default.

`style(columns)` specifies that vertical lines be displayed after each column.

`style(noline)` specifies that no vertical lines be displayed.

`coded` specifies that a compact table be displayed. This format is especially useful for comparing variables that are included in a large collection of models.

#### Reporting

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [\[R\] estimation options](#).

The following option is available with `estimates table` but is not shown in the dialog box:

`title(string)` specifies the title to appear above the table.

## Remarks

If you type `estimates table` without arguments, a table of the most recent estimation results will be shown:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. regress mpg weight displ
 (output omitted)
. estimates table
```

| Variable     | active     |
|--------------|------------|
| weight       | -.00656711 |
| displacement | .00528078  |
| _cons        | 40.084522  |

The real use of `estimates table`, however, is for comparing estimation results, and that requires using it after `estimates store`:

```
. regress mpg weight displ
 (output omitted)
. estimates store base
. regress mpg weight displ foreign
 (output omitted)
. estimates store alt
. qreg mpg weight displ foreign
 (output omitted)
. estimates store qreg
. estimates table base alt qreg, stats(r2)
```

| Variable     | base       | alt        | qreg       |
|--------------|------------|------------|------------|
| weight       | -.00656711 | -.00677449 | -.00595056 |
| displacement | .00528078  | .00192865  | .00018552  |
| foreign      |            | -1.6006312 | -2.1326004 |
| _cons        | 40.084522  | 41.847949  | 39.213348  |
| r2           | .6529307   | .66287957  |            |

## Saved results

`estimates table` saves the following in `r()`:

- Macros
- r(names)

names of results used
- Matrices
- r(coef)

matrix  $M$ :  $n \times 2*m$   
 $M[i, 2j-1]$  =  $i$ th parameter estimate for model  $j$ ;  
 $M[i, 2j]$  = variance of  $M[i, 2j-1]$ ;  $i=1,...,n$ ;  $j=1,...,m$
- r(stats)

matrix  $S$ :  $k \times m$  (if option `stats()` specified)  
 $S[i, j]$  =  $i$ th statistic for model  $j$ ;  $i=1,...,k$ ;  $j=1,...,m$

## Methods and formulas

`estimates table` is implemented as an ado-file.

## Also see

[R] [estimates](#) — Save and manipulate estimation results

# Title

**estimates title** — Set title for estimation results

## Syntax

```
estimates title: [text]
```

```
estimates title
```

## Menu

Statistics > Postestimation > Manage estimation results > Title/retitle results

## Description

`estimates title:` (note the colon) sets or clears the title for the current estimation results. The title is used by `estimates table` and `estimates stats` (see [\[R\] estimates table](#) and [\[R\] estimates stats](#)).

`estimates title` without the colon displays the current title.

## Remarks

After setting the title, if estimates have been stored, do not forget to store them again:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. regress mpg gear turn
 (output omitted)
. estimates store reg
```

Now let's add a title:

```
. estimates title: "My regression"
. estimates store reg
```

## Methods and formulas

`estimates title:` and `estimates title` are implemented as ado-files.

## Also see

[\[R\] estimates](#) — Save and manipulate estimation results

## Description

This entry describes the options common to many estimation commands. Not all the options documented below work with all estimation commands. See the documentation for the particular estimation command; if an option is listed there, it is applicable.

## Options

### Model

`noconstant` suppresses the constant term (intercept) in the model.

`offset(varname)` specifies that *varname* be included in the model with the coefficient constrained to be 1.

`exposure(varname)` specifies a variable that reflects the amount of exposure over which the *depvar* events were observed for each observation; `ln(varname)` with coefficient constrained to be 1 is entered into the log-link function.

`constraints(numlist | matname)` specifies the linear constraints to be applied during estimation. The default is to perform unconstrained estimation. See [R] [reg3](#) for the use of constraints in multiple-equation contexts.

`constraints(numlist)` specifies the constraints by number after they have been defined by using the `constraint` command; see [R] [constraint](#). Some commands (for example, `slogit`) allow only `constraints(numlist)`.

`constraints(matname)` specifies a matrix containing the constraints; see [P] [makecns](#).

`constraints(clist)` is used by some estimation commands, such as `mlogit`, where *clist* has the form `#[-#] [ , #[-#] ... ]`.

`collinear` specifies that the estimation command not omit collinear variables. Usually, there is no reason to leave collinear variables in place, and, in fact, doing so usually causes the estimation to fail because of the matrix singularity caused by the collinearity. However, with certain models, the variables may be collinear, yet the model is fully identified because of constraints or other features of the model. In such cases, using the `collinear` option allows the estimation to take place, leaving the equations with collinear variables intact. This option is seldom used.

`force` specifies that estimation be forced even though the time variable is not equally spaced. This is relevant only for correlation structures that require knowledge of the time variable. These correlation structures require that observations be equally spaced so that calculations based on lags correspond to a constant time change. If you specify a time variable indicating that observations are not equally spaced, the (time dependent) model will not be fit. If you also specify `force`, the model will be fit, and it will be assumed that the lags based on the data ordered by the time variable are appropriate.

### Correlation

`corr(correlation)` specifies the within-group correlation structure; the default corresponds to the equal-correlation model, `corr(exchangeable)`.

When you specify a correlation structure that requires a lag, you indicate the lag after the structure's name with or without a blank; e.g., `corr(ar 1)` or `corr(ar1)`.

If you specify the fixed correlation structure, you specify the name of the matrix containing the assumed correlations following the word `fixed`, e.g., `corr(fixed myr)`.

#### Reporting

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see [U] 20.7 Specifying the width of confidence intervals.

`noskip` specifies that a full maximum-likelihood model with only a constant for the regression equation be fit. This model is not displayed but is used as the base model to compute a likelihood-ratio test for the model test statistic displayed in the estimation header. By default, the overall model test statistic is an asymptotically equivalent Wald test of all the parameters in the regression equation being zero (except the constant). For many models, this option can substantially increase estimation time.

`nocnsreport` specifies that no constraints be reported. The default is to display user-specified constraints above the coefficient table.

`noomitted` specifies that variables that were omitted because of collinearity not be displayed. The default is to include in the table any variables omitted because of collinearity and to label them as “(omitted)”.

`vsquish` specifies that the blank space separating factor-variable terms or time-series-operated variables from other variables in the model be suppressed.

`noemptycells` specifies that empty cells for interactions of factor variables not be displayed. The default is to include in the table interaction cells that do not occur in the estimation sample and to label them as “(empty)”.

`baselevels` and `allbaselevels` control whether the base levels of factor variables and interactions are displayed. The default is to exclude from the table all base categories.

`baselevels` specifies that base levels be reported for factor variables and for interactions whose bases cannot be inferred from their component factor variables.

`allbaselevels` specifies that all base levels of factor variables and interactions be reported.

#### Integration

`intmethod(intmethod)` specifies the integration method to be used for the random-effects model. It accepts one of three arguments: `mvaghermite`, the default, performs mean and variance adaptive Gauss–Hermite quadrature first on every and then on alternate iterations; `aghermite` performs mode and curvature adaptive Gauss–Hermite quadrature on the first iteration only; `ghermite` performs nonadaptive Gauss–Hermite quadrature.

`intpoints(#)` specifies the number of integration points to use for integration by quadrature. The default is `intpoints(12)`; the maximum is `intpoints(195)`. Increasing this value slightly improves the accuracy but also increases computation time. Computation time is roughly proportional to its value.

The following option is not shown in the dialog box:

`coeflegend` specifies that the legend of the coefficients and how to specify them in an expression be displayed rather than the coefficient table.

---

## Also see

[U] [20 Estimation and postestimation commands](#)

# Title

**exit** — Exit Stata

## Syntax

```
exit [, clear]
```

## Description

Typing **exit** causes Stata to stop processing and return control to the operating system. If the dataset in memory has changed since the last **save** command, you must specify the **clear** option before Stata will let you exit.

**exit** may also be used for exiting do-files or programs; see [\[P\] exit](#).

Stata for Windows users may also exit Stata by clicking on the **Close** button or by pressing *Alt+F4*.

Stata for Mac users may also exit Stata by pressing *Command+Q*.

Stata(GUI) users may also exit Stata by clicking on the **Close** button.

## Option

**clear** permits you to **exit**, even if the current dataset has not been saved.

## Remarks

Type **exit** to leave Stata and return to the operating system. If the dataset in memory has changed since the last time it was saved, however, Stata will refuse. At that point, you can either **save** the dataset and then type **exit**, or type **exit, clear**:

```
. exit
no; data in memory would be lost
r(4);
. exit, clear
```

## Also see

[\[P\] exit](#) — Exit from a program or do-file



Syntax

exlogistic *depvar indepvars* [*if*] [*in*] [*weight*] [, *options*]

| <i>options</i>                                                                                  | description                                                                                                      |
|-------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|
| Model                                                                                           |                                                                                                                  |
| <u>condvars</u> ( <i>varlist</i> )                                                              | condition on variables in <i>varlist</i>                                                                         |
| <u>group</u> ( <i>varname</i> )                                                                 | groups/strata are stratified by unique values of <i>varname</i>                                                  |
| <u>binomial</u> ( <i>varname</i>   #)                                                           | data are in binomial form and the number of trials is contained in <i>varname</i> or in #                        |
| <u>estconstant</u>                                                                              | estimate constant term; do not condition on the number of successes                                              |
| <u>noconstant</u>                                                                               | suppress constant term                                                                                           |
| Terms                                                                                           |                                                                                                                  |
| <u>terms</u> ( <i>termsdef</i> )                                                                | terms definition                                                                                                 |
| Options                                                                                         |                                                                                                                  |
| <u>memory</u> (# [ <i>b</i>   <i>k</i>   <i>m</i>   <i>g</i> ])                                 | set limit on memory usage; default is <code>memory(10m)</code>                                                   |
| <u>saving</u> ( <i>filename</i> )                                                               | save the joint conditional distribution to <i>filename</i>                                                       |
| Reporting                                                                                       |                                                                                                                  |
| <u>level</u> (#)                                                                                | set confidence level; default is <code>level(95)</code>                                                          |
| <u>coef</u>                                                                                     | report estimated coefficients                                                                                    |
| <u>test</u> ( <i>testopt</i> )                                                                  | report significance of observed sufficient statistic, conditional scores test, or conditional probabilities test |
| <u>mue</u> ( <i>varlist</i> )                                                                   | compute the median unbiased estimates for <i>varlist</i>                                                         |
| <u>midp</u>                                                                                     | use the mid- <i>p</i> -value rule                                                                                |
| <u>nolog</u>                                                                                    | do not display the enumeration log                                                                               |
| by, statsby, and xi are allowed; see [U] 11.1.10 Prefix commands.                               |                                                                                                                  |
| fweights are allowed; see [U] 11.1.6 weight.                                                    |                                                                                                                  |
| See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands. |                                                                                                                  |

Menu

Statistics > Exact statistics > Exact logistic regression

Description

exlogistic fits an exact logistic regression model of *depvar* on *indepvars*.

exlogistic is an alternative to logistic, the standard maximum-likelihood-based logistic regression estimator; see [R] logistic. exlogistic produces more-accurate inference in small samples because it does not depend on asymptotic results and exlogistic can better deal with one-way causation, such as the case where all females are observed to have a positive outcome.

**exlogistic** with the **group(varname)** option is an alternative to **clogit**, the conditional logistic regression estimator; see [R] **clogit**. Like **clogit**, **exlogistic** conditions on the number of positive outcomes within stratum.

**depvar** can be specified in two ways. It can be zero/nonzero, with zero indicating failure and nonzero representing positive outcomes (successes), or if you specify the **binomial(varname| #)** option, **depvar** may contain the number of positive outcomes within each trial.

**exlogistic** is computationally intensive. Unlike most estimators, rather than calculating coefficients for all independent variables at once, results for each independent variable are calculated separately with the other independent variables temporarily conditioned out. You can save considerable computer time by skipping the parameter calculations for variables that are not of direct interest. Specify such variables in the **condvars()** option rather than among the *indepvars*; see **condvars()** below.

Unlike Stata's other estimation commands, you may not use **test**, **lincom**, or other postestimation commands after **exlogistic**. Given the method used to calculate estimates, hypothesis tests must be performed during estimation by using **exlogistic's terms()** option; see **terms()** below.

## Options

### Model

**condvars(varlist)** specifies variables whose parameter estimates are not of interest to you. You can save substantial computer time and memory moving such variables from *indepvars* to **condvars()**. Understand that you will get the same results for **x1** and **x3** whether you type

```
. exlogistic y x1 x2 x3 x4
```

or

```
. exlogistic y x1 x3, condvars(x2 x4)
```

**group(varname)** specifies the variable defining the strata, if any. A constant term is assumed for each stratum identified in *varname*, and the sufficient statistics for *indepvars* are conditioned on the observed number of successes within each group. This makes the model estimated equivalent to that estimated by **clogit**, Stata's conditional logistic regression command (see [R] **clogit**). **group()** may not be specified with **noconstant** or **estconstant**.

**binomial(varname| #)** indicates that the data are in binomial form and *depvar* contains the number of successes. *varname* contains the number of trials for each observation. If all observations have the same number of trials, you can instead specify the number as an integer. The number of trials must be a positive integer at least as great as the number of successes. If **binomial()** is not specified, the data are assumed to be Bernoulli, meaning that *depvar* equaling zero or nonzero records one failure or success.

**estconstant** estimates the constant term. By default, the models are assumed to have an intercept (constant), but the value of the intercept is not calculated. That is, the conditional distribution of the sufficient statistics for the *indepvars* is computed given the number of successes in *depvar*, thus conditioning out the constant term of the model. Use **estconstant** if you want the estimate of the intercept reported. **estconstant** may not be specified with **group()**.

**noconstant**; see [R] **estimation options**. **noconstant** may not be specified with **group()**.

## Terms

`terms(termname = variable ... variable [ , termname = variable ... variable ... ])` defines additional terms of the model on which you want `exlogistic` to perform joint-significance hypothesis tests. By default, `exlogistic` reports tests individually on each variable in `indepvars`. For instance, if variables `x1` and `x3` are in `indepvars`, and you want to jointly test their significance, specify `terms(t1=x1 x3)`. To also test the joint significance of `x2` and `x4`, specify `terms(t1=x1 x3, t2=x2 x4)`. Each variable can be assigned to only one term.

Joint tests are computed only for the conditional scores tests and the conditional probabilities tests. See the `test()` option below.

## Options

`memory(#[b|k|m|g])` sets a limit on the amount of memory `exlogistic` can use when computing the conditional distribution of the parameter sufficient statistics. The default is `memory(10m)`, where `m` stands for megabyte, or 1,048,576 bytes. The following are also available: `b` stands for byte; `k` stands for kilobyte, which is equal to 1,024 bytes; and `g` stands for gigabyte, which is equal to 1,024 megabytes. The minimum setting allowed is `1m` and the maximum is `2048m` or `2g`, but do not attempt to use more memory than is available on your computer.

`saving(filename [ , replace ])` saves the joint conditional distribution to `filename`. This distribution is conditioned on those variables specified in `condvars()`. Use `replace` to replace an existing file with `filename`. A Stata data file is created containing all the feasible values of the parameter sufficient statistics. The variable names are the same as those in `indepvars`, in addition to a variable named `_f_` containing the feasible value frequencies (sometimes referred to as the condition numbers).

## Reporting

`level(#)`; see [\[R\] estimation options](#). The `level(#)` option will not work on replay because confidence intervals are based on estimator-specific enumerations. To change the confidence level, you must refit the model.

`coef` reports the estimated coefficients rather than odds ratios (exponentiated coefficients). `coef` may be specified when the model is fit or upon replay. `coef` affects only how results are displayed and not how they are estimated.

`test(sufficient|score|probability)` reports the significance level of the observed sufficient statistics, the conditional scores tests, or the conditional probabilities tests, respectively. The default is `test(sufficient)`. If `terms()` is included in the specification, the conditional scores test and the conditional probabilities test are applied to each term providing conditional inference for several parameters simultaneously. All the statistics are computed at estimation time regardless of which is specified. Each statistic may thus also be displayed postestimation without having to refit the model; see [\[R\] exlogistic postestimation](#).

`mue(varlist)` specifies that median unbiased estimates (MUEs) be reported for the variables in `varlist`. By default, the conditional maximum likelihood estimates (CMLEs) are reported, except for those parameters for which the CMLEs are infinite. Specify `mue(_all)` if you want MUEs for all the `indepvars`.

`midp` instructs `exlogistic` to use the mid-*p*-value rule when computing the MUEs, significance levels, and confidence intervals. This adjustment is for the discreteness of the distribution and halves the value of the discrete probability of the observed statistic before adding it to the *p*-value. The mid-*p*-value rule cannot be applied to MUEs whose corresponding parameter CMLE is infinite.

`nolog` prevents the display of the enumeration log. By default, the enumeration log is displayed, showing the progress of computing the conditional distribution of the sufficient statistics.

## Remarks

Exact logistic regression is the estimation of the logistic model parameters by using the conditional distribution of the parameter sufficient statistics. The estimates are referred to as the conditional maximum likelihood estimates (CMLEs). This technique was first introduced by [Cox and Snell \(1989\)](#) as an alternative to using maximum likelihood estimation, which can perform poorly for small sample sizes. For stratified data, exact logistic regression is a small-sample alternative to conditional logistic regression. See [\[R\] logit](#), [\[R\] logistic](#), and [\[R\] clogit](#) to obtain maximum likelihood estimates (MLEs) for the logistic model and the conditional logistic model. For a comprehensive overview of exact logistic regression, see [Mehta and Patel \(1995\)](#).

Let  $Y_i$  denote a Bernoulli random variable where we observe the outcome  $Y_i = y_i$ ,  $i = 1, \dots, n$ . Associated with each independent observation is a  $1 \times p$  vector of covariates,  $\mathbf{x}_i$ . We will denote  $\pi_i = \Pr(Y_i = 1 \mid \mathbf{x}_i)$  and let the logit function model the relationship between  $Y_i$  and  $\mathbf{x}_i$ ,

$$\log\left(\frac{\pi_i}{1 - \pi_i}\right) = \theta + \mathbf{x}_i\boldsymbol{\beta}$$

where the constant term  $\theta$  and the  $p \times 1$  vector of regression parameters  $\boldsymbol{\beta}$  are unknown. The probability of observing  $Y_i = y_i$ ,  $i = 1, \dots, n$ , is

$$\Pr(\mathbf{Y} = \mathbf{y}) = \prod_{i=1}^n \pi_i^{y_i} (1 - \pi_i)^{1-y_i}$$

where  $\mathbf{Y} = (Y_1, \dots, Y_n)$  and  $\mathbf{y} = (y_1, \dots, y_n)$ . The MLEs for  $\theta$  and  $\boldsymbol{\beta}$  maximize the log of this function.

The sufficient statistics for  $\theta$  and  $\beta_j$ ,  $j = 1, \dots, p$ , are  $M = \sum_{i=1}^n Y_i$  and  $T_j = \sum_{i=1}^n Y_i x_{ij}$ , respectively, and we observe  $M = m$  and  $T_j = t_j$ . By default, `exlogistic` tallies the conditional distribution of  $\mathbf{T} = (T_1, \dots, T_p)$  given  $M = m$ . This distribution will have a size of  $\binom{n}{m}$ . (It would have a size of  $2^n$  without conditioning on  $M = m$ .) Denote one of these vectors  $\mathbf{T}^{(k)} = (t_1^{(k)}, \dots, t_p^{(k)})$ ,  $k = 1, \dots, N$ , with combinatorial coefficient (frequency)  $c_k$ ,  $\sum_{k=1}^N c_k = \binom{n}{m}$ . For each independent variable  $x_j$ ,  $j = 1, \dots, p$ , we reduce the conditional distribution further by conditioning on all other observed sufficient statistics  $T_l = t_l$ ,  $l \neq j$ . The conditional probability of observing  $T_j = t_j$  has the form

$$\Pr(T_j = t_j \mid T_l = t_l, l \neq j, M = m) = \frac{c_k e^{t_j \beta_j}}{\sum_k c_k e^{t_j^{(k)} \beta_j}}$$

where the sum is over the subset of  $\mathbf{T}$  vectors such that  $(T_1^{(k)} = t_1, \dots, T_j^{(k)} = t_j^{(k)}, \dots, T_p^{(k)} = t_p)$  and  $c$  is the combinatorial coefficient associated with the observed  $\mathbf{t}$ . The CMLE for  $\beta_j$  maximizes the log of this function.

Specifying nuisance variables in `condvars()` will reduce the size of the conditional distribution by conditioning on their observed sufficient statistics as well as conditioning on  $M = m$ . This reduces the amount of memory consumed at the cost of not obtaining regression estimates for those variables specified in `condvars()`.

Inferences from MLEs rely on asymptotics, and if your sample size is small, these inferences may not be valid. On the other hand, inferences from the CMLEs are exact in the sense that they use the conditional distribution of the sufficient statistics outlined above.

For small datasets, it is common for the dependent variable to be completely determined by the data. Here the MLEs and the CMLEs are unbounded. `exlogistic` will instead compute the MUE, the regression estimate that places the observed sufficient statistic at the median of the conditional distribution.

## ► Example 1

One example presented by [Mehta and Patel \(1995\)](#) is data from a prospective study of perinatal infection and human immunodeficiency virus type 1 (HIV-1). We use a variation of this dataset. There was an investigation [Hutto et al. \(1991\)](#) into whether the blood serum levels of glycoproteins CD4 and CD8 measured in infants at 6 months of age might predict their development of HIV infection. The blood serum levels are coded as ordinal values 0, 1, and 2.

```
. use http://www.stata-press.com/data/r11/hiv1
(prospective study of perinatal infection of HIV-1)
. list
```

|     | hiv              | cd4 | cd8 |
|-----|------------------|-----|-----|
| 1.  | 1                | 0   | 0   |
| 2.  | 0                | 0   | 0   |
| 3.  | 1                | 0   | 2   |
| 4.  | 1                | 1   | 0   |
| 5.  | 0                | 1   | 0   |
|     | (output omitted) |     |     |
| 46. | 0                | 2   | 1   |
| 47. | 0                | 2   | 2   |

We first obtain the MLEs from `logistic` so that we can compare the estimates and associated statistics with the CMLEs from `exlogistic`.

```
. logistic hiv cd4 cd8, coef
Logistic regression Number of obs = 47
 LR chi2(2) = 15.75
 Prob > chi2 = 0.0004
Log likelihood = -20.751687 Pseudo R2 = 0.2751
```

| hiv   | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-------|-----------|-----------|-------|-------|----------------------|-----------|
| cd4   | -2.541669 | .8392231  | -3.03 | 0.002 | -4.186517            | -.8968223 |
| cd8   | 1.658586  | .821113   | 2.02  | 0.043 | .0492344             | 3.267938  |
| _cons | .5132389  | .6809007  | 0.75  | 0.451 | -.8213019            | 1.84778   |

```
. exlogistic hiv cd4 cd8, coef
Enumerating sample-space combinations:
observation 1: enumerations = 2
observation 2: enumerations = 3
(output omitted)
observation 46: enumerations = 601
observation 47: enumerations = 326
Exact logistic regression
Number of obs = 47
Model score = 13.34655
Pr >= score = 0.0006
```

| hiv | Coef.     | Suff. | 2*Pr(Suff.) | [95% Conf. Interval] |           |
|-----|-----------|-------|-------------|----------------------|-----------|
| cd4 | -2.387632 | 10    | 0.0004      | -4.699633            | -.8221807 |
| cd8 | 1.592366  | 12    | 0.0528      | -.0137905            | 3.907876  |

`exlogistic` produced a log showing how many records are generated as it processes each observation. The primary purpose of the log is to provide feedback because generating the distribution can be time consuming, but we also see from the last entry that the joint distribution for the sufficient statistics for `cd4` and `cd8` conditioned on the total number of successes has 326 unique values (but a size of  $\binom{47}{14} = 341,643,774,795$ ).

The statistics for `logistic` are based on asymptotics: for a large sample size, each  $Z$  statistic will be approximately normally distributed (with a mean of zero and a standard deviation of one) if the associated regression parameter is zero. The question is whether a sample size of 47 is large enough.

On the other hand, the  $p$ -values computed by `exlogistic` are from the conditional distributions of the sufficient statistics for each parameter given the sufficient statistics for all other parameters. In this sense, these  $p$ -values are exact. By default, `exlogistic` reports the sufficient statistics for the regression parameters and the probability of observing a more extreme value. These are single-parameter tests for  $H_0: \beta_{cd4} = 0$  and  $H_0: \beta_{cd8} = 0$  versus the two-sided alternatives. The conditional scores test, located in the coefficient table header, is testing that both  $H_0: \beta_{cd4} = 0$  and  $H_0: \beta_{cd8} = 0$ . We find these  $p$ -values to be in fair agreement with the Wald and likelihood-ratio tests from `logistic`.

The confidence intervals for `exlogistic` are computed from the exact conditional distributions. The exact confidence intervals are asymmetrical about the estimate and are wider than the normal-based confidence intervals from `logistic`.

Both estimation techniques indicate that the incidence of HIV infection decreases with increasing CD4 blood serum levels and increases with increasing CD8 blood serum levels. The constant term is missing from the exact logistic coefficient table because we conditioned out its observed sufficient statistic when tallying the joint distribution of the sufficient statistics for the `cd4` and `cd8` parameters.

The `test()` option provides two other test statistics used in exact logistic: the conditional scores test, `test(score)`, and the conditional probabilities test, `test(probability)`. For comparison, we display the individual parameter conditional scores tests.

```
. exlogistic, test(score) coef
```

```
Exact logistic regression
```

```
Number of obs = 47
Model score = 13.34655
Pr >= score = 0.0006
```

| hiv | Coef.     | Score    | Pr>=Score | [95% Conf. Interval] |           |
|-----|-----------|----------|-----------|----------------------|-----------|
| cd4 | -2.387632 | 12.88022 | 0.0003    | -4.699633            | -.8221807 |
| cd8 | 1.592366  | 4.604816 | 0.0410    | -.0137905            | 3.907876  |

For the probabilities test, the probability statistic is computed from (1) in *Methods and formulas* with  $\beta = 0$ . For this example, the significance of the probabilities tests matches the scores tests so they are not displayed here.

◀

## □ Technical note

Typically, the value of  $\theta$ , the constant term, is of little interest, as well as perhaps some of the parameters in  $\beta$ , but we need to include all parameters in the model to correctly specify it. By conditioning out the nuisance parameters, we can reduce the size of the joint conditional distribution that is used to estimate the regression parameters of interest. The `condvars()` option allows you to specify a *varlist* of nuisance variables. By default, `exlogistic` conditions on the sufficient statistic of  $\theta$ , which is the number of successes. You can save computation time and computer memory by using the `condvars()` option because infeasible values of the sufficient statistics associated with the variables in `condvars()` can be dropped from consideration before all  $n$  observations are processed.

Specifying some of your independent variables in `condvars()` will not change the estimated regression coefficients of the remaining independent variables. For instance, in [example 1](#), if we instead type

```
. exlogistic hiv cd4, condvars(cd8) coef
```

the regression coefficient for `cd4` (as well as all associated inference) will be identical.

One reason to have multiple variables in *indepvars* is to make conditional inference of several parameters simultaneously by using the `terms()` option. If you do not wish to test several parameters simultaneously, it may be more efficient to obtain estimates for individual variables by calling `exlogistic` multiple times with one variable in *indepvars* and all other variables listed in `condvars()`. The estimates will be the same as those with all variables in *indepvars*.

□

## □ Technical note

If you fit a `clogit` model to the HIV data from [example 1](#), you will find that the estimates differ from those with `exlogistic`. (To fit the `clogit` model, you will have to create a group variable that includes all observations.) The regression estimates will be different because `clogit` conditions on the constant term only, whereas the estimates from `exlogistic` condition on the sufficient statistic of the other regression parameter as well as the constant term.

□

## ▷ Example 2

The HIV data presented in table IV of [Mehta and Patel \(1995\)](#) are in a binomial form, where the variable `hiv` contains the HIV cases that tested positive and the variable `n` contains the number of individuals with the same CD4 and CD8 levels, the binomial number-of-trials parameter. Here *deprvar* is `hiv`, and we use the `binomial(n)` option to identify the number-of-trials variable.

```
. use http://www.stata-press.com/data/r11/hiv_n, clear
(prospective study of perinatal infection of HIV-1; binomial form)
. list
```

|    | cd4 | cd8 | hiv | n  |
|----|-----|-----|-----|----|
| 1. | 0   | 2   | 1   | 1  |
| 2. | 1   | 2   | 2   | 2  |
| 3. | 0   | 0   | 4   | 7  |
| 4. | 1   | 1   | 4   | 12 |
| 5. | 2   | 2   | 1   | 3  |
| 6. | 1   | 0   | 2   | 7  |
| 7. | 2   | 0   | 0   | 2  |
| 8. | 2   | 1   | 0   | 13 |

Further, the `cd4` and `cd8` variables of the `hiv` dataset are actually factor variables, where each has the ordered levels of (0, 1, 2). Another approach to the analysis is to use indicator variables, and following [Mehta and Patel \(1995\)](#), we used a 0–1 coding scheme that will give us the odds ratio of level 0 versus 2 and level 1 versus 2.

```
. gen byte cd4_0 = (cd4==0)
. gen byte cd4_1 = (cd4==1)
. gen byte cd8_0 = (cd8==0)
. gen byte cd8_1 = (cd8==1)
. exlogistic hiv cd4_0 cd4_1 cd8_0 cd8_1, terms(cd4=cd4_0 cd4_1,
> cd8=cd8_0 cd8_1) binomial(n) test(probability) saving(dist) nolog
note: saving distribution to file dist
note: CMLE estimate for cd4_0 is +inf; computing MUE
note: CMLE estimate for cd4_1 is +inf; computing MUE
note: CMLE estimate for cd8_0 is -inf; computing MUE
note: CMLE estimate for cd8_1 is -inf; computing MUE
```

```
Exact logistic regression Number of obs = 47
Binomial variable: n Model prob. = 3.19e-06
 Pr <= prob. = 0.0011
```

|     | hiv   | Odds Ratio | Prob.    | Pr<=Prob. | [95% Conf. Interval] |          |
|-----|-------|------------|----------|-----------|----------------------|----------|
| cd4 |       |            | .0007183 | 0.0055    |                      |          |
|     | cd4_0 | 18.82831*  | .007238  | 0.0072    | 1.714079             | +Inf     |
|     | cd4_1 | 11.53732*  | .0063701 | 0.0105    | 1.575285             | +Inf     |
| cd8 |       |            | .0053212 | 0.0323    |                      |          |
|     | cd8_0 | .1056887*  | .0289948 | 0.0290    | 0                    | 1.072531 |
|     | cd8_1 | .0983388*  | .0241503 | 0.0242    | 0                    | .9837203 |

```
(*) median unbiased estimates (MUE)
. matrix list e(sufficient)
e(sufficient)[1,4]
 cd4_0 cd4_1 cd8_0 cd8_1
r1 5 8 6 4
. display e(n_possible)
1091475
```

Here we used `terms()` to specify two terms in the model, `cd4` and `cd8`, that make up the `cd4` and `cd8` indicator variables. By doing so, we obtained a conditional probabilities test for `cd4`, simultaneously testing both `cd4_0` and `cd4_1`, and for `cd8`, simultaneously testing both `cd8_0` and `cd8_1`. The significance levels for the two terms are 0.0055 and 0.0323, respectively.



This example also illustrates instances where the dependent variable is completely determined by the independent variables and CMLEs are infinite. If we try to obtain MLEs, `logistic` will drop each variable and then terminate with a no-data error, error number 2000.

```
. use http://www.stata-press.com/data/r11/hiv_n, clear
(prospective study of perinatal infection of HIV-1; binomial form)

. gen byte cd4_0 = (cd4==0)
. gen byte cd4_1 = (cd4==1)
. gen byte cd8_0 = (cd8==0)
. gen byte cd8_1 = (cd8==1)

. expand n
(39 observations created)

. logistic hiv cd4_0 cd4_1 cd8_0 cd8_1
note: cd4_0 != 0 predicts success perfectly
 cd4_0 dropped and 8 obs not used

note: cd4_1 != 0 predicts success perfectly
 cd4_1 dropped and 21 obs not used

note: cd8_0 != 0 predicts failure perfectly
 cd8_0 dropped and 2 obs not used

outcome = cd8_1 <= 0 predicts data perfectly
r(2000);
```

In the previous example, `exlogistic` generated the joint conditional distribution of  $T_{cd4\_0}$ ,  $T_{cd4\_1}$ ,  $T_{cd8\_0}$ , and  $T_{cd8\_1}$  given  $M = 14$  (the number of individuals that tested positive), and for reference, we listed the observed sufficient statistics that are stored in the matrix `e(sufficient)`. Below we take that distribution and further condition on  $T_{cd4\_1} = 8$ ,  $T_{cd8\_0} = 6$ , and  $T_{cd8\_1} = 4$ , giving the conditional distribution of  $T_{cd4\_0}$ . Here we see that the observed sufficient statistic  $T_{cd4\_0} = 5$  is last in the sorted listing or, equivalently,  $T_{cd4\_0}$  is at the domain boundary of the conditional probability distribution. When this occurs, the conditional probability distribution is monotonically increasing in  $\beta_{cd4\_0}$  and a maximum does not exist.

```
. use dist, clear
. keep if cd4_1==8 & cd8_0==6 & cd8_1==4
(4139 observations deleted)
. list, sep(0)
```

|    | _f_      | cd4_0 | cd4_1 | cd8_0 | cd8_1 |
|----|----------|-------|-------|-------|-------|
| 1. | 1668667  | 0     | 8     | 6     | 4     |
| 2. | 18945542 | 1     | 8     | 6     | 4     |
| 3. | 55801053 | 2     | 8     | 6     | 4     |
| 4. | 55867350 | 3     | 8     | 6     | 4     |
| 5. | 17423175 | 4     | 8     | 6     | 4     |
| 6. | 1091475  | 5     | 8     | 6     | 4     |

When the CMLEs are infinite, the MUEs are computed (Hirji, Tsiatis, and Mehta 1989). For the  $cd4\_0$  estimate, we compute the value  $\bar{\beta}_{cd4\_0}$  such that

$$\Pr(T_{cd4\_0} \geq 5 \mid \beta_{cd4\_0} = \bar{\beta}_{cd4\_0}, T_{cd4\_1} = 8, T_{cd8\_0} = 6, T_{cd8\_1} = 4, M = 14) = 1/2$$

using (1) in *Methods and formulas*.

The output is in agreement with example 1: there is an increase in risk of HIV infection for a CD4 blood serum level of 0 relative to a level of 2 and for a level of 1 relative to a level of 2; there is a decrease in risk of HIV infection for a CD8 blood serum level of 0 relative to a level of 2 and for a level of 1 relative to a level of 2.

We also displayed `e(n_possible)`. This is the combinatorial coefficient associated with the observed sufficient statistics. The same value is found in the `_f_` variable of the conditional distribution dataset listed above. The size of the distribution is  $\binom{47}{14} = 341,643,774,795$ . This can be verified by summing the `_f_` variable of the generated conditional distribution dataset.

```
. use dist, clear
. summarize _f_, meanonly
. di %15.1f r(sum)
341643774795.0
```



► Example 3

One can think of exact logistic regression as a covariate-adjusted exact binomial. To demonstrate this point, we will use `exlogistic` to compute a binomial confidence interval for  $m$  successes of  $n$  trials, by fitting the constant-only model, and we will compare it with the confidence interval computed by `ci` (see [R] `ci`). We will use the `saving()` option to retain the dataset containing the feasible values for the constant term sufficient statistic, namely, the number of successes,  $m$ , given  $n$  trials and their associated combinatorial coefficients  $\binom{n}{m}$ ,  $m = 0, 1, \dots, n$ .

```
. input y
 y
1. 1
2. 0
3. 1
4. 0
5. 1
6. 1
7. end
. ci y, binomial
```

| Variable | Obs | Mean     | Std. Err. | — Binomial Exact —<br>[95% Conf. Interval] |          |
|----------|-----|----------|-----------|--------------------------------------------|----------|
| y        | 6   | .6666667 | .1924501  | .2227781                                   | .9567281 |

```
. exlogistic y, estconstant nolog coef saving(binom)
note: saving distribution to file binom
```

Exact logistic regression

| Number of obs = 6 |          |       |             |                      |          |
|-------------------|----------|-------|-------------|----------------------|----------|
| y                 | Coef.    | Suff. | 2*Pr(Suff.) | [95% Conf. Interval] |          |
| _cons             | .6931472 | 4     | 0.6875      | -1.24955             | 3.096017 |

We use the postestimation program `estat predict` to transform the estimated constant term and its confidence bounds by using the inverse logit function, `invlogit()` (see [D] `functions`). The standard error for the estimated probability is computed using the delta method.

```
. estat predict
```

| y           | Predicted | Std. Err. | [95% Conf. Interval] |        |
|-------------|-----------|-----------|----------------------|--------|
| Probability | 0.6667    | 0.1925    | 0.2228               | 0.9567 |

```
. use binom, replace
```

```
. list, sep(0)
```

|    | _f_ | _cons_ |
|----|-----|--------|
| 1. | 1   | 0      |
| 2. | 6   | 1      |
| 3. | 15  | 2      |
| 4. | 20  | 3      |
| 5. | 15  | 4      |
| 6. | 6   | 5      |
| 7. | 1   | 6      |

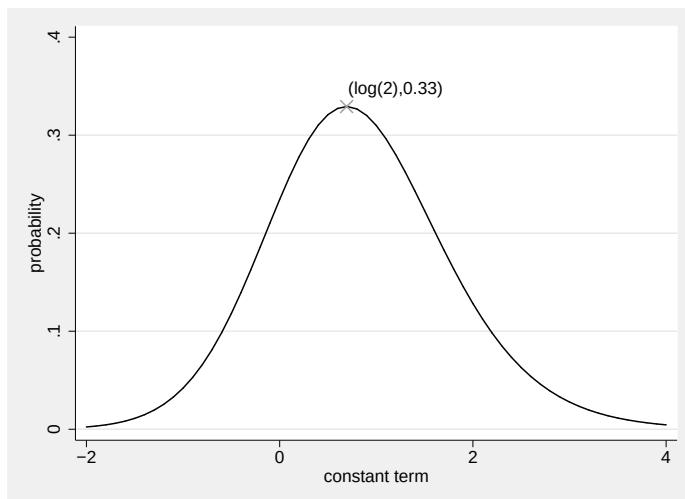
Examining the listing of the generated data, the values contained in the variable `_cons_` are the feasible values of  $M$ , and the values contained in the variable `_f_` are the binomial coefficients  $\binom{6}{m}$

with total  $\sum_{m=0}^6 \binom{6}{m} = 2^6 = 64$ . In the coefficient table, the sufficient statistic for the constant term, labeled `Suff.`, is  $m = 4$ . This value is located at record 5 of the dataset. Therefore, the two-tailed probability of the sufficient statistic is computed as  $0.6875 = 2(15 + 6 + 1)/64$ .

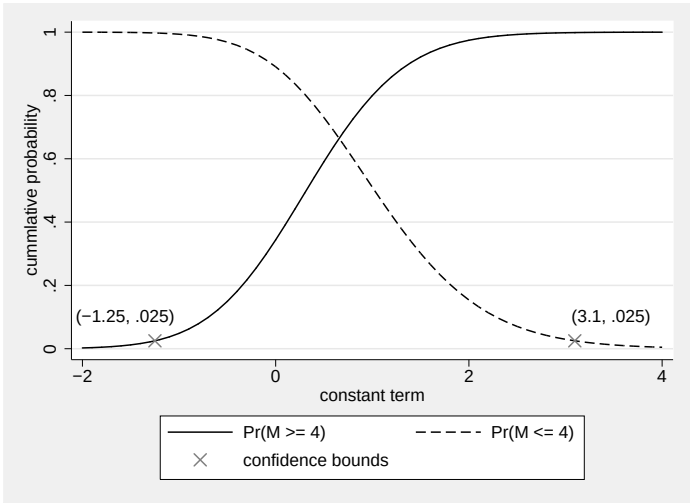
The constant term is the value of  $\theta$  that maximizes the probability of observing  $M = 4$ ; see (1) of *Methods and formulas*:

$$\Pr(M = 4|\theta) = \frac{15e^{4\alpha}}{1 + 6e^{\alpha} + 15e^{2\alpha} + 20e^{3\alpha} + 15e^{4\alpha} + 6e^{5\alpha} + e^{6\alpha}}$$

The maximum is at the value  $\theta = \log 2$ , which is demonstrated in the figure below.



The lower and upper confidence bounds are the values of  $\theta$  such that  $\Pr(M \geq 4|\theta) = 0.025$  and  $\Pr(M \leq 4|\theta) = 0.025$ , respectively. These probabilities are plotted in the figure below for  $\theta \in [-2, 4]$ .



◀

➤ **Example 4**

This example demonstrates the `group()` option, which allows the analysis of stratified data. Here the logistic model is

$$\log \left( \frac{\pi_{ik}}{1 - \pi_{ik}} \right) = \theta_k + \mathbf{x}_{ki} \boldsymbol{\beta}$$

where  $k$  indexes the  $s$  strata,  $k = 1, \dots, s$ , and  $\theta_k$  is the strata-specific constant term whose sufficient statistic is  $M_k = \sum_{i=1}^{n_k} Y_{ki}$ .

Mehta and Patel (1995) use a case-control study to demonstrate this model, which is useful in comparing the estimates from `exlogistic` and `clogit`. This study was intended to determine the role of birth complications in people with schizophrenia (Garsd 1988). Siblings from seven families took part in the study, and each individual was classified as normal or schizophrenic. A birth complication index is recorded for each individual that ranges from 0, an uncomplicated birth, to 15, a very complicated birth. Some of the frequencies contained in variable `f` are greater than 1, and these count different births at different times where the individual has the same birth complications index, found in variable `BCindex`.

```
. use http://www.stata-press.com/data/r11/schizophrenia, clear
(case-control study on birth complications for people with schizophrenia)
. list, sepby(family)
```

|     | family | BCindex | schizo | f |
|-----|--------|---------|--------|---|
| 1.  | 1      | 6       | 0      | 1 |
| 2.  | 1      | 7       | 0      | 1 |
| 3.  | 1      | 3       | 0      | 2 |
| 4.  | 1      | 2       | 0      | 3 |
| 5.  | 1      | 5       | 0      | 1 |
| 6.  | 1      | 0       | 0      | 1 |
| 7.  | 1      | 15      | 1      | 1 |
|     |        |         |        |   |
| 8.  | 2      | 2       | 1      | 1 |
| 9.  | 2      | 0       | 0      | 1 |
|     |        |         |        |   |
| 10. | 3      | 2       | 0      | 1 |
| 11. | 3      | 9       | 1      | 1 |
| 12. | 3      | 1       | 0      | 1 |
|     |        |         |        |   |
| 13. | 4      | 2       | 1      | 1 |
| 14. | 4      | 0       | 0      | 4 |
|     |        |         |        |   |
| 15. | 5      | 3       | 1      | 1 |
| 16. | 5      | 6       | 0      | 1 |
| 17. | 5      | 0       | 1      | 1 |
|     |        |         |        |   |
| 18. | 6      | 3       | 0      | 1 |
| 19. | 6      | 0       | 1      | 1 |
| 20. | 6      | 0       | 0      | 2 |
|     |        |         |        |   |
| 21. | 7      | 2       | 0      | 1 |
| 22. | 7      | 6       | 1      | 1 |

```
. exlogistic schizo BCindex [fw=f], group(family) test(score) coef
```

Enumerating sample-space combinations:

```
observation 1: enumerations = 2
observation 2: enumerations = 3
observation 3: enumerations = 4
observation 4: enumerations = 5
observation 5: enumerations = 6
```

(output omitted)

```
observation 21: enumerations = 72
observation 22: enumerations = 40
```

Exact logistic regression

Group variable: family

```
Number of obs = 29
Number of groups = 7
Obs per group: min = 2
 avg = 4.1
 max = 10
Model score = 6.32803
Pr >= score = 0.0167
```

| schizo  | Coef.    | Score    | Pr>=Score | [95% Conf. Interval] |          |
|---------|----------|----------|-----------|----------------------|----------|
| BCindex | .3251178 | 6.328033 | 0.0167    | .0223423             | .7408832 |

The asymptotic alternative for this model can be estimated using `clogit` (equivalently, `xtlogit, fe`) and is listed below for comparison. We must expand the data because `clogit` will not accept frequency weights if they are not constant within the groups.

```
. expand f
(7 observations created)
. clogit schizo BCindex, group(family) nolog
note: multiple positive outcomes within groups encountered.
Conditional (fixed-effects) logistic regression Number of obs = 29
 LR chi2(1) = 5.20
 Prob > chi2 = 0.0226
 Pseudo R2 = 0.2927

Log likelihood = -6.2819819
```

| schizo  | Coef.    | Std. Err. | z    | P> z  | [95% Conf. Interval] |         |
|---------|----------|-----------|------|-------|----------------------|---------|
| BCindex | .3251178 | .1678981  | 1.94 | 0.053 | -.0039565            | .654192 |

Both techniques compute the same regression estimate for the `BCindex`, which might not be too surprising because both estimation techniques condition on the total number of successes in each group. The difference lies in the  $p$ -values and confidence intervals. The  $p$ -value testing  $H_0: \beta_{\text{BCindex}} = 0$  is approximately 0.0167 for the exact conditional scores test and 0.053 for the asymptotic Wald test. Moreover, the exact confidence interval is asymmetric about the estimate and does not contain zero.

❑ Technical note

The `memory(#)` option limits the amount of memory that `exlogistic` will consume when computing the conditional distribution of the parameter sufficient statistics. `memory()` is independent of the system setting `c(memory)` (see `set memory` in [\[D\] memory](#)), and it is possible for `exlogistic` to exceed the memory limit specified in `c(memory)` without terminating. By default, a log is provided that displays the number of enumerations (the size of the conditional distribution) after processing each observation. Typically, you will see the number of enumerations increase, and then at some point they will decrease as the multivariate shift algorithm ([Hirji, Mehta, and Patel 1987](#)) determines that some of the enumerations cannot achieve the observed sufficient statistics of the conditioning variables. When the algorithm is complete, however, it is necessary to store the conditional distribution of the parameter sufficient statistics as a dataset. It is possible, therefore, to get a memory error when the algorithm has completed and `c(memory)` is not large enough to store the conditional distribution.

❑ Technical note

Computing the conditional distributions and reported statistics requires data sorting and numerical comparisons. If there is at least one single-precision variable specified in the model, `exlogistic` will make comparisons with a relative precision of  $2^{-5}$ . Otherwise, a relative precision of  $2^{-11}$  is used. Be careful if you use `recast` to promote a single-precision variable to double precision (see [\[D\] recast](#)). You might try listing the data in full precision (maybe `%20.15g`; see [\[D\] format](#)) to make sure that this is really what you want. See [\[D\] data types](#) for information on precision of numeric storage types.

## Saved results

exlogistic saves the following in `e()`:

### Scalars

|                            |                                                                                                                    |
|----------------------------|--------------------------------------------------------------------------------------------------------------------|
| <code>e(N)</code>          | number of observations                                                                                             |
| <code>e(k_groups)</code>   | number of groups                                                                                                   |
| <code>e(n_possible)</code> | number of distinct possible outcomes where <code>sum(sufficient)</code> equals observed <code>e(sufficient)</code> |
| <code>e(n_trials)</code>   | binomial number-of-trials parameter                                                                                |
| <code>e(sum_y)</code>      | sum of <i>depvar</i>                                                                                               |
| <code>e(k_indvars)</code>  | number of independent variables                                                                                    |
| <code>e(k_terms)</code>    | number of model terms                                                                                              |
| <code>e(k_condvars)</code> | number of conditioning variables                                                                                   |
| <code>e(condcons)</code>   | conditioned on the constant(s) indicator                                                                           |
| <code>e(midp)</code>       | mid- <i>p</i> -value rule indicator                                                                                |
| <code>e(eps)</code>        | relative difference tolerance                                                                                      |

### Macros

|                                   |                                           |
|-----------------------------------|-------------------------------------------|
| <code>e(cmd)</code>               | exlogistic                                |
| <code>e(cmdline)</code>           | command as typed                          |
| <code>e(title)</code>             | Exact logistic regression                 |
| <code>e(depvar)</code>            | dependent variable                        |
| <code>e(indvars)</code>           | independent variables                     |
| <code>e(condvars)</code>          | conditional variables                     |
| <code>e(groupvar)</code>          | group variable                            |
| <code>e(binomial)</code>          | binomial number-of-trials variable        |
| <code>e(level)</code>             | confidence level                          |
| <code>e(wtype)</code>             | weight type                               |
| <code>e(wexp)</code>              | weight expression                         |
| <code>e(datasignature)</code>     | the checksum                              |
| <code>e(datasignaturevars)</code> | variables used in calculation of checksum |
| <code>e(properties)</code>        | b                                         |
| <code>e(estat_cmd)</code>         | program used to implement estat           |
| <code>e(predict)</code>           | program used to implement predict         |
| <code>e(marginsnotok)</code>      | predictions disallowed by margins         |

*(Continued on next page)*

**Matrices**

|                                |                                                                                 |
|--------------------------------|---------------------------------------------------------------------------------|
| <code>e(b)</code>              | coefficient vector                                                              |
| <code>e(mue_indicators)</code> | indicator for elements of <code>e(b)</code> estimated using MUE instead of CMLE |
| <code>e(se)</code>             | <code>e(b)</code> standard errors (CMLEs only)                                  |
| <code>e(ci)</code>             | matrix of <code>e(level)</code> confidence intervals for <code>e(b)</code>      |
| <code>e(sum_y_groups)</code>   | sum of <code>e(depvar)</code> for each group                                    |
| <code>e(N_g)</code>            | number of observations in each group                                            |
| <code>e(sufficient)</code>     | sufficient statistics for <code>e(b)</code>                                     |
| <code>e(p_sufficient)</code>   | $p$ -value for <code>e(sufficient)</code>                                       |
| <code>e(scoretest)</code>      | conditional scores tests for <i>indepvars</i>                                   |
| <code>e(p_scoretest)</code>    | $p$ -values for <code>e(scoretest)</code>                                       |
| <code>e(probtest)</code>       | conditional probabilities tests for <i>indepvars</i>                            |
| <code>e(p_probtest)</code>     | $p$ -value for <code>e(probtest)</code>                                         |
| <code>e(scoretest_m)</code>    | conditional scores tests for model terms                                        |
| <code>e(p_scoretest_m)</code>  | $p$ -value for <code>e(scoretest_m)</code>                                      |
| <code>e(probtest_m)</code>     | conditional probabilities tests for model terms                                 |
| <code>e(p_probtest_m)</code>   | $p$ -value for <code>e(probtest_m)</code>                                       |

**Functions**

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`exlogistic` is implemented as an ado-file.

Methods and formulas are presented under the following headings:

*Sufficient statistics*  
*Conditional distribution and CMLE*  
*Median unbiased estimates and exact CI*  
*Conditional hypothesis tests*  
*Sufficient-statistic  $p$ -value*

## Sufficient statistics

Let  $\{Y_1, Y_2, \dots, Y_n\}$  be a set of  $n$  independent Bernoulli random variables, each of which can realize two outcomes,  $\{0, 1\}$ . For each  $i = 1, \dots, n$ , we observe  $Y_i = y_i$ , and associated with each observation is the covariate row vector of length  $p$ ,  $\mathbf{x}_i = (x_{i1}, \dots, x_{ip})$ . Denote  $\boldsymbol{\beta} = (\beta_1, \dots, \beta_p)^T$  to be the column vector of regression parameters and  $\theta$  to be the constant. The sufficient statistic for  $\beta_j$  is  $T_j = \sum_{i=1}^n Y_i x_{ij}$ ,  $j = 1, \dots, p$ , and for  $\theta$  is  $M = \sum_{i=1}^n Y_i$ . We observe  $T_j = t_j$ ,  $t_j = \sum_{i=1}^n y_i x_{ij}$ , and  $M = m$ ,  $m = \sum_{i=1}^n y_i$ . The probability of observing  $(Y_1 = y_1, Y_2 = y_2, \dots, Y_n = y_n)$  is

$$\Pr(Y_1 = y_1, \dots, Y_n = y_n \mid \boldsymbol{\beta}, \mathbf{X}) = \frac{\exp(m\theta + \mathbf{t}\boldsymbol{\beta})}{\prod_{i=1}^n \{1 + \exp(\theta + \mathbf{x}_i\boldsymbol{\beta})\}}$$

where  $\mathbf{t} = (t_1, \dots, t_p)$  and  $\mathbf{X} = (\mathbf{x}_1^T, \dots, \mathbf{x}_n^T)^T$ .

The joint distribution of the sufficient statistics  $\mathbf{T}$  is obtained by summing over all possible binary sequences  $Y_1, \dots, Y_n$  such that  $\mathbf{T} = \mathbf{t}$  and  $M = m$ . This probability function is

$$\Pr(T_1 = t_1, \dots, T_p = t_p, M = m \mid \boldsymbol{\beta}, \mathbf{X}) = \frac{c(\mathbf{t}, m) \exp(m\theta + \mathbf{t}\boldsymbol{\beta})}{\prod_{i=1}^n \{1 + \exp(\theta + \mathbf{x}_i\boldsymbol{\beta})\}}$$

where  $c(\mathbf{t}, m)$  is the combinatorial coefficient of  $(\mathbf{t}, m)$  or the number of distinct binary sequences  $Y_1, \dots, Y_n$  such that  $\mathbf{T} = \mathbf{t}$  and  $M = m$  (Cox and Snell 1989).



## Conditional distribution and CMLE

Without loss of generality, we will restrict our discussion to computing the CMLE of  $\beta_1$ . If we condition on observing  $M = m$  and  $T_2 = t_2, \dots, T_p = t_p$ , the probability function of  $(T_1 \mid \beta_1, T_2 = t_2, \dots, T_p = t_p, M = m)$  is

$$\Pr(T_1 = t_1 \mid \beta_1, T_2 = t_2, \dots, T_p = t_p, M = m) = \frac{c(\mathbf{t}, m)e^{t_1\beta_1}}{\sum_u c(u, t_2, \dots, t_p, m)e^{u\beta_1}} \quad (1)$$

where the sum in the denominator is over all possible values of  $T_1$  such that  $M = m$  and  $T_2 = t_2, \dots, T_p = t_p$  and  $c(u, t_2, \dots, t_p, m)$  is the combinatorial coefficient of  $(u, t_2, \dots, t_p, m)$  (Cox and Snell 1989). The CMLE for  $\beta_1$  is the value  $\hat{\beta}_1$  that maximizes the log of (1). This optimization task is carried out by `m1`, using the conditional frequency distribution of  $(T_1 \mid T_2 = t_2, \dots, T_p = t_p, M = m)$  as a dataset. Generating the joint conditional distribution is efficiently computed using the multivariate shift algorithm described by Hirji, Mehta, and Patel (1987).

Difficulties in computing  $\hat{\beta}_1$  arise if the observed  $(T_1 = t_1, \dots, T_p = t_p, M = m)$  lies on the boundaries of the distribution of  $(T_1 \mid T_2 = t_2, \dots, T_p = t_p, M = m)$ , where the conditional probability function is monotonically increasing (or decreasing) in  $\beta_1$ . Here the CMLE is plus infinity if it is on the upper boundary,  $\Pr(T_1 \leq t_1 \mid T_2 = t_2, \dots, T_p = t_p, M = m) = 1$ , and is minus infinity if it is on the lower boundary of the distribution,  $\Pr(T_1 \geq t_1 \mid T_2 = t_2, \dots, T_p = t_p, M = m) = 1$ . This concept is demonstrated in example 2. When infinite CMLEs occur, the MUE is computed.

## Median unbiased estimates and exact CI

The MUE is computed using the technique outlined by Hirji, Tsiatis, and Mehta (1989). First, we find the values of  $\beta_1^{(u)}$  and  $\beta_1^{(l)}$  such that

$$\begin{aligned} \Pr(T_1 \leq t_1 \mid \beta_1 = \beta_1^{(u)}, T_2 = t_2, \dots, T_p = t_p, M = m) &= \\ \Pr(T_1 \geq t_1 \mid \beta_1 = \beta_1^{(l)}, T_2 = t_2, \dots, T_p = t_p, M = m) &= 1/2 \end{aligned} \quad (2)$$

The MUE is then  $\bar{\beta}_1 = (\beta_1^{(l)} + \beta_1^{(u)})/2$ . However, if  $T_1$  is equal to the minimum of the domain of the conditional distribution,  $\beta_1^{(l)}$  does not exist and  $\bar{\beta}_1 = \beta_1^{(u)}$ . If  $T_1$  is equal to the maximum of the domain of the conditional distribution,  $\beta_1^{(u)}$  does not exist and  $\bar{\beta}_1 = \beta_1^{(l)}$ .

Confidence bounds for  $\beta$  are computed similarly, except that we substitute  $\alpha/2$  for  $1/2$  in (2), where  $1 - \alpha$  is the confidence level. Here  $\beta_1^{(l)}$  would then be the lower confidence bound and  $\beta_1^{(u)}$  would be the upper confidence bound (see example 3).

## Conditional hypothesis tests

To test  $H_0: \beta_1 = 0$  versus  $H_1: \beta_1 \neq 0$ , we obtain the exact  $p$ -value from  $\sum_{u \in E} f_1(u) - f_1(t_1)/2$  if the mid- $p$ -value rule is used and  $\sum_{u \in E} f_1(u)$  otherwise. Here  $E$  is a critical region, and we define  $f_1(u) = \Pr(T_1 = u \mid \beta_1 = 0, T_2 = t_2, \dots, T_p = t_p, M = m)$  for ease of notation. There are two popular ways to define the critical region: the conditional probabilities test and the conditional scores test (Mehta and Patel 1995). The critical region when using the conditional probabilities test is all values of the sufficient statistic for  $\beta_1$  that have a probability less than or equal to that of the observed  $t_1$ ,  $E_p = \{u : f_1(u) \leq f_1(t_1)\}$ . The critical region of the conditional scores test is defined as all values of the sufficient statistic for  $\beta_1$  such that its score is greater than or equal to that of  $t_1$ ,

$$E_s = \{u : (u - \mu_1)^2/\sigma_1^2 \geq (t_1 - \mu_1)^2/\sigma_1^2\}$$

Here  $\mu_1$  and  $\sigma_1^2$  are the mean and variance of  $(T_1 \mid \beta_1 = 0, T_2 = t_2, \dots, T_p = t_p, M = m)$ .

The score statistic is defined as

$$\left\{ \frac{\partial \ell(\beta)}{\partial \beta} \right\}^2 \left[ -E \left\{ \frac{\partial^2 \ell(\beta)}{\partial \beta^2} \right\} \right]^{-1}$$

evaluated at  $H_0: \beta = 0$ , where  $\ell$  is the log of (1). The score test simplifies to  $(t - E[T|\beta])^2 / \text{var}(T|\beta)$  (Hirji 2006), where the mean and variance are computed from the conditional distribution of the sufficient statistic with  $\beta = 0$  and  $t$  is the observed sufficient statistic.

## Sufficient-statistic p-value

The  $p$ -value for testing  $H_0: \beta_1 = 0$  versus the two-sided alternative when  $(T_1 = t_1 \mid T_2 = t_2, \dots, T_p = t_p)$  is computed as  $2 \times \min(p_l, p_u)$ , where

$$p_l = \frac{\sum_{u \leq t_1} c(u, t_2, \dots, t_p, m)}{\sum_u c(u, t_2, \dots, t_p, m)}$$

$$p_u = \frac{\sum_{u \geq t_1} c(u, t_2, \dots, t_p, m)}{\sum_u c(u, t_2, \dots, t_p, m)}$$

It is the probability of observing a more extreme  $T_1$ .

## References

- Cox, D. R., and E. J. Snell. 1989. *Analysis of Binary Data*. 2nd ed. London: Chapman & Hall.
- Garsd, A. 1988. Schizophrenia and birth complications. Unpublished manuscript.
- Hirji, K. F. 2006. *Exact Analysis of Discrete Data*. Boca Raton: Chapman & Hall/CRC.
- Hirji, K. F., C. R. Mehta, and N. R. Patel. 1987. Computing distributions for exact logistic regression. *Journal of the American Statistical Association* 82: 1110–1117.
- Hirji, K. F., A. A. Tsiatis, and C. R. Mehta. 1989. Median unbiased estimation for binary data. *American Statistician* 43: 7–11.
- Hutto, C., W. P. Parks, S. Lai, M. T. Mastrucci, C. Mitchell, J. Muñoz, E. Trapido, I. M. Master, and G. B. Scott. 1991. A hospital-based prospective study of perinatal infection with human immunodeficiency virus type 1. *Journal of Pediatrics* 118: 347–353.
- Mehta, C. R., and N. R. Patel. 1995. Exact logistic regression: Theory and examples. *Statistics in Medicine* 14: 2143–2160.

## Also see

- [R] **exlogistic postestimation** — Postestimation tools for exlogistic
- [R] **binreg** — Generalized linear models: Extensions to the binomial family
- [R] **clogit** — Conditional (fixed-effects) logistic regression
- [R] **expoisson** — Exact Poisson regression
- [R] **logit** — Logistic regression, reporting coefficients
- [R] **logistic** — Logistic regression, reporting odds ratios
- [U] **20 Estimation and postestimation commands**

Description

The following postestimation commands are of special interest after `exlogistic`:

| command                    | description                                                 |
|----------------------------|-------------------------------------------------------------|
| <code>estat predict</code> | single-observation prediction                               |
| <code>estat se</code>      | report odds ratio or coefficient asymptotic standard errors |

For information about these commands, see below.

The following standard postestimation command is also available:

| command                      | description               |
|------------------------------|---------------------------|
| <code>estat summarize</code> | estimation sample summary |

`estat summarize` is not allowed if the `binomial()` option was specified in `exlogistic`.

Special-interest postestimation commands

`estat predict` computes a predicted probability (or linear predictor), its asymptotic standard error, and its exact confidence interval for 1 observation. Predictions are carried out by estimating the constant coefficient after shifting the independent variables and conditioned variables by the values specified in the `at()` option or by their medians. Therefore, predictions must be done with the estimation sample in memory. If a different dataset is used or if the dataset is modified, then an error will result.

`estat se` reports odds ratio or coefficient asymptotic standard errors. The estimates are stored in the matrix `r(estimates)`.

Syntax for `estat predict`

`estat predict` [*, options*]

| <i>options</i>                         | description                                                                     |
|----------------------------------------|---------------------------------------------------------------------------------|
| <code>pr</code>                        | probability; the default                                                        |
| <code>xb</code>                        | linear effect                                                                   |
| <code>at(<i>atspec</i>)</code>         | use the specified values for the <i>indepvars</i> and <i>condvars()</i>         |
| <code>level(#)</code>                  | set confidence level for the predicted value; default is <code>level(95)</code> |
| <code>memory(#[<i>b k m g</i>])</code> | set limit on memory usage; default is <code>memory(10m)</code>                  |
| <code>nolog</code>                     | do not display the enumeration log                                              |

These statistics are available only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for estat predict

`pr`, the default, calculates the probability.

`xb` calculates the linear effect.

`at(varname = # [[varname = #] [...]])` specifies values to use in computing the predicted value. Here *varname* is one of the independent variables, *indepvars*, or the conditioned variables, *condvars*(). The default is to use the median of each independent and conditioned variable.

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see [\[U\] 20.7 Specifying the width of confidence intervals](#).

`memory(#[b|k|m|g])` sets a limit on the amount of memory `estat predict` can use when generating the conditional distribution of the constant parameter sufficient statistic. The default is `memory(10m)`, where *m* stands for megabyte, or 1,048,576 bytes. The following are also available: *b* stands for byte; *k* stands for kilobyte, which is equal to 1,024 bytes; and *g* stands for gigabyte, which is equal to 1,024 megabytes. The minimum setting allowed is `1m` and the maximum is `512m` or 0.5g, but do not attempt to use more memory than is available on your computer. Also see *Remarks* in [\[R\] exlogistic](#) for details on enumerating the conditional distribution.

`nolog` prevents the display of the enumeration log. By default, the enumeration log is displayed showing the progress of enumerating the distribution of the observed successes conditioned on the independent variables shifted by the values specified in `at()` (or by their medians). See [Methods and formulas](#) in [\[R\] exlogistic](#) for details of the computations.

## Syntax for estat se

```
estat se [, coef]
```

## Menu

Statistics > Postestimation > Reports and statistics

## Option for estat se

`coef` requests that the estimated coefficients and their asymptotic standard errors be reported. The default is to report the odds ratios and their asymptotic standard errors.

## Remarks

Predictions must be done using the estimation sample. This is because the prediction is really an estimated constant coefficient (the intercept) after shifting the independent variables and conditioned variables by the values specified in `at()` or by their medians. The justification for this approach can be seen by rewriting the model as

$$\log \left( \frac{\pi_i}{1 - \pi_i} \right) = (\alpha + \mathbf{x}_0\beta) + (\mathbf{x}_i - \mathbf{x}_0)\beta$$

where  $\mathbf{x}_0$  are the specified values for the *indepvars* (Metha and Patel 1995). Because the estimation of the constant term is required, this technique is not appropriate for stratified models that used the `group()` option.

## ► Example 1

To demonstrate, we return to the [example 2](#) in [\[R\] exlogistic](#) using data from a prospective study of perinatal infection and HIV-1. Here there was an investigation into whether the blood serum levels of CD4 and CD8 measured in infants at 6 months of age might predict their development of HIV infection. The blood serum levels are coded as ordinal values 0, 1, and 2. These data are used by [Mehta and Patel \(1995\)](#) as an exposition of exact logistic.

```
. use http://www.stata-press.com/data/r11/hiv_n
(prospective study of perinatal infection of HIV-1; binomial form)
. gen byte cd4_0 = (cd4==0)
. gen byte cd4_1 = (cd4==1)
. gen byte cd8_0 = (cd8==0)
. gen byte cd8_1 = (cd8==1)
. exlogistic hiv cd4_0 cd4_1 cd8_0 cd8_1, terms(cd4=cd4_0 cd4_1,
> cd8=cd8_0 cd8_1) binomial(n) test(probability) saving(dist, replace)
(output omitted)
. estat predict
```

Enumerating sample-space combinations:

```
observation 1: enumerations = 3
observation 2: enumerations = 12
observation 3: enumerations = 5
observation 4: enumerations = 5
observation 5: enumerations = 5
observation 6: enumerations = 35
observation 7: enumerations = 15
observation 8: enumerations = 15
observation 9: enumerations = 9
observation 10: enumerations = 9
observation 11: enumerations = 5
observation 12: enumerations = 18
note: CMLE estimate for _cons is -inf; computing MUE
```

Predicted value at cd4\_0 = 0, cd4\_1 = 0, cd8\_0 = 0, cd8\_1 = 1

| hiv         | Predicted | Std. Err. | [95% Conf. Interval] |        |
|-------------|-----------|-----------|----------------------|--------|
| Probability | 0.0390*   | N/A       | 0.0000               | 0.1962 |

(\*) identifies median unbiased estimates (MUE); because an MUE is computed, there is no SE estimate

Because we did not specify values by using the `at()` option, the median values of the *indepvars* are used for the prediction. By default, medians are used instead of means because we want to use values that are observed in the dataset. If the means of the binary variables `cd4_0`–`cd8_1` were used, we would have created floating point variables in  $(0, 1)$  that not only do not properly represent the indicator variables but also would be a source of computational inefficiency in generating the conditional distribution. Because the MUE is computed for the predicted value, there is no standard-error estimate.

From the example discussions in [\[R\] exlogistic](#), the infants at highest risk are those with a CD4 level of 0 and a CD8 level of 2. Below we use the `at()` option to make a prediction at these blood serum levels.

```
. estat predict, at(cd4_0=1 cd4_1=0 cd8_0=0 cd8_1=0) nolog
note: CMLE estimate for _cons is +inf; computing MUE
Predicted value at cd4_0 = 1, cd4_1 = 0, cd8_0 = 0, cd8_1 = 0
```

| hiv         | Predicted | Std. Err. | [95% Conf. Interval] |        |
|-------------|-----------|-----------|----------------------|--------|
| Probability | 0.9063*   | N/A       | 0.4637               | 1.0000 |

(\*) identifies median unbiased estimates (MUE); because an MUE is computed, there is no SE estimate



## Saved results

estat predict saves the following in `r()`:

- Scalars
- `r(imue)`

1 if `r(pred)` is an MUE and 0 if a CMLE
- `r(pred)`

estimated probability or the linear effect
- `r(se)`

asymptotic standard error of `r(pred)`
- Macros
- `r(estimate)`

prediction type: `pr` or `xb`
- `r(level)`

confidence level
- Matrices
- `r(ci)`

confidence interval
- `r(x)`

`indepvars` and `condvars()` values

## Methods and formulas

All postestimation commands listed above are implemented as ado-files using Mata.

## Reference

Mehta, C. R., and N. R. Patel. 1995. Exact logistic regression: Theory and examples. *Statistics in Medicine* 14: 2143–2160.

## Also see

- [\[R\] exlogistic](#) — Exact logistic regression
- [\[U\] 20 Estimation and postestimation commands](#)

Syntax

```
expoisson devar indepvars [if] [in] [weight] [, options]
```

| <i>options</i>                                                  | description                                                                                                      |
|-----------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|
| Model                                                           |                                                                                                                  |
| <u>condvars</u> ( <i>varlist</i> )                              | condition on variables in <i>varlist</i>                                                                         |
| <u>group</u> ( <i>varname</i> )                                 | groups/strata are stratified by unique values of <i>varname</i>                                                  |
| <u>exposure</u> ( <i>varname<sub>e</sub></i> )                  | include ln( <i>varname<sub>e</sub></i> ) in model with coefficient constrained to 1                              |
| <u>offset</u> ( <i>varname<sub>o</sub></i> )                    | include <i>varname<sub>o</sub></i> in model with coefficient constrained to 1                                    |
| Options                                                         |                                                                                                                  |
| <u>memory</u> (# [ <i>b</i>   <i>k</i>   <i>m</i>   <i>g</i> ]) | set limit on memory usage; default is memory(25m)                                                                |
| <u>saving</u> ( <i>filename</i> )                               | save the joint conditional distribution to <i>filename</i>                                                       |
| Reporting                                                       |                                                                                                                  |
| <u>level</u> (#)                                                | set confidence level; default is level(95)                                                                       |
| <u>irr</u>                                                      | report incidence-rate ratios                                                                                     |
| <u>test</u> ( <i>testopt</i> )                                  | report significance of observed sufficient statistic, conditional scores test, or conditional probabilities test |
| <u>mue</u> ( <i>varlist</i> )                                   | compute the median unbiased estimates for <i>varlist</i>                                                         |
| <u>midp</u>                                                     | use the mid- <i>p</i> -value rule                                                                                |
| <u>nolog</u>                                                    | do not display the enumeration log                                                                               |

by, statsby, and xi are allowed; see [U] 11.1.10 Prefix commands.  
fweights are allowed; see [U] 11.1.6 weight.  
See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Exact statistics > Exact Poisson regression

Description

expoisson fits an exact Poisson regression model of *devar* on *indepvars*. Exact Poisson regression is an alternative to standard maximum-likelihood-based Poisson regression (see [R] poisson) that offers more accurate inference in small samples because it does not depend on asymptotic results. For stratified data, expoission is an alternative to fixed-effects Poisson regression (see xtpoisson, fe in [XT] xtpoisson); like fixed-effects Poisson regression, exact Poisson regression conditions on the number of events in each stratum.

Exact Poisson regression is computationally intensive, so if you have regressors whose parameter estimates are not of interest (i.e., nuisance parameters), you should specify those variables in the condvars() option instead of in indepvars.

## Options

### Model

`condvars(varlist)` specifies variables whose parameter estimates are not of interest to you. You can save substantial computer time and memory by moving such variables from *indepvars* to `condvars()`. Understand that you will get the same results for *x1* and *x3* whether you type

```
. expoissn y x1 x2 x3 x4
```

or

```
. expoissn y x1 x3, condvars(x2 x4)
```

`group(varname)` specifies the variable defining the strata, if any. A constant term is assumed for each stratum identified in *varname*, and the sufficient statistics for *indepvars* are conditioned on the observed number of successes within each group (as well as other variables in the model). The group variable must be integer valued.

`exposure(varnamee), offset(varnameo)`; see [R] [estimation options](#).

### Options

`memory(#[b|k|m|g])` sets a limit on the amount of memory `expoissn` can use when computing the conditional distribution of the parameter sufficient statistics. The default is `memory(25m)`, where *m* stands for megabyte, or 1,048,576 bytes. The following are also available: *b* stands for byte; *k* stands for kilobyte, which is equal to 1,024 bytes; and *g* stands for gigabyte, which is equal to 1,024 megabytes. The minimum setting allowed is 1m and the maximum is 2048m or 2g, but do not attempt to use more memory than is available on your computer.

`saving(filename [, replace])` saves the joint conditional distribution for each independent variable specified in *indepvars*. There is one file for each variable, and it is named using the prefix *filename* with the variable name appended. For example, `saving(mydata)` with an independent variable named *X* would generate a data file named `mydata_X.dta`. Use `replace` to replace an existing file. Each file contains the conditional distribution for one of the independent variables specified in *indepvars* conditioned on all other *indepvars* and those variables specified in `condvars()`. There are two variables in each data file: the feasible sufficient statistics for the variable's parameter and their associated weights. The weights variable is named `_w_`.

### Reporting

`level(#)`; see [R] [estimation options](#). The `level(#)` option will not work on replay because confidence intervals are based on estimator-specific enumerations. To change the confidence level, you must refit the model.

`irr` reports estimated coefficients transformed to incidence-rate ratios, that is,  $\exp(\beta)$  rather than  $\beta$ . Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated or stored. `irr` may be specified at estimation or when replaying previously estimated results.

`test(sufficient|score|probability)` reports the significance level of the observed sufficient statistic, the conditional scores test, or the conditional probabilities test. The default is `test(sufficient)`. All the statistics are computed at estimation time, and each statistic may be displayed postestimation; see [R] [expoissn postestimation](#).

`mue(varlist)` specifies that median unbiased estimates (MUEs) be reported for the variables in *varlist*. By default, the conditional maximum likelihood estimates (CMLEs) are reported, except for those parameters for which the CMLEs are infinite. Specify `mue(_all)` if you want MUEs for all the *indepvars*.



`midp` instructs `expoisson` to use the mid- $p$ -value rule when computing the MUEs, significance levels, and confidence intervals. This adjustment is for the discreteness of the distribution by halving the value of the discrete probability of the observed statistic before adding it to the  $p$ -value. The mid- $p$ -value rule cannot be MUEs whose corresponding parameter CMLE is infinite.

`nolog` prevents the display of the enumeration log. By default, the enumeration log is displayed, showing the progress of computing the conditional distribution of the sufficient statistics.

## Remarks

Exact Poisson regression estimates the model parameters by using the conditional distributions of the parameters' sufficient statistics, and the resulting parameter estimates are known as CMLEs. Exact Poisson regression is a small-sample alternative to the maximum-likelihood ML Poisson model. See [R] [poisson](#) and [XT] [xtpoisson](#) to obtain maximum likelihood estimates (MLEs) for the Poisson model and the fixed-effects Poisson model.

Let  $Y_i$  denote a Poisson random variable where we observe the outcome  $Y_i = y_i$ ,  $i = 1, \dots, n$ . Associated with each independent observation is a  $1 \times p$  vector of covariates,  $\mathbf{x}_i$ . We will denote  $\mu_i = E[Y_i | \mathbf{x}_i]$  and use the log linear model to model the relationship between  $Y_i$  and  $\mathbf{x}_i$ ,

$$\log(\mu_i) = \theta + \mathbf{x}_i\beta$$

where the constant term,  $\theta$ , and the  $p \times 1$  vector of regression parameters,  $\beta$ , are unknown. The probability of observing  $Y_i = y_i$ ,  $i = 1, \dots, n$ , is

$$\Pr(\mathbf{Y} = \mathbf{y}) = \prod_{i=1}^n \frac{\mu_i^{y_i} e^{-\mu_i}}{y_i!}$$

where  $\mathbf{Y} = (Y_1, \dots, Y_n)$  and  $\mathbf{y} = (y_1, \dots, y_n)$ . The MLEs for  $\theta$  and  $\beta$  maximize the log of this function.

The sufficient statistics for  $\theta$  and  $\beta_j$ ,  $j = 1, \dots, p$ , are  $M = \sum_{i=1}^n Y_i$  and  $T_j = \sum_{i=1}^n Y_i x_{ij}$ , respectively, and we observe  $M = m$  and  $T_j = t_j$ . `expoisson` tallies the conditional distribution for each  $T_j$ , given the other sufficient statistics  $T_l = t_l$ ,  $l \neq j$  and  $M = m$ . Denote one of these values to be  $t_j^{(k)}$ ,  $k = 1, \dots, N$ , with weight  $w_k$  that accounts for all the generated  $\mathbf{Y}$  vectors that give rise to  $t_j^{(k)}$ . The conditional probability of observing  $T_j = t_j$  has the form

$$\Pr(T_j = t_j \mid T_l = t_l, l \neq j, M = m) = \frac{w e^{t_j \beta_j}}{\sum_k w_k e^{t_j^{(k)} \beta_j}} \quad (1)$$

where the sum is over the subset of  $\mathbf{T}$  vectors such that  $(T_1^{(k)} = t_1, \dots, T_j^{(k)} = t_j^{(k)}, \dots, T_p^{(k)} = t_p)$  and  $w$  is the weight associated with the observed  $\mathbf{t}$ . The CMLE for  $\beta_j$  maximizes the log of this function.

Specifying nuisance variables in `condvars()` prevents `expoisson` from estimating their associated regression coefficients. These variables are still conditional variables when tallying the conditional distribution for the variables in `indepvars`.

Inferences from MLEs rely on asymptotics, and if your sample size is small, these inferences may not be valid. On the other hand, inferences from the CMLEs are exact in that they use the conditional distribution of the sufficient statistics outlined above.

For small datasets, the dependent variable can be completely determined by the data. Here the MLEs and the CMLEs are unbounded. When this occurs, `expoisson` will compute the MUE, the regression estimate that places the observed sufficient statistic at the median of the conditional distribution.

See [R] [xlogistic](#) for a more thorough discussion of exact estimation and related statistics.

➤ Example 1

[Armitage, Berry, and Matthews \(2002, 499–501\)](#) fit a log-linear model to data containing the number of cerebrovascular accidents experienced by 41 men during a fixed period, each of whom had recovered from a previous cerebrovascular accident and was hypertensive. Sixteen men received treatment, and in the original data, there are three age groups (40–49, 50–59,  $\geq 60$ ), but we pool the first two age groups to simplify the example. [Armitage, Berry, and Matthews](#) point out that this was not a controlled trial, but the data are useful to inquire whether there is evidence of fewer accidents for the treatment group and if age may be an important factor. The dependent variable `count` contains the number of accidents, variable `treat` is an indicator for the treatment group (1 = treatment, 0 = control), and variable `age` is an indicator for the age group (0 = 40–59; 1 =  $\geq 60$ ).

First, we load the data, list it, and tabulate the cerebrovascular accident counts by treatment and age group.

```
. use http://www.stata-press.com/data/r11/cerebacc
(cerebrovascular accidents in hypotensive-treated and control groups)
. list
```

|                  | treat     | count | age   |
|------------------|-----------|-------|-------|
| 1.               | control   | 0     | 40/59 |
| 2.               | control   | 0     | >=60  |
| 3.               | control   | 1     | 40/59 |
| 4.               | control   | 1     | >=60  |
| 5.               | control   | 2     | 40/59 |
| (output omitted) |           |       |       |
| 35.              | treatment | 0     | 40/59 |
| 36.              | treatment | 0     | 40/59 |
| 37.              | treatment | 0     | 40/59 |
| 38.              | treatment | 0     | 40/59 |
| 39.              | treatment | 1     | 40/59 |
| 40.              | treatment | 1     | 40/59 |
| 41.              | treatment | 1     | 40/59 |

```
. tabulate treat age [fw=count]
```

| hypotensive<br>drug<br>treatment | age group |      | Total |
|----------------------------------|-----------|------|-------|
|                                  | 40/59     | >=60 |       |
| control                          | 15        | 10   | 25    |
| treatment                        | 4         | 0    | 4     |
| Total                            | 19        | 10   | 29    |

Next we estimate the CMLE with `expoisson` and, for comparison, the MLE with `poisson`.

```
. expoisscount treat age
Estimating: treat
Enumerating sample-space combinations:
observation 1: enumerations = 11
observation 2: enumerations = 11
observation 3: enumerations = 11
(output omitted)
observation 39: enumerations = 410
observation 40: enumerations = 410
observation 41: enumerations = 30
Estimating: age
Enumerating sample-space combinations:
observation 1: enumerations = 5
observation 2: enumerations = 15
observation 3: enumerations = 15
(output omitted)
observation 39: enumerations = 455
observation 40: enumerations = 455
observation 41: enumerations = 30
Exact Poisson regression
```

Number of obs = 41

| count | Coef.     | Suff. | 2*Pr(Suff.) | [95% Conf. Interval] |           |
|-------|-----------|-------|-------------|----------------------|-----------|
| treat | -1.594306 | 4     | 0.0026      | -3.005089            | -.4701708 |
| age   | -.5112067 | 10    | 0.2794      | -1.416179            | .3429232  |

```
. poissoncount treat age, nolog
```

Poisson regression

Number of obs = 41

LR chi2(2) = 10.64

Prob > chi2 = 0.0049

Pseudo R2 = 0.1201

Log likelihood = -38.97981

| count | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-------|-----------|-----------|-------|-------|----------------------|-----------|
| treat | -1.594306 | .5573614  | -2.86 | 0.004 | -2.686714            | -.5018975 |
| age   | -.5112067 | .4043525  | -1.26 | 0.206 | -1.303723            | .2813096  |
| _cons | .233344   | .2556594  | 0.91  | 0.361 | -.2677391            | .7344271  |

`expoisscount` generates an enumeration log for each independent variable in *indepvars*. The conditional distribution of the parameter sufficient statistic is tallied for each independent variable. The conditional distribution for `treat`, for example, has 30 records containing the weights,  $w_k$ , and feasible sufficient statistics,  $t_{\text{treat}}^{(k)}$ . In essence, the set of points  $(w_k, t_{\text{treat}}^{(k)})$ ,  $k = 1, \dots, 30$ , tallied by `expoisscount` now become the data to estimate the regression coefficient for `treat`, using (1) as the likelihood. Remember that one of the 30  $(w_k, t_{\text{treat}}^{(k)})$  must contain the observed sufficient statistic,  $t_{\text{treat}} = \sum_{i=1}^{41} \text{treat}_i \times \text{count}_i = 4$ , and its relative position in the sorted set of points (sorted by  $t_{\text{treat}}^{(k)}$ ) is how the sufficient-statistic significance is computed. This algorithm is repeated for the `age` variable.

The regression coefficients for `treat` and `age` are numerically identical for both Poisson models. Both models indicate that the treatment is significant at reducing the rate of cerebrovascular accidents,  $\approx e^{-1.59} \approx 0.204$ , or a reduction of about 80%. There is no significant age effect.

The  $p$ -value for the treatment regression-coefficient sufficient statistic indicates that the treatment effect is a bit more significant than for the corresponding asymptotic  $Z$  statistic from `poisson`. However, the exact confidence intervals are wider than their asymptotic counterparts.

➤ Example 2

Agresti (2002) used the data from Laird and Olivier (1981) to demonstrate the Poisson model for modeling rates. The data consist of patient survival after heart valve replacement operations. The sample consists of 109 patients that are classified by type of heart valve (aortic, mitral) and by age ( $<55$ ,  $\geq 55$ ). Follow-up observations cover lengths from 3 to 97 months, and the time at risk, or exposure, is stored in the variable TAR. The response is whether the subject died. First, we take a look at the data and then estimate the incidence rates (IRs) with `expoisson` and `poisson`.

```
. use http://www.stata-press.com/data/r11/heartvalve
(heart valve replacement data)

. list
```

|    | age   | valve  | deaths | TAR  |
|----|-------|--------|--------|------|
| 1. | < 55  | aortic | 4      | 1259 |
| 2. | < 55  | mitral | 1      | 2082 |
| 3. | >= 55 | aortic | 7      | 1417 |
| 4. | >= 55 | mitral | 9      | 1647 |

The `age` variable is coded 0 for age  $<55$  and 1 for age  $\geq 55$ , and the `valve` variable is coded 0 for the aortic valve and 1 for the mitral valve. The total number of deaths,  $M = 21$ , is small enough that enumerating the conditional distributions for age and valve type is feasible and asymptotic inferences associated with standard ML Poisson regression may be questionable.

```
. expoisson deaths age valve, exposure(TAR) irr
```

```
Estimating: age
Enumerating sample-space combinations:
observation 1: enumerations = 11
observation 2: enumerations = 11
observation 3: enumerations = 132
observation 4: enumerations = 22
```

```
Estimating: valve
Enumerating sample-space combinations:
observation 1: enumerations = 17
observation 2: enumerations = 17
observation 3: enumerations = 102
observation 4: enumerations = 22
```

```
Exact Poisson regression
```

| Number of obs = 4 |            |       |             |                      |
|-------------------|------------|-------|-------------|----------------------|
| deaths            | IRR        | Suff. | 2*Pr(Suff.) | [95% Conf. Interval] |
| age               | 3.390401   | 16    | 0.0194      | 1.182297 11.86935    |
| valve             | .7190197   | 10    | 0.5889      | .2729881 1.870068    |
| TAR               | (exposure) |       |             |                      |

```
. poisson deaths age valve, exposure(TAR) irr nolog
Poisson regression
Log likelihood = -8.1747285
Number of obs = 4
LR chi2(2) = 7.62
Prob > chi2 = 0.0222
Pseudo R2 = 0.3178
```

| deaths | IRR        | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|--------|------------|-----------|-------|-------|----------------------|----------|
| age    | 3.390401   | 1.741967  | 2.38  | 0.017 | 1.238537             | 9.280965 |
| valve  | .7190197   | .3150492  | -0.75 | 0.452 | .3046311             | 1.6971   |
| TAR    | (exposure) |           |       |       |                      |          |

The CMLE and the MLE are numerically identical. The death rate for the older age group is about 3.4 times higher than the younger age group, and this difference is significant at the 5% level. This means that for every death in the younger group each month, we would expect about three deaths in the older group. The IR estimate for valve type is approximately 0.72, but it is not significantly different from one. The exact Poisson confidence intervals are a bit wider than the asymptotic CIs.

You can use `ir` (see [\[ST\] epitab](#)) to estimate IRs and exact CIs for one covariate, and we compare these CIs with those from `expoisson`, where we estimate the incidence rate by using age only.

```
. ir deaths age TAR
```

|                  | age of patient |               |                           |
|------------------|----------------|---------------|---------------------------|
|                  | Exposed        | Unexposed     | Total                     |
| number of deaths | 16             | 5             | 21                        |
| time at risk     | 3064           | 3341          | 6405                      |
| Incidence rate   | .0052219       | .0014966      | .0032787                  |
|                  | Point estimate |               | [95% Conf. Interval]      |
| Inc. rate diff.  | .0037254       |               | .00085 .0066007           |
| Inc. rate ratio  | 3.489295       |               | 1.221441 12.17875 (exact) |
| Attr. frac. ex.  | .7134092       |               | .1812948 .9178898 (exact) |
| Attr. frac. pop  | .5435498       |               |                           |
|                  | (midp)         | Pr(k>=16) =   | 0.0049 (exact)            |
|                  | (midp)         | 2*Pr(k>=16) = | 0.0099 (exact)            |

```
. expoisson deaths age, exposure(TAR) irr midp nolog
Exact Poisson regression
Number of obs = 4
```

| deaths | IRR        | Suff. | 2*Pr(Suff.) | [95% Conf. Interval] |          |
|--------|------------|-------|-------------|----------------------|----------|
| age    | 3.489295   | 16    | 0.0099      | 1.324926             | 10.64922 |
| TAR    | (exposure) |       |             |                      |          |

mid-p-value computed for the probabilities and CIs

Both `ir` and `expoisson` give identical IRs and  $p$ -values. Both report the two-sided exact significance by using the mid- $p$ -value rule that accounts for the discreteness in the distribution by subtracting  $p_{1/2} = \Pr(T = t)/2$  from  $p_l = \Pr(T \leq t)$  and  $p_g = \Pr(T \geq t)$ , computing  $2 \times \min(p_l - p_{1/2}, p_g - p_{1/2})$ . By default, `expoisson` will not use the mid- $p$ -value rule (when you exclude the `midp` option), and here the two-sided exact significance would be  $2 \times \min(p_l, p_g) = 0.0158$ . The confidence intervals differ because `expoisson` uses the mid- $p$ -value rule when computing the confidence intervals, yet

`ir` does not. You can verify this by executing `expoisson` without the `midp` option for this example; you will get the same CIs as `ir`.

You can replay `expoisson` to view the conditional scores test or the conditional probabilities test by using the `test()` option.

```
. expoisson, test(score) irr
Exact Poisson regression
```

|            |                        |         |           |                      | Number of obs = | 4 |
|------------|------------------------|---------|-----------|----------------------|-----------------|---|
| deaths     | IRR                    | Score   | Pr>=Score | [95% Conf. Interval] |                 |   |
| age<br>TAR | 3.489295<br>(exposure) | 6.76528 | 0.0113    | 1.324926             | 10.64922        |   |

mid-p-value computed for the probabilities and CIs

All the statistics for `expoisson` are defined in [Methods and formulas](#) of [R] `exlogistic`. Apart from enumerating the conditional distributions for the logistic and Poisson sufficient statistics, computationally, the primary difference between `exlogistic` and `expoisson` is the weighting values in the likelihood for the parameter sufficient statistics.

➤ Example 3

In this example, we fabricate data that will demonstrate the difference between the CMLE and the MUE when the CMLE is not infinite. A difference in these estimates will be more pronounced when the probability of the coefficient sufficient statistic is skewed when plotted as a function of the regression coefficient.

```
. clear
. input y x
 y x
1. 0 2
2. 1 1
3. 1 0
4. 0 0
5. 0 .5
6. 1 .5
7. 2 .01
8. 3 .001
9. 4 .0001
10. end

. expoisson y x, test(score)
Enumerating sample-space combinations:
observation 1: enumerations = 13
observation 2: enumerations = 91
observation 3: enumerations = 169
observation 4: enumerations = 169
observation 5: enumerations = 313
observation 6: enumerations = 313
observation 7: enumerations = 1469
observation 8: enumerations = 5525
observation 9: enumerations = 5479

Exact Poisson regression
```

|   |           |          |           |                      | Number of obs = | 9 |
|---|-----------|----------|-----------|----------------------|-----------------|---|
| y | Coef.     | Score    | Pr>=Score | [95% Conf. Interval] |                 |   |
| x | -1.534468 | 2.955316 | 0.0810    | -3.761718            | .0485548        |   |

```
. expoiss y x, test(score) mue(x) nolog
```

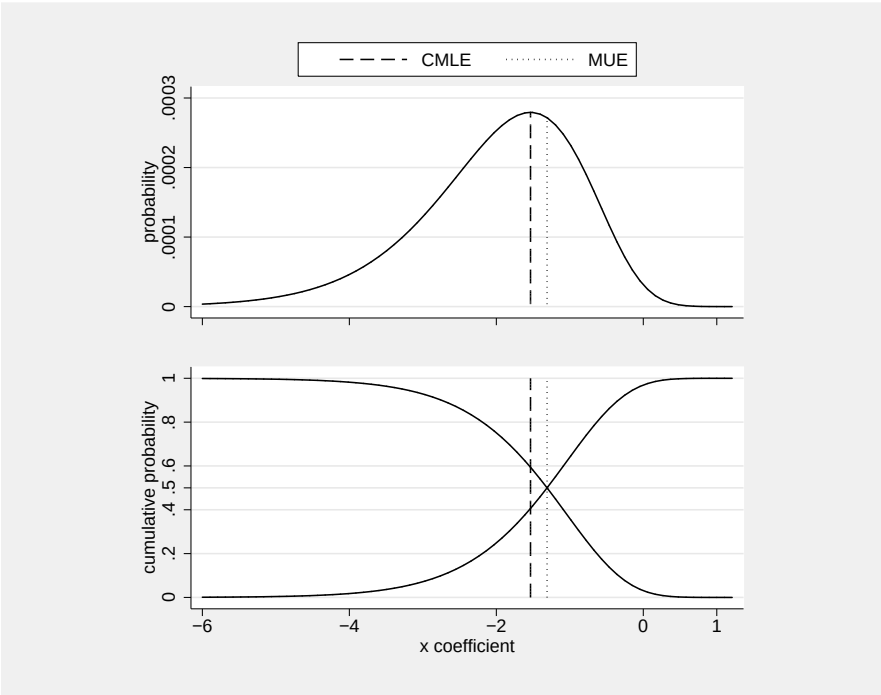
Exact Poisson regression

| Number of obs = 9 |            |          |           |                      |
|-------------------|------------|----------|-----------|----------------------|
| y                 | Coef.      | Score    | Pr>=Score | [95% Conf. Interval] |
| x                 | -1.309268* | 2.955316 | 0.0810    | -3.761718 .0485548   |

(\*) median unbiased estimates (MUE)

We observe  $(x_i, y_i)$ ,  $i = 1, \dots, 9$ . If we condition on  $m = \sum_{i=1}^9 y_i = 12$ , the conditional distribution of  $T_x = \sum_i Y_i x_i$  has a size of 5,479 elements. For each entry in this enumeration, a realization of  $Y_i = y_i^{(k)}$ ,  $k = 1, \dots, 5,479$ , is generated such that  $\sum_i y_i^{(k)} = 12$ . One of these realizations produces the observed  $t_x = \sum_i y_i x_i \approx 1.5234$ .

Below is a graphical display comparing the CMLE with the MUE. We plot  $\Pr(T_x = t_x \mid M = 12, \beta_x)$  versus  $\beta_x$ ,  $-6 \leq \beta_x \leq 1$ , in the upper panel and the cumulative probabilities,  $\Pr(T_x \leq t_x \mid M = 12, \beta_x)$  and  $\Pr(T_x \geq t_x \mid M = 12, \beta_x)$ , in the lower panel.



The location of the CMLE, indicated by the dashed line, is at the mode of the probability profile, and the MUE, indicated by the dotted line, is to the right of the mode. If we solve for the  $\beta_x^{(u)}$  and  $\beta_x^{(l)}$  such that  $\Pr(T_x \leq t_x \mid M = 12, \beta_x^{(u)}) = 1/2$  and  $\Pr(T_x \geq t_x \mid M = 12, \beta_x^{(l)}) = 1/2$ , the MUE is  $(\beta_x^{(u)} + \beta_x^{(l)})/2$ . As you can see in the lower panel, the MUE cuts through the intersection of these cumulative probability profiles.

## □ Technical note

The `memory(#)` option limits the amount of memory that `expoisson` will consume when computing the conditional distribution of the parameter sufficient statistics. `memory()` is independent of the system setting `c(memory)` (see `set memory` in [D] [memory](#)), and `expoisson` can exceed the memory limit specified in `c(memory)` without terminating. By default, a log is provided that displays the number of enumerations (the size of the conditional distribution) after processing each observation. Typically, you will see the number of enumerations increase, and then at some point they will decrease as the multivariate shift algorithm (Hirji, Mehta, and Patel 1987) determines that some of the enumerations cannot achieve the observed sufficient statistics of the conditioning variables. When the algorithm is complete, however, it is necessary to store the conditional distribution of the parameter sufficient statistics as a dataset. It is possible, therefore, to get a memory error when the algorithm has completed and `c(memory)` is not large enough to store the conditional distribution. □

## □ Technical note

Computing the conditional distributions and reported statistics requires data sorting and numerical comparisons. If there is at least one single-precision variable specified in the model, `expoisson` will make comparisons with a relative precision of  $2^{-5}$ . Otherwise, a relative precision of  $2^{-11}$  is used. Be careful if you use `recast` to promote a single-precision variable to double precision (see [D] [recast](#)). You might try listing the data in full precision (maybe `%20.15g`; see [D] [format](#)) to make sure that this is really what you want. See [D] [data types](#) for information on precision of numeric storage types. □



## Saved results

expoisson saves the following in `e()`:

### Scalars

|                                 |                                                                                          |
|---------------------------------|------------------------------------------------------------------------------------------|
| <code>e(N)</code>               | number of observations                                                                   |
| <code>e(k_groups)</code>        | number of groups                                                                         |
| <code>e(relative_weight)</code> | relative weight for the observed <code>e(sufficient)</code> and <code>e(condvars)</code> |
| <code>e(sum_y)</code>           | sum of <i>depvar</i>                                                                     |
| <code>e(k_indvars)</code>       | number of independent variables                                                          |
| <code>e(k_condvars)</code>      | number of conditioning variables                                                         |
| <code>e(midp)</code>            | mid- <i>p</i> -value rule indicator                                                      |
| <code>e(eps)</code>             | relative difference tolerance                                                            |

### Macros

|                                   |                                           |
|-----------------------------------|-------------------------------------------|
| <code>e(cmd)</code>               | expoisson                                 |
| <code>e(cmdline)</code>           | command as typed                          |
| <code>e(title)</code>             | Exact Poisson regression                  |
| <code>e(depvar)</code>            | dependent variable                        |
| <code>e(indvars)</code>           | independent variables                     |
| <code>e(condvars)</code>          | conditional variables                     |
| <code>e(groupvar)</code>          | group variable                            |
| <code>e(exposure)</code>          | exposure variable                         |
| <code>e(level)</code>             | confidence level                          |
| <code>e(wtype)</code>             | weight type                               |
| <code>e(wexp)</code>              | weight expression                         |
| <code>e(datasignature)</code>     | the checksum                              |
| <code>e(datasignaturevars)</code> | variables used in calculation of checksum |
| <code>e(properties)</code>        | b V                                       |
| <code>e(estat_cmd)</code>         | program used to implement estat           |
| <code>e(predict)</code>           | program used to implement predict         |
| <code>e(marginsnotok)</code>      | predictions disallowed by margins         |

(Continued on next page)

**Matrices**

|                                |                                                                                 |
|--------------------------------|---------------------------------------------------------------------------------|
| <code>e(b)</code>              | coefficient vector                                                              |
| <code>e(mue_indicators)</code> | indicator for elements of <code>e(b)</code> estimated using MUE instead of CMLE |
| <code>e(se)</code>             | <code>e(b)</code> standard errors (CMLEs only)                                  |
| <code>e(ci)</code>             | matrix of <code>e(level)</code> confidence intervals for <code>e(b)</code>      |
| <code>e(sum_y_groups)</code>   | sum of <code>e(depvar)</code> for each group                                    |
| <code>e(N_g)</code>            | number of observations in each group                                            |
| <code>e(sufficient)</code>     | sufficient statistics for <code>e(b)</code>                                     |
| <code>e(p_sufficient)</code>   | $p$ -value for <code>e(sufficient)</code>                                       |
| <code>e(scoretest)</code>      | conditional scores tests for <i>indepvars</i>                                   |
| <code>e(p_scoretest)</code>    | $p$ -values for <code>e(scoretest)</code>                                       |
| <code>e(probtest)</code>       | conditional probability tests for <i>indepvars</i>                              |
| <code>e(p_probtest)</code>     | $p$ -value for <code>e(probtest)</code>                                         |

**Functions**

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`expoisson` is implemented as an ado-file.

Let  $\{Y_1, Y_2, \dots, Y_n\}$  be a set of  $n$  independent Poisson random variables. For each  $i = 1, \dots, n$ , we observe  $Y_i = y_i \geq 0$ , and associated with each observation is the covariate row vector of length  $p$ ,  $\mathbf{x}_i = (x_{i1}, \dots, x_{ip})$ . Denote  $\boldsymbol{\beta} = (\beta_1, \dots, \beta_p)^T$  to be the column vector of regression parameters and  $\theta$  to be the constant. The sufficient statistic for  $\beta_j$  is  $T_j = \sum_{i=1}^n Y_i x_{ij}$ ,  $j = 1, \dots, p$ , and for  $\theta$  is  $M = \sum_{i=1}^n Y_i$ . We observe  $T_j = t_j$ ,  $t_j = \sum_{i=1}^n y_i x_{ij}$ , and  $M = m$ ,  $m = \sum_{i=1}^n y_i$ . Let  $\kappa_i$  be the exposure for the  $i$ th observation. Then the probability of observing  $(Y_1 = y_1, Y_2 = y_2, \dots, Y_n = y_n)$  is

$$\Pr(Y_1 = y_1, \dots, Y_n = y_n \mid \boldsymbol{\beta}, \mathbf{X}, \boldsymbol{\kappa}) = \frac{\exp(m\theta + \mathbf{t}\boldsymbol{\beta})}{\exp\{\sum_{i=1}^n \kappa_i \exp(\theta + \mathbf{x}_i\boldsymbol{\beta})\}} \prod_{i=1}^n \frac{\kappa_i^{y_i}}{y_i!}$$

where  $\mathbf{t} = (t_1, \dots, t_p)$ ,  $\mathbf{X} = (\mathbf{x}_1^T, \dots, \mathbf{x}_n^T)^T$ , and  $\boldsymbol{\kappa} = (\kappa_1, \dots, \kappa_n)^T$ .

The joint distribution of the sufficient statistics  $(\mathbf{T}, M)$  is obtained by summing over all possible sequences  $Y_1 \geq 0, \dots, Y_n \geq 0$  such that  $\mathbf{T} = \mathbf{t}$  and  $M = m$ . This probability function is

$$\Pr(T_1 = t_1, \dots, T_p = t_p, M = m \mid \boldsymbol{\beta}, \mathbf{X}, \boldsymbol{\kappa}) = \frac{\exp(m\theta + \mathbf{t}\boldsymbol{\beta})}{\exp\{\sum_{i=1}^n \kappa_i \exp(\theta + \mathbf{x}_i\boldsymbol{\beta})\}} \left( \sum_{\mathbf{u}} \prod_{i=1}^n \frac{\kappa_i^{u_i}}{u_i!} \right)$$

where the sum  $\sum_{\mathbf{u}}$  is over all nonnegative vectors  $\mathbf{u}$  of length  $n$  such that  $\sum_{i=1}^n u_i = m$  and  $\sum_{i=1}^n u_i \mathbf{x}_i = \mathbf{t}$ .

## Conditional distribution

Without loss of generality, we will restrict our discussion to the conditional distribution of the sufficient statistic for  $\beta_1$ ,  $T_1$ . If we condition on observing  $M = m$  and  $T_2 = t_2, \dots, T_p = t_p$ , the probability function of  $(T_1 \mid \beta_1, T_2 = t_2, \dots, T_p = t_p, M = m)$  is

$$\Pr(T_1 = t_1 \mid \beta_1, T_2 = t_2, \dots, T_p = t_p, M = m) = \frac{\left( \sum_{\mathbf{u}} \prod_{i=1}^n \frac{\kappa_i^{u_i}}{u_i!} \right) e^{t_1 \beta_1}}{\sum_{\mathbf{v}} \left( \prod_{i=1}^n \frac{\kappa_i^{v_i}}{v_i!} \right) e^{\beta_1 \sum_i v_i x_{i1}}} \quad (2)$$

where the sum  $\sum_{\mathbf{u}}$  is over all nonnegative vectors  $\mathbf{u}$  of length  $n$  such that  $\sum_{i=1}^n u_i = m$  and  $\sum_{i=1}^n u_i \mathbf{x}_i = \mathbf{t}$ , and the sum  $\sum_{\mathbf{v}}$  is over all nonnegative vectors  $\mathbf{v}$  of length  $n$  such that  $\sum_{i=1}^n v_i = m$ ,  $\sum_{i=1}^n v_i x_{i2} = t_2, \dots, \sum_{i=1}^n v_i x_{ip} = t_p$ . The CMLE for  $\beta_1$  is the value that maximizes the log of (1). This optimization task is carried out by `ml` (see [R] `ml`), using the conditional distribution of  $(T_1 \mid T_2 = t_2, \dots, T_p = t_p, M = m)$  as a dataset. This dataset consists of the feasible values and weights for  $T_1$ ,

$$\left\{ \left( s_1, \prod_{i=1}^n \frac{\kappa_i^{v_i}}{v_i!} \right) : \sum_{i=1}^n v_i = m, \sum_{i=1}^n v_i x_{i1} = s_1, \sum_{i=1}^n v_i x_{i2} = t_2, \dots, \sum_{i=1}^n v_i x_{ip} = t_p \right\}$$

Computing the CMLE, MUE, confidence intervals, conditional hypothesis tests, and sufficient statistic  $p$ -values is discussed in *Methods and formulas* of [R] `exlogistic`. The only difference between the two techniques is the use of the weights; that is, the weights for exact logistic are the combinatorial coefficients,  $c(\mathbf{t}, m)$ , in (1) of *Methods and formulas* in [R] `exlogistic`. `expoisson` and `exlogistic` use the same `ml` likelihood evaluator to compute the CMLEs as well as the same ado-programs and Mata functions to compute the MUEs and estimate statistics.

## References

- Agresti, A. 2002. *Categorical Data Analysis*. 2nd ed. Hoboken, NJ: Wiley.
- Armitage, P., G. Berry, and J. N. S. Matthews. 2002. *Statistical Methods in Medical Research*. 4th ed. Oxford: Blackwell.
- Cox, D. R., and E. J. Snell. 1989. *Analysis of Binary Data*. 2nd ed. London: Chapman & Hall.
- Hirji, K. F., C. R. Mehta, and N. R. Patel. 1987. Computing distributions for exact logistic regression. *Journal of the American Statistical Association* 82: 1110–1117.
- Laird, N. M., and D. Olivier. 1981. Covariance analysis of censored survival data using log-linear analysis techniques. *Journal of the American Statistical Association* 76: 231–240.

## Also see

- [R] `expoisson postestimation` — Postestimation tools for `expoisson`
- [R] `poisson` — Poisson regression
- [XT] `xtpoisson` — Fixed-effects, random-effects, and population-averaged Poisson models
- [U] **20 Estimation and postestimation commands**

# Title

## Description

The following postestimation command is of special interest after `expoisson`:

| command               | description                                                 |
|-----------------------|-------------------------------------------------------------|
| <code>estat se</code> | report odds ratio or coefficient asymptotic standard errors |

For information about this command, see below.

The following standard postestimation command is also available:

| command                      | description               |
|------------------------------|---------------------------|
| <code>estat summarize</code> | estimation sample summary |

## Special-interest postestimation command

`estat se` reports regression coefficients or incidence-rate asymptotic standard errors. The estimates are stored in the matrix `r(estimates)`.

## Syntax for estat se

```
estat se [, irr]
```

## Menu

Statistics > Postestimation > Reports and statistics

## Option for estat se

`irr` reports estimated coefficients transformed to incidence-rate ratios, that is,  $\exp(\beta)$  rather than  $\beta$ . Standard errors and confidence intervals are similarly transformed. The default is to report the regression coefficients and their asymptotic standard errors.

## Remarks

### ► Example 1

To demonstrate estat se after expoission, we use the British physicians smoking data.

```
. use http://www.stata-press.com/data/r11/smokes
(cigarette smoking and lung cancer among British physicians (45-49 years))
. expoission cases smokes, exposure(peryrs) irr nolog
```

Exact Poisson regression

Number of obs = 7

| cases            | IRR                    | Suff. | 2*Pr(Suff.) | [95% Conf. Interval] |          |
|------------------|------------------------|-------|-------------|----------------------|----------|
| smokes<br>peryrs | 1.077718<br>(exposure) | 797.4 | 0.0000      | 1.04552              | 1.111866 |

```
. estat se, irr
```

| cases  | IRR      | Std. Err. |
|--------|----------|-----------|
| smokes | 1.077718 | .0168547  |

◀

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [expoission](#) — Exact Poisson regression

[U] [20 Estimation and postestimation commands](#)

Syntax

Fractional polynomial regression

```
fracpoly [, fracpoly_options] : regression_cmd [yvar1 [yvar2]]
 xvar1 [# [#...]] [xvar2 [# [#...]]] [...] [xvarlist] [if] [in] [weight]
 [, regression_cmd_options]
```

Display table showing the best fractional polynomial model for each degree

```
fracpoly, compare
```

Create variables containing fractional polynomial powers

```
fracgen varname # [# ...] [if] [in] [, fracgen_options]
```

| fracpoly_options          | description                                                               |
|---------------------------|---------------------------------------------------------------------------|
| Model                     |                                                                           |
| <u>degree</u> (#)         | degree of fractional polynomial to fit; default is <code>degree(2)</code> |
| Model 2                   |                                                                           |
| <u>noscaling</u>          | suppress scaling of first independent variable                            |
| <u>noconstant</u>         | suppress constant term                                                    |
| <u>powers</u> (numlist)   | list of fractional polynomial powers from which models are chosen         |
| <u>center</u> (cent_list) | specification of centering for the independent variables                  |
| Reporting                 |                                                                           |
| <u>log</u>                | display iteration log                                                     |
| <u>compare</u>            | compare models by degree                                                  |
| <u>all</u>                | include out-of-sample observations in generated variables                 |

| regression_cmd_options        | description                                          |
|-------------------------------|------------------------------------------------------|
| Model 2                       |                                                      |
| <u>regression_cmd_options</u> | options appropriate to the regression command in use |

All weight types supported by `regression_cmd` are allowed; see [U] 11.1.6 **weight**.  
See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

where

*cent\_list* is a comma-separated list with elements *varlist*:{mean|#|no}, except that the first element may optionally be of the form {mean|#|no} to specify the default for all variables.

*regression\_cmd* may be clogit, glm, intreg, logistic, logit, mlogit, nbreg, ologit, oprobit, poisson, probit, qreg, regress, rreg, stcox, streg, or xtgee.

| <i>fracgen_options</i>                             | description                                               |
|----------------------------------------------------|-----------------------------------------------------------|
| Main                                               |                                                           |
| <b>center</b> (no   mean   #)                      | center <i>varname</i> as specified; default is center(no) |
| <b>noscaling</b>                                   | suppress scaling of <i>varname</i>                        |
| <b>restrict</b> ([ <i>varname</i> ] [ <i>if</i> ]) | compute centering and scaling using specified subsample   |
| <b>replace</b>                                     | replace variables if they exist                           |

Menu

fracpoly

Statistics > Linear models and related > Fractional polynomials > Fractional polynomial regression

fracgen

Statistics > Linear models and related > Fractional polynomials > Create fractional polynomial powers

Description

**fracpoly** fits fractional polynomials (FPs) in *xvar*<sub>1</sub> as part of the specified regression model. After execution, **fracpoly** leaves variables in the dataset named *lxvar*\_\_1, *lxvar*\_\_2, ..., where *xvar* represents the first four letters of the name of *xvar*<sub>1</sub>. The new variables contain the best-fitting FP powers of *xvar*<sub>1</sub>.

Covariates other than *xvar*<sub>1</sub>, which are optional, are specified in *xvar*<sub>2</sub>, ..., and *xvarlist*. They may be modeled linearly and with specified FP transformations. Fractional polynomial powers are specified by typing numbers after the variable's name. A variable name typed without numbers is entered linearly.

**fracgen** creates new variables named *varname*\_\_1, *varname*\_\_2, ..., containing FP powers of *varname* by using the powers (# [#...]) specified.

See [R] [fracpoly postestimation](#) for information on **fracplot** and **fracpred**.

See [R] [mfp](#) for multivariable FP model fitting.

Options for fracpoly

Model

---

**degree**(#) determines the degree of FP to be fit. The default is **degree**(2), i.e., a model with two power terms.

Model 2

---

**noscaling** suppresses scaling of *xvar*<sub>1</sub> and its powers.

`noconstant` suppresses the regression constant if this is permitted by *regression\_cmd*.

`powers(numlist)` is the set of FP powers from which models are to be chosen. The default is `powers(-2,-1,-.5,0,.5,1,2,3)` (0 means log).

`center(cent_list)` defines the centering for the covariates  $xvar_1, xvar_2, \dots, xvarlist$ . The default is `center(mean)`. A typical item in *cent\_list* is *varlist*:{`mean`|`#`|`no`}. Items are separated by commas. The first item is special because *varlist*: is optional, and if omitted, the default is (re)set to the specified value (`mean` or `#` or `no`). For example, `center(no, age:mean)` sets the default to `no` and sets the centering for `age` to `mean`.

*regression\_cmd\_options* are options appropriate to the regression command in use. For example, for `stcox`, *regression\_cmd\_options* may include `efron` or some alternate method for handling tied failures.

---

#### Reporting

`log` displays deviances and (for `regress`) residual standard deviations for each FP model fit.

`compare` reports a closed-test comparison between FP models.

`all` includes out-of-sample observations when generating the best-fitting FP powers of  $xvar_1, xvar_2$ , etc. By default, the generated FP variables contain missing values outside the estimation sample.

## Options for fracgen

---

#### Main

`center(no|mean|#)` specifies whether *varname* is to be centered; the default is `center(no)`.

`noscaling` suppresses scaling of *varname*.

`restrict([varname] [if])` specifies that centering and scaling be computed using the subsample identified by *varname* and *if*.

The subsample is defined by the observations for which  $varname \neq 0$  that also meet the *if* conditions. Typically,  $varname = 1$  defines the subsample and  $varname = 0$  indicates observations not belonging to the subsample. For observations whose subsample status is uncertain, *varname* should be set to a missing value; such observations are dropped from the subsample.

By default, `fracgen` computes the centering and scaling by using the sample of observations identified in the [*if*] [*in*] options. The `restrict()` option identifies a subset of this sample.

`replace` specifies that any existing variables named *varname\_1*, *varname\_2*, ... may be replaced.

## Remarks

Remarks are presented under the following headings:

[Introduction](#)

[fracpoly](#)

[Centering](#)

[Output with the compare option](#)

[fracgen](#)

[Models with several continuous covariates](#)

[Examples](#)



## Introduction

Regression models based on FP functions of a continuous covariate are described by [Royston and Altman \(1994b\)](#). Detailed examples using an earlier and rather more complex version of this set of commands are presented by [Royston and Altman \(1994a\)](#).

FPs increase the flexibility afforded by the family of conventional polynomial models. Although polynomials are popular in data analysis, linear and quadratic functions are severely limited in their range of curve shapes, whereas cubic and higher-order curves often produce undesirable artifacts, such as edge effects and waves.

A polynomial of degree  $m$  may be written as

$$\beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_m x^m$$

whereas FP of degree  $m$  has  $m$  integer and/or fractional powers  $p_1 < \cdots < p_m$ ,

$$\beta_0 + \beta_1 x^{(p_1)} + \beta_2 x^{(p_2)} + \cdots + \beta_m x^{(p_m)}$$

where for a power,  $p$ ,

$$x^{(p)} = \begin{cases} x^p & \text{if } p \neq 0 \\ \log x & \text{if } p = 0 \end{cases}$$

$x$  must be positive. An FP of first degree ( $m = 1$ ) involves one power or log transformation of  $x$ .

This family of FP functions may be extended in a mathematically natural way to include repeated powers. An FP of degree  $m$  with exactly  $m$  repeated powers of  $p$  is defined as

$$\beta_0 + \beta_1 x^{(p)} + \beta_2 x^{(p)} \log x + \cdots + \beta_m x^{(p)} (\log x)^{m-1}$$

For example, an FP of second degree ( $m = 2$ ) with repeated powers of 0.5 is

$$\beta_0 + \beta_1 x^{0.5} + \beta_2 x^{0.5} \log x$$

A general FP may include some unique and some repeated powers. For example, one with powers  $(-1, 1, 3, 3)$  is

$$\beta_0 + \beta_1 x^{-1} + \beta_2 x + \beta_3 x^3 + \beta_4 x^3 \log x$$

The permitted powers are restricted to the set  $\{-2, -1, -0.5, 0, 0.5, 1, 2, 3\}$  because our experience using FPs in data analysis indicates that including extra powers in the set is not often worthwhile.

Now we consider using FPs in regression modeling. If the values of the powers  $p_1, \dots, p_m$  were known, the FP would resemble a conventional multiple linear regression model with coefficients  $\beta_0, \beta_1, \dots, \beta_m$ . However, the powers are not (usually) known and must be estimated, together with the coefficients, from the data. Estimation involves a systematic search for the best power or combination of powers from the permitted set. For each possible combination, a linear regression model as just described is fit, and the corresponding deviance (defined as minus twice the log likelihood) is noted. The model with the lowest deviance is deemed to have the best fit, and the corresponding powers and regression coefficients constitute the final FP model. In practice,  $m = 2$  is often sufficient ([Royston and Sauerbrei 2008](#), 76).

## fracpoly

**fracpoly** finds and reports a multiple regression model comprising the best-fitting powers of  $xvar_1$  together with other covariates specified by  $xvar_2, \dots, xvarlist$ . The model that is fit depends on the type of *regression\_cmd* used.

The regression output for the best-fitting model may be reproduced by typing *regression\_cmd* without variables or options. **predict**, **test**, etc., may be used after **fracpoly**; the results will depend on *regression\_cmd*.

The standard errors of the fitted values (as estimated after use of **fracpoly** by using **predict** or **fracpred** with the **stdp** option) are somewhat too low because no allowance has been made for the estimation of the powers.

If  $xvar_1$  has any negative or zero values, **fracpoly** subtracts the minimum of  $xvar$  from  $xvar$  and then adds the rounding (or counting) interval. The interval is defined as the smallest positive difference between the ordered values of  $xvar$ . After this change of origin, the minimum value of  $xvar_1$  is positive, so FPs (which require  $xvar_1 > 0$ ) can be used. Unless the **noscaling** option is used, **fracpoly** scales the resulting variable by a power of 10 calculated from the data. The scaling is designed to improve numerical stability when fitting FP models.

After execution, **fracpoly** leaves in the dataset variables named  $lxvar\_1, lxvar\_2, \dots$ , which are the best-fitting FP powers of  $xvar_1$  (calculated, if necessary, after a change in origin and scale as just described, and if centering is specified, with a constant added to or subtracted from the values after FP transformation). Other variables, whose names follow the same convention, are left in the dataset if  $xvar_2$  has been specified.

## Centering

As discussed by Garrett (1995, 1998), covariate centering is a sensible, indeed often essential, step when reporting and interpreting the results of multiple regression models. For this and other reasons, centering has been introduced as the default option in **fracpoly**. As written, the familiar straight-line regression function  $E[y|x] = \beta_0 + \beta_1 x$  is “centered” to 0 in that  $\beta_0 = E[y|0]$ . This is fine if  $x = 0$  is a sensible base point. However, the sample values of  $x$  may not even encompass 0 (this is usually the case when FP models are contemplated). Then  $\beta_0$  is a meaningless intercept, and the standard error of its estimate  $\hat{\beta}_0$  will be large. For FP model  $E[y|x] = \beta_0 + \beta_1 x^{(p)}$ , the point  $x^{(p)} = 0$  may even correspond to  $x = \infty$  (consider  $p < 0$ ). The scheme adopted by **fracpoly** is to center on the mean of  $x$ . For example, for the FP  $E[y|x] = \beta_0 + \beta_1 x^p + \beta_2 x^q$ , **fracpoly** actually fits the model

$$E[y|x] = \beta_0 + \beta_1 (x^p - \bar{x}^p) + \beta_2 (x^q - \bar{x}^q)$$

where  $\bar{x}$  is the sample mean of the  $x$  values and  $E[y|\bar{x}] = \beta_0$ , giving  $\beta_0$  a respectable interpretation as the predicted value of  $y$  at the mean of  $x$ . This approach has the advantage that plots of the fitted values and 95% confidence intervals for  $E[y|x]$  as a function of  $x$ , even within a multiple regression model, are always sensible (provided that the other predictors are suitably centered—otherwise, the confidence limits can be alarmingly wide).

Sometimes centering on the mean is not appropriate, an example being a binary covariate where often you will want to center on the lower value, usually 0 (i.e., not center). You should then use the **center()** option to override the default. An example is **center(x1:mean,x2-x5:no,x6:1)**.

## Output with the compare option

If the `compare` option is used, `fracpoly` displays a table showing the best FP model for each degree  $k < m$  (including the model without  $x$  and the model linear in  $x$ ). Deviance differences between each FP model and the degree  $m$  model are also reported along with the corresponding  $p$ -values (Royston and Altman 1994b; Royston and Sauerbrei 2008).

The `compare` option implements a closed-test approach to selecting an FP model. It has the advantage of preserving the type I error probability at a nominal value. For example, suppose a nominal 5% significance level was chosen, and the test of FP2 versus the null model (i.e., omitting  $x$ ) was not significant. No further tests among FP models would then be done, and  $x$  would be considered nonsignificant, regardless of the results of any further model comparisons.

## fracgen

The basic syntax of `fracgen` is

```
fracgen varname # [# ...]
```

Each power (represented by `#` in the syntax diagram) should be separated by a space. `fracgen` creates new variables called `varname_1`, `varname_2`, etc. Each variable is labeled according to its power, preliminary linear transformation, and centering, if applied.

Positive or negative powers of `varname` are defined in the usual way. A power of zero is interpreted as `log`.

## Models with several continuous covariates

`fracpoly` estimates powers for FP models in just one continuous covariate ( $xvar_1$ ), though other covariates of any kind ( $xvar_2, \dots, xvarlist$ ) may be included as linear or predefined FP terms. An algorithm was suggested by Royston and Altman (1994b) for the joint estimation of FP models in several continuous covariates. It was later refined by Sauerbrei and Royston (1999) and is implemented in the Stata command `mfp`. See [R] `mfp` as well as Royston and Ambler (1998) and Royston and Sauerbrei (2008).

## Examples

### ► Example 1

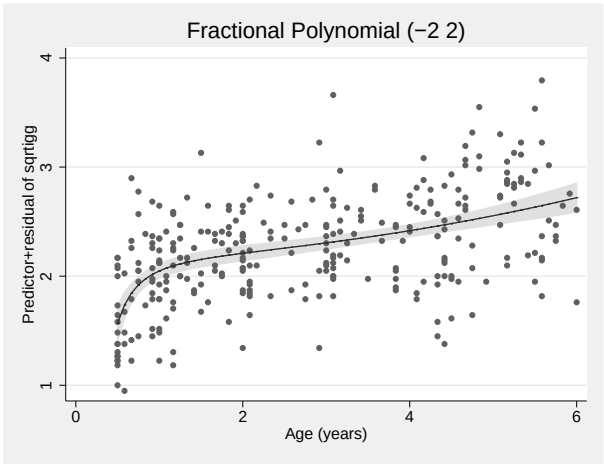
Consider the serum immunoglobulin G (IgG) dataset from Isaacs et al. (1983), which consists of 298 independent observations in young children. The dependent variable `sqrtigg` is the square root of the IgG concentration, and the independent variable `age` is the age of each child. (Preliminary Box–Cox analysis shows that a square root transformation removes the skewness in IgG.) The aim is to find a model that accurately predicts the mean of `sqrtigg` given `age`. We use `fracpoly` to find the best FP model of degree 2 (the default option) and graph the resulting fit and 95% confidence interval:

```
. use http://www.stata-press.com/data/r11/igg
(Immunoglobulin in children)
. fracpoly: regress sqrtigg age
.....
-> gen double Iage__1 = age^-2-.1299486216 if e(sample)
-> gen double Iage__2 = age^-2-7.695349038 if e(sample)
```

|          |            |     |            |                        |  |  |
|----------|------------|-----|------------|------------------------|--|--|
| Source   | SS         | df  | MS         | Number of obs = 298    |  |  |
| Model    | 22.2846976 | 2   | 11.1423488 | F( 2, 295) = 64.49     |  |  |
| Residual | 50.9676492 | 295 | .172771692 | Prob > F = 0.0000      |  |  |
|          |            |     |            | R-squared = 0.3042     |  |  |
|          |            |     |            | Adj R-squared = 0.2995 |  |  |
| Total    | 73.2523469 | 297 | .246640898 | Root MSE = .41566      |  |  |

|         |           |           |       |       |                      |          |
|---------|-----------|-----------|-------|-------|----------------------|----------|
| sqrtigg | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
| Iage__1 | -.1562156 | .027416   | -5.70 | 0.000 | -.2101713            | -.10226  |
| Iage__2 | .0148405  | .0027767  | 5.34  | 0.000 | .0093757             | .0203052 |
| _cons   | 2.283145  | .0305739  | 74.68 | 0.000 | 2.222974             | 2.343315 |

```
Deviance: 319.45. Best powers of age among 44 models fit: -2 2.
. fracplot age, msize(small)
```



The new variables created by `fracpoly` contain the best-fitting FP powers of `age`, as centered by `fracpoly`. For example, `Iage__1` is centered by subtracting the mean of `age` raised to the power  $-2$ . In general, the variables created by `fracpoly` are centered and possibly scaled, which is reflected in the estimated regression coefficients and intercept. Centering does have its advantages (see the [Centering](#) section earlier in this entry); however, sometimes you may want estimation for uncentered variables. To obtain regression results for uncentered and unscaled FP variables, specify options `center(no)` and `noscaling` to `fracpoly`. For a more detailed discussion, see [Royston and Sauerbrei \(2008, sec. 4.11\)](#).

The fitted curve has an asymmetric S shape. This model has powers  $(-2, 2)$  and deviance 319.45. As many as 44 models have been quietly fit in the search for the best powers. Now let’s look at models of degree  $\leq 4$ :

```
. fracpoly, degree(4) compare: regress sqrtigg age
.....
>
>
-> gen double lage__1 = ln(age)-1.020308063 if e(sample)
-> gen double lage__2 = age^3-21.34727694 if e(sample)
-> gen double lage__3 = age^3*ln(age)-21.78079878 if e(sample)
-> gen double lage__4 = age^3*ln(age)^2-22.22312461 if e(sample)
```

| Source   | SS         | df  | MS         | Number of obs = 298    |  |  |
|----------|------------|-----|------------|------------------------|--|--|
| Model    | 22.5754541 | 4   | 5.64386353 | F( 4, 293) = 32.63     |  |  |
| Residual | 50.6768927 | 293 | .172958678 | Prob > F = 0.0000      |  |  |
|          |            |     |            | R-squared = 0.3082     |  |  |
|          |            |     |            | Adj R-squared = 0.2987 |  |  |
| Total    | 73.2523469 | 297 | .246640898 | Root MSE = .41588      |  |  |

| sqrtigg | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|---------|-----------|-----------|-------|-------|----------------------|-----------|
| lage__1 | .8761824  | .1898721  | 4.61  | 0.000 | .5024962             | 1.249869  |
| lage__2 | -.1922029 | .0684934  | -2.81 | 0.005 | -.3270044            | -.0574015 |
| lage__3 | .2043794  | .074947   | 2.73  | 0.007 | .0568767             | .3518821  |
| lage__4 | -.0560067 | .0212969  | -2.63 | 0.009 | -.097921             | -.0140924 |
| _cons   | 2.238735  | .0482705  | 46.38 | 0.000 | 2.143734             | 2.333736  |

Deviance: 317.74. Best powers of age among 494 models fit: 0 3 3 3.

Fractional polynomial model comparisons:

| age          | df | Deviance | Res. SD | Dev. dif. | P (*) | Powers  |
|--------------|----|----------|---------|-----------|-------|---------|
| Not in model | 0  | 427.539  | .49663  | 109.795   | 0.000 |         |
| Linear       | 1  | 337.561  | .42776  | 19.818    | 0.006 | 1       |
| m = 1        | 2  | 327.436  | .420554 | 9.692     | 0.140 | 0       |
| m = 2        | 4  | 319.448  | .415658 | 1.705     | 0.794 | -2 2    |
| m = 3        | 6  | 319.275  | .416243 | 1.532     | 0.473 | -2 1 1  |
| m = 4        | 8  | 317.744  | .415883 | —         | —     | 0 3 3 3 |

(\*) P-value from deviance difference comparing reported model with m = 4 model

It appears that the degree 4 FP model is not significantly different from the other FP models (at the 5% level).

Let's compare the curve shape from the  $m = 2$  model with that from a conventional quartic polynomial, whose fit turns out to be significantly better than a cubic (not shown). We use the ability of `fracpoly` both to generate the required powers of `age`, namely,  $(1, 2, 3, 4)$  for the quartic and  $(-2, 2)$  for the second-degree FP, and to fit the model. We fit both models and graph the resulting curves:

(Continued on next page)

```
. fracpoly: regress sqrtigg age 1 2 3 4
-> gen double lage__1 = age-2.774049213 if e(sample)
-> gen double lage__2 = age^2-7.695349038 if e(sample)
-> gen double lage__3 = age^3-21.34727694 if e(sample)
-> gen double lage__4 = age^4-59.21839681 if e(sample)
```

| Source   | SS         | df  | MS         | Number of obs = 298    |  |  |
|----------|------------|-----|------------|------------------------|--|--|
| Model    | 22.5835458 | 4   | 5.64588646 | F( 4, 293) = 32.65     |  |  |
| Residual | 50.668801  | 293 | .172931061 | Prob > F = 0.0000      |  |  |
|          |            |     |            | R-squared = 0.3083     |  |  |
|          |            |     |            | Adj R-squared = 0.2989 |  |  |
| Total    | 73.2523469 | 297 | .246640898 | Root MSE = .41585      |  |  |

| sqrtigg | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|---------|-----------|-----------|-------|-------|----------------------|-----------|
| lage__1 | 2.047831  | .4595962  | 4.46  | 0.000 | 1.143302             | 2.952359  |
| lage__2 | -1.058902 | .2822803  | -3.75 | 0.000 | -1.614456            | -.5033479 |
| lage__3 | .2284917  | .0667591  | 3.42  | 0.001 | .0971037             | .3598798  |
| lage__4 | -.0168534 | .0053321  | -3.16 | 0.002 | -.0273475            | -.0063594 |
| _cons   | 2.240012  | .0480157  | 46.65 | 0.000 | 2.145512             | 2.334511  |

Deviance: 317.70.

```
. predict fit1
(option xb assumed; fitted values)
. fracpoly: regress sqrtigg age -2 2
-> gen double lage__1 = age^-2-.1299486216 if e(sample)
-> gen double lage__2 = age^2-7.695349038 if e(sample)
```

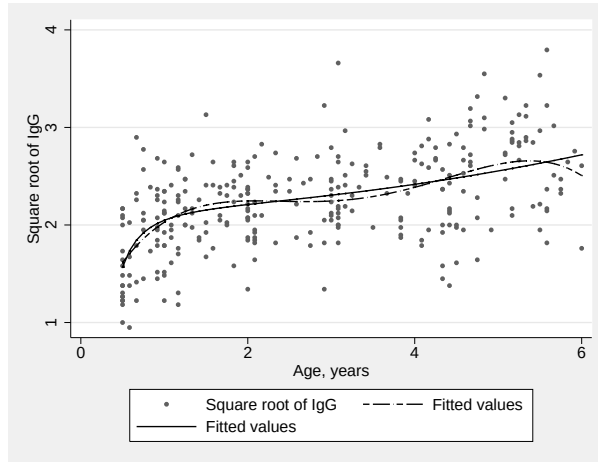
| Source   | SS         | df  | MS         | Number of obs = 298    |  |  |
|----------|------------|-----|------------|------------------------|--|--|
| Model    | 22.2846976 | 2   | 11.1423488 | F( 2, 295) = 64.49     |  |  |
| Residual | 50.9676492 | 295 | .172771692 | Prob > F = 0.0000      |  |  |
|          |            |     |            | R-squared = 0.3042     |  |  |
|          |            |     |            | Adj R-squared = 0.2995 |  |  |
| Total    | 73.2523469 | 297 | .246640898 | Root MSE = .41566      |  |  |

| sqrtigg | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|---------|-----------|-----------|-------|-------|----------------------|----------|
| lage__1 | -.1562156 | .027416   | -5.70 | 0.000 | -.2101713            | -.10226  |
| lage__2 | .0148405  | .0027767  | 5.34  | 0.000 | .0093757             | .0203052 |
| _cons   | 2.283145  | .0305739  | 74.68 | 0.000 | 2.222974             | 2.343315 |

Deviance: 319.45.

```
. predict fit2
(option xb assumed; fitted values)
```

```
. scatter sqrtigg fit1 fit2 age, c(. 1 1) m(o i i) msize(small)
> clpattern(. -_.) ytitle("Square root of IgG") xtitle("Age, years")
```



The quartic curve has an unsatisfactory wavy appearance that is implausible for the known behavior of IgG, the serum level of which increases throughout early life. The FP curve increases monotonically and is therefore biologically the more plausible curve. The two models have approximately the same deviance.

◀

## ► Example 2

Data from Smith et al. (1992) contain times to complete healing of leg ulcers in a randomized controlled clinical trial of two treatments in 192 elderly patients. Several covariates were available, of which an important one is `mothson`, the number of months since the recorded onset of the ulcer. Because the response variable is time to an event of interest and some (in fact, about one-half) of the times are censored, using Cox regression to analyze the data is appropriate. We consider FPs in `mothson`, adjusting for four other covariates: `age`; `ulcarea`, the area of tissue initially affected by the ulcer; `deepppg`, a binary variable indicating the presence or absence of deep vein involvement; and `treat`, a binary variable indicating treatment type. We fit FPs of degree 1 and 2:

```
. use http://www.stata-press.com/data/r11/legulcer, clear
(Leg ulcer clinical trial)
. stset ttevent, fail(cens)
(output omitted)
```

(Continued on next page)

```
. fracpoly, compare: stcox mthson age ulcarea deepppg treat, nohr
-> gen double lage__1 = age-73.453125 if e(sample)
-> gen double lulca__1 = ulcarea-1326.203125 if e(sample)
-> gen double ltrea__1 = treat-1 if e(sample)
.....
-> gen double lmths__1 = X^.5-.4930242557 if e(sample)
-> gen double lmths__2 = X^.5*ln(X)+.6973304564 if e(sample)
 (where: X = (mthson+1)/100)

 failure _d: censored
 analysis time _t: ttevent

Iteration 0: log likelihood = -422.65089
Iteration 1: log likelihood = -390.49313
Iteration 2: log likelihood = -383.44258
Iteration 3: log likelihood = -374.28707
Iteration 4: log likelihood = -369.31417
Iteration 5: log likelihood = -368.38104
Iteration 6: log likelihood = -368.35448
Iteration 7: log likelihood = -368.35446
Refining estimates:
Iteration 0: log likelihood = -368.35446

Cox regression -- Breslow method for ties

No. of subjects = 192 Number of obs = 192
No. of failures = 92
Time at risk = 13825

Log likelihood = -368.35446 LR chi2(6) = 108.59
 Prob > chi2 = 0.0000
```

| _t       | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| lmths__1 | -2.81425  | .6996385  | -4.02 | 0.000 | -4.185516            | -1.442984 |
| lmths__2 | 1.541451  | .4703143  | 3.28  | 0.001 | .6196521             | 2.46325   |
| lage__1  | -.0261111 | .0087983  | -2.97 | 0.003 | -.0433556            | -.0088667 |
| lulca__1 | -.0017491 | .000359   | -4.87 | 0.000 | -.0024527            | -.0010455 |
| deepppg  | -.5850499 | .2163173  | -2.70 | 0.007 | -1.009024            | -.1610758 |
| ltrea__1 | -.1624663 | .2171048  | -0.75 | 0.454 | -.5879838            | .2630513  |

Deviance: 736.71. Best powers of mthson among 44 models fit: .5 .5.

Fractional polynomial model comparisons:

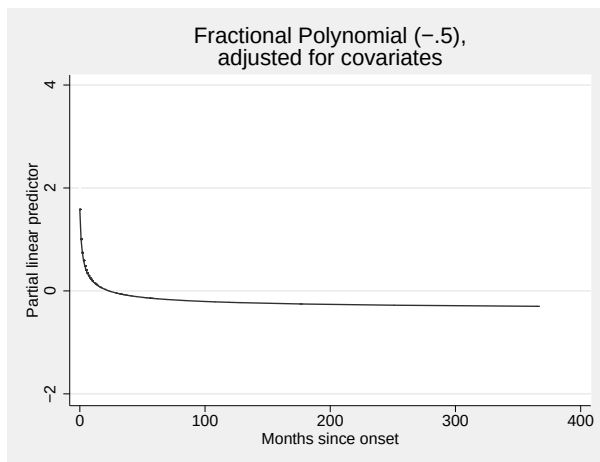
| mthson       | df | Deviance | Dev. dif. | P (*) | Powers |  |
|--------------|----|----------|-----------|-------|--------|--|
| Not in model | 0  | 754.345  | 17.636    | 0.001 |        |  |
| Linear       | 1  | 751.680  | 14.971    | 0.002 | 1      |  |
| m = 1        | 2  | 738.969  | 2.260     | 0.323 | -.5    |  |
| m = 2        | 4  | 736.709  | —         | —     | .5 .5  |  |

(\*) P-value from deviance difference comparing reported model with m = 2 model

The best-fit FP of degree 2 has powers (0.5,0.5) and deviance 736.71. However, this model does not fit significantly better than the FP of degree 1 (at the 5% level), which has power −0.5 and deviance 738.97. We prefer the model with  $m = 1$ , for which the partial linear predictor is shown on the next page.



```
. quietly fracpoly, degree(1): stcox mthson age ulcarea deepppg treat, nohr
. fracplot, ytitle(Partial linear predictor) m(i) ciopts(bcolor(white))
```



The hazard for healing is much higher for patients whose ulcer is of recent onset than for those who have had an ulcer for many months.

`fracpoly` has automatically centered the predictors on their mean values, but because in Cox regression there is no constant term, we cannot see the effects of centering in the table of regression estimates. The effects would be present if we were to graph the baseline hazard or survival function because these functions are defined with all predictors set equal to 0.

A more appropriate analysis of this dataset, if one wished to model all the predictors, possibly with FP functions, would be to use `mfp`; see [\[R\]](#) `mfp`.

◀

*(Continued on next page)*

## Saved results

In addition to what *regression\_cmd* saves, *fracpoly* saves the following in `e()`:

### Scalars

|                                                     |                                                   |
|-----------------------------------------------------|---------------------------------------------------|
| <code>e(fp_N)</code>                                | number of nonmissing observations                 |
| <code>e(fp_dev)</code>                              | deviance for FP model of degree $m$               |
| <code>e(fp_df)</code>                               | FP model degrees of freedom                       |
| <code>e(fp_d0)</code>                               | deviance for model without $xvar_1$               |
| <code>e(fp_s0)</code>                               | residual SD for model without $xvar_1$            |
| <code>e(fp_dlin)</code>                             | deviance for model linear in $xvar_1$             |
| <code>e(fp_slin)</code>                             | residual SD for model linear in $xvar_1$          |
| <code>e(fp_d1)</code> , <code>e(fp_d2)</code> , ... | deviances for FP models of degree 1,2,..., $m$    |
| <code>e(fp_s1)</code> , <code>e(fp_s2)</code> , ... | residual SDs for FP models of degree 1,2,..., $m$ |

### Macros

|                                                     |                                                                               |
|-----------------------------------------------------|-------------------------------------------------------------------------------|
| <code>e(fp_cmd)</code>                              | <code>fracpoly</code>                                                         |
| <code>e(cmdline)</code>                             | command as typed                                                              |
| <code>e(fp_dep)</code>                              | <code>yvar1</code> ( <code>yvar2</code> )                                     |
| <code>e(fp_rhs)</code>                              | $xvar_1$                                                                      |
| <code>e(fp_base)</code>                             | variables in $xvar_2$ , ..., $xvarlist$ after centering and FP transformation |
| <code>e(fp_xp)</code>                               | <code>lxvar__1</code> , <code>lxvar__2</code> , etc.                          |
| <code>e(fp_fvl)</code>                              | variables in model finally estimated                                          |
| <code>e(fp_wgt)</code>                              | weight type or ""                                                             |
| <code>e(fp_wexp)</code>                             | weight expression if ' <code>e(fp_wgt)</code> ' != ""                         |
| <code>e(fp_pwr)</code>                              | powers for FP model of degree $m$                                             |
| <code>e(fp_x1)</code> , <code>e(fp_x2)</code> , ... | $xvar_1$ and variables in model                                               |
| <code>e(fp_k1)</code> , <code>e(fp_k2)</code> , ... | powers for FP models of degree 1,2,..., $m$                                   |

Residual SDs are stored only when *regression\_cmd* is **regress**.

## Methods and formulas

*fracpoly* and *fracgen* are implemented as ado-files.

The general definition of an FP, accommodating possible repeated powers, may be written for functions  $H_1(x), \dots, H_m(x)$  as

$$\beta_0 + \sum_{j=1}^m \beta_j H_j(x)$$

where  $H_1(x) = x^{(p_1)}$  and for  $j = 2, \dots, m$ ,

$$H_j(x) = \begin{cases} x^{(p_j)} & \text{if } p_j \neq p_{j-1} \\ H_{j-1}(x) \log x & \text{if } p_j = p_{j-1} \end{cases}$$

For example, an FP of degree 3 with powers (1,3,3) has  $H_1(x) = x$ ,  $H_2(x) = x^3$ , and  $H_3(x) = x^3 \log x$  and equals  $\beta_0 + \beta_1 x + \beta_2 x^3 + \beta_3 x^3 \log x$ .

An FP model of degree  $m$  is taken to have  $2m + 1$  degrees of freedom (df): one for  $\beta_0$  and one for each  $\beta_j$  and its associated power. Because the powers in an FP are chosen from a finite set rather than from the entire real line, the df defined in this way are approximate.

The deviance  $D$  of a model is defined as  $-2$  times its maximized log likelihood. For normal-errors models, we use the formula

$$D = n \left( 1 - \bar{l} + \log \frac{2\pi \text{RSS}}{n} \right)$$

where  $n$  is the sample size,  $\bar{l}$  is the mean of the lognormalized weights ( $\bar{l} = 0$  if the weights are all equal), and RSS is the residual sum of squares as fit by `regress`.

The `compare` option causes `fracpoly` to report a table comparing FP models of degree  $k < m$  to the degree  $m$  FP model. Suppose that we are comparing FP regression models with degrees  $k$  and  $m$ . The  $p$ -values reported by `compare` are calculated differently for normal and nonnormal regressions. Let  $D_k$  and  $D_m$  be the deviances of the models with degrees  $k$  and  $m$ , respectively. For normal-errors models such as `regress`, a variance ratio  $F$  is calculated as

$$F = \frac{n_2}{n_1} \left\{ \exp \left( \frac{D_k - D_m}{n} \right) - 1 \right\}$$

where  $n_1$  is the numerator df (here,  $2m - 2k$ ) and  $n_2$  is the denominator df (equal to `rdf`  $- 2m$ , where `rdf` is the residual df for the regression model involving only the covariates in `xvar2`, if any, but not  $x$ ). The  $p$ -value is obtained by referring  $F$  to an  $F$  distribution on  $(2, \text{rdf})$  df.

For nonnormal models (`clogit`, `glm`, `logistic`, ... `stcox`, or `streg`; not `regress`), the  $p$ -value is obtained by referring  $D_k - D_m$  to a  $\chi^2$  distribution on  $2m - 2k$  df. These  $p$ -values for comparing models are approximate and are typically somewhat conservative (Royston and Altman 1994b).

The component-plus-residual values graphed by `fracplot` are calculated as follows: let the data consist of triplets  $(y_i, x_i, \mathbf{z}_i)$ ,  $i = 1, \dots, n$ , where  $\mathbf{z}_i$  is the vector of covariates for the  $i$ th observation, after applying possible FP transformation and centering as described earlier. Let  $\hat{\eta}_i = \hat{\beta}_0 + \{\mathbf{H}(x_i) - \mathbf{H}(x_0)\}' \hat{\beta} + \mathbf{z}_i' \hat{\gamma}$  be the linear predictor from the FP model, as given by the `fracpred` command, or equivalently, by the `predict` command with the `xb` option, after the use of `fracpoly`. Here  $\mathbf{H}(x_i) = \{H_1(x_i), \dots, H_m(x_i)\}'$  is the vector of FP functions described above,  $\mathbf{H}(x_0) = \{H_1(x_0), \dots, H_m(x_0)\}'$  is the vector of centering to  $x_0$  ( $x_0$  is often chosen to be the mean of the  $x_i$ ),  $\hat{\beta}$  is the estimated parameter vector, and  $\hat{\gamma}$  is the estimated parameter vector for the covariates. The values  $\hat{\eta}_i^* = \hat{\beta}_0 + \{\mathbf{H}(x_i) - \mathbf{H}(x_0)\}' \hat{\beta}$  represent the behavior of the FP model for  $x$  at fixed values  $\mathbf{z} = \mathbf{0}$  of the (centered) covariates. The  $i$ th component-plus-residual is defined as  $\hat{\eta}_i^* + d_i$ , where  $d_i$  is the deviance residual for the  $i$ th observation. For normal-errors models,  $d_i = \sqrt{w_i}(y_i - \hat{\eta}_i)$ , where  $w_i$  is the case weight (or 1, if `weight` is not specified). For logistic, Cox, and generalized linear regression models, see [R] `logistic`, [R] `probit`, [ST] `stcox`, and [R] `glm`, respectively, for the formula for  $d_i$ . The formula for `poisson` models is the same as that for `glm` with `family(poisson)`. For `stcox`,  $d_i$  is the partial martingale residual (see [ST] `stcox postestimation`).

## Acknowledgment

`fracpoly` and `fracgen` were written by Patrick Royston, MRC Clinical Trials Unit, London.

## References

- Beckett, S. 1995. [sg26.2: Calculating and graphing fractional polynomials](#). *Stata Technical Bulletin* 24: 14–16. Reprinted in *Stata Technical Bulletin Reprints*, vol. 4, pp. 129–132. College Station, TX: Stata Press.
- Garrett, J. M. 1995. [sg33: Calculation of adjusted means and adjusted proportions](#). *Stata Technical Bulletin* 24: 22–25. Reprinted in *Stata Technical Bulletin Reprints*, vol. 4, pp. 161–165. College Station, TX: Stata Press.

- . 1998. [sg33.1: Enhancements for calculation of adjusted means and adjusted proportions](#). *Stata Technical Bulletin* 43: 16–24. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 111–123. College Station, TX: Stata Press.
- Isaacs, D., D. G. Altman, C. E. Tidmarsh, H. B. Valman, and A. D. Webster. 1983. Serum immunoglobulin concentrations in preschool children measured by laser nephelometry: Reference ranges for IgG, IgA, IgM. *Journal of Clinical Pathology* 36: 1193–1196.
- Royston, P. 1995. [sg26.3: Fractional polynomial utilities](#). *Stata Technical Bulletin* 25: 9–13. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 82–87. College Station, TX: Stata Press.
- Royston, P., and D. G. Altman. 1994a. [sg26: Using fractional polynomials to model curved regression relationships](#). *Stata Technical Bulletin* 21: 11–23. Reprinted in *Stata Technical Bulletin Reprints*, vol. 4, pp. 110–128. College Station, TX: Stata Press.
- . 1994b. Regression using fractional polynomials of continuous covariates: Parsimonious parametric modelling (with discussion). *Applied Statistics* 43: 429–467.
- Royston, P., and G. Ambler. 1998. [sg81: Multivariable fractional polynomials](#). *Stata Technical Bulletin* 43: 24–32. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 123–132. College Station, TX: Stata Press.
- . 1999a. [sg112: Nonlinear regression models involving power or exponential functions of covariates](#). *Stata Technical Bulletin* 49: 25–30. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 173–179. College Station, TX: Stata Press.
- . 1999b. [sg81.1: Multivariable fractional polynomials: Update](#). *Stata Technical Bulletin* 49: 17–23. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 161–168. College Station, TX: Stata Press.
- . 1999c. [sg112.1: Nonlinear regression models involving power or exponential functions of covariates: Update](#). *Stata Technical Bulletin* 50: 26. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, p. 180. College Station, TX: Stata Press.
- . 1999d. [sg81.2: Multivariable fractional polynomials: Update](#). *Stata Technical Bulletin* 50: 25. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, p. 168. College Station, TX: Stata Press.
- Royston, P., and W. Sauerbrei. 2008. *Multivariable Model-building: A Pragmatic Approach to Regression Analysis Based on Fractional Polynomials for Modelling Continuous Variables*. Chichester, UK: Wiley.
- Sauerbrei, W., and P. Royston. 1999. Building multivariable prognostic and diagnostic models: Transformation of the predictors by using fractional polynomials. *Journal of the Royal Statistical Society, Series A* 162: 71–94.
- Smith, J. M., C. J. Dore, A. Charlett, and J. D. Lewis. 1992. A randomized trial of Biofilm dressing for venous leg ulcers. *Phlebology* 7: 108–113.

## Also see

[R] [fracpoly postestimation](#) — Postestimation tools for fracpoly

[R] [mfp](#) — Multivariable fractional polynomial models

[U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are of special interest after `fracpoly`:

| command               | description                                                                        |
|-----------------------|------------------------------------------------------------------------------------|
| <code>fracplot</code> | plot data and fit from most recently fit fractional polynomial model               |
| <code>fracpred</code> | create variable containing prediction, deviance residuals, or SEs of fitted values |

For information about these commands, see below.

The following standard postestimation commands are also available if available after `regression_cmd`:

| command                | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>     | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estimates</code> | cataloging estimation results                                                                       |
| <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>  | link test for model specification                                                                   |
| <code>lrtest</code>    | likelihood-ratio test                                                                               |
| <code>margins</code>   | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>   | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code> | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details.

Special-interest postestimation commands

- `fracplot` plots the data and fit, with 95% confidence limits, from the most recently fit fractional polynomial (FP) model. The data and fit are plotted against *varname*, which may be *xvar*<sub>1</sub> or another of the covariates (*xvar*<sub>2</sub>, ..., or a variable from *xvarlist*). If *varname* is not specified, *xvar*<sub>1</sub> is assumed.

`fracpred` creates *newvar* containing the fitted index or deviance residuals for the whole model, or the fitted index or its standard error for *varname*, which may be *xvar*<sub>1</sub> or another covariate.

## Syntax for predict

The behavior of `predict` following `fracpoly` is determined by *regression\_cmd*. See the corresponding *regression\_cmd* **postestimation** entry for available `predict` options.

Also see information on `fracpred` below.

## Syntax for fracplot and fracpred

Plot data and fit from most recently fit fractional polynomial model

```
fracplot [varname] [if] [in] [, fracplot_options]
```

Create variable containing the prediction, deviance residuals, or SEs of fitted values

```
fracpred newvar [, fracpred_options]
```

| <i>fracplot_options</i>                  | description                                                                      |
|------------------------------------------|----------------------------------------------------------------------------------|
| Plot                                     |                                                                                  |
| <i>marker_options</i>                    | change look of markers (color, size, etc.)                                       |
| <i>marker_label_options</i>              | add marker labels; change look or position                                       |
| Fitted line                              |                                                                                  |
| <u>lineopts</u> ( <i>cline_options</i> ) | affect rendition of the fitted line                                              |
| CI plot                                  |                                                                                  |
| <u>ciopts</u> ( <i>area_options</i> )    | affect rendition of the confidence bands                                         |
| Add plots                                |                                                                                  |
| <u>addplot</u> ( <i>plot</i> )           | add other plots to the generated graph                                           |
| Y axis, X axis, Titles, Legend, Overall  |                                                                                  |
| <i>twoway_options</i>                    | any options other than <code>by()</code> documented in [G] <i>twoway_options</i> |

| <i>fracpred_options</i>       | description                                                 |
|-------------------------------|-------------------------------------------------------------|
| <u>for</u> ( <i>varname</i> ) | compute prediction for <i>varname</i>                       |
| <u>dresid</u>                 | compute deviance residuals                                  |
| <u>stdp</u>                   | compute standard errors of the fitted values <i>varname</i> |

`fracplot` is not allowed after `fracpoly` or `mfp` with `clogit`, `mlogit`, or `stcrreg`. `fracpred`, `dresid` is not allowed after `fracpoly` or `mfp` with `clogit`, `mlogit`, or `stcrreg`.

## Menu

### fracplot

Statistics > Linear models and related > Fractional polynomials > Fractional polynomial regression plot

### fracpred

Statistics > Linear models and related > Fractional polynomials > Fractional polynomial prediction

## Options for fracplot

### Plot

*marker\_options* affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [\[G\] marker\\_options](#).

*marker\_label\_options* specify if and how the markers are to be labeled; see [\[G\] marker\\_label\\_options](#).

### Fitted line

`lineopts(cline_options)` affect the rendition of the fitted line; see [\[G\] cline\\_options](#).

### CI plot

`ciopts(area_options)` affect the rendition of the confidence bands; see [\[G\] area\\_options](#).

### Add plots

`addplot(plot)` provides a way to add other plots to the generated graph. See [\[G\] addplot\\_option](#).

### Y axis, X axis, Titles, Legend, Overall

*twoway\_options* are any of the options documented in [\[G\] twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [\[G\] title\\_options](#)) and for saving the graph to disk (see [\[G\] saving\\_option](#)).

## Options for fracpred

`for(varname)` specifies (partial) prediction for variable *varname*. The fitted values are adjusted to the value specified by the `adjust()` option in `fracpoly`.

`dresid` specifies that deviance residuals be calculated.

`stdp` specifies calculation of the standard errors of the fitted values *varname*, adjusted for all the other predictors at the values specified by `adjust()`.

## Remarks

`fracplot` actually produces a component-plus-residual plot. For normal-error models with constant weights and one covariate, this amounts to a plot of the observations with the fitted line inscribed. For other normal-error models, weighted residuals are calculated and added to the fitted values.

For models with additional covariates, the line is the partial linear predictor for the variable in question (*xvar*<sub>1</sub> or a covariate) and includes the intercept  $\beta_0$ .

For generalized linear and Cox models, the fitted values are plotted on the scale of the “index” (linear predictor). Deviance residuals are added to the (partial) linear predictor to give component-plus-residual values. These values are plotted as small circles.

See [R] [fracpoly](#) for examples using `fracplot`.

## Methods and formulas

All postestimation commands listed above and `fracplot` and `fracpred` are implemented as ado-files.

See [Methods and formulas](#) in [R] [fracpoly](#) for notation.

The component-plus-residual values graphed by `fracplot` are calculated as follows: Let the data consist of triplets  $(y_i, x_i, \mathbf{z}_i)$ ,  $i = 1, \dots, n$ , where  $\mathbf{z}_i$  is the vector of covariates for the  $i$ th observation, after applying possible fractional polynomial transformation and adjustment as described earlier. Let  $\hat{\eta}_i = \hat{\beta}_0 + \{\mathbf{H}(x_i) - \mathbf{H}(x_0)\}' \hat{\beta} + \mathbf{z}_i' \hat{\gamma}$  be the linear predictor from the FP model, as given by the `fracpred` command or, equivalently, by the `predict` command with the `xb` option, following `fracpoly`. Here  $\mathbf{H}(x_i) = \{H_1(x_i), \dots, H_m(x_i)\}'$  is the vector of FP functions described above,  $\mathbf{H}(x_0) = \{H_1(x_0), \dots, H_m(x_0)\}'$  is the vector of adjustments to  $x_0$  (often,  $x_0$  is chosen to be the mean of the  $x_i$ ),  $\hat{\beta}$  is the estimated parameter vector, and  $\hat{\gamma}$  is the estimated parameter vector for the covariates. The values  $\hat{\eta}_i^* = \hat{\beta}_0 + \{\mathbf{H}(x_i) - \mathbf{H}(x_0)\}' \hat{\beta}$  represent the behavior of the FP model for  $x$  at fixed values  $\mathbf{z} = \mathbf{0}$  of the (adjusted) covariates. The  $i$ th component-plus-residual is defined as  $\hat{\eta}_i^* + d_i$ , where  $d_i$  is the deviance residual for the  $i$ th observation. For normal-errors models,  $d_i = \sqrt{w_i}(y_i - \hat{\eta}_i)$ , where  $w_i$  is the case weight (or 1, if *weight* is not specified). For logistic, Cox, and generalized linear regression models, see [R] [logistic](#), [R] [probit](#), [ST] [stcox](#), and [R] [glm](#) for the formula for  $d_i$ . The formula for `poisson` models is the same as that for `glm` with `family(poisson)`. For `stcox`,  $d_i$  is the partial martingale residual (see [ST] [stcox postestimation](#)).

`fracplot` plots the values of  $d_i$  and the curve represented by  $\hat{\eta}_i^*$  against  $x_i$ . The confidence interval for  $\hat{\eta}_i^*$  is obtained from the variance–covariance matrix of the entire model and takes into account the uncertainty in estimating  $\beta_0$ ,  $\beta$ , and  $\gamma$  (but not in estimating the FP powers for  $x$ ).

`fracpred` with the `for(varname)` option calculates the predicted index at  $x_i = x_0$  and  $\mathbf{z}_i = \mathbf{0}$ ; that is,  $\hat{\eta}_i = \hat{\beta}_0 + \{\mathbf{H}(x_i) - \mathbf{H}(x_0)\}' \hat{\beta}$ . The standard error is calculated from the variance–covariance matrix of  $(\hat{\beta}_0, \hat{\beta})$ , again ignoring estimation of the powers.

## Acknowledgment

`fracplot` and `fracpred` were written by Patrick Royston of the MRC Clinical Trials Unit, London.

## Also see

[R] [fracpoly](#) — Fractional polynomial regression

[U] [20 Estimation and postestimation commands](#)



Syntax

| frontier <i>depvar</i> [ <i>indepvars</i> ] [ <i>if</i> ] [ <i>in</i> ] [ <i>weight</i> ] [, <i>options</i> ]                       |                                                                                                                     |
|-------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| <i>options</i>                                                                                                                      | description                                                                                                         |
| Model                                                                                                                               |                                                                                                                     |
| <u>noconstant</u>                                                                                                                   | suppress constant term                                                                                              |
| <u>distribution</u> ( <u>hnormal</u> )                                                                                              | half-normal distribution for the inefficiency term                                                                  |
| <u>distribution</u> ( <u>exponential</u> )                                                                                          | exponential distribution for the inefficiency term                                                                  |
| <u>distribution</u> ( <u>tnormal</u> )                                                                                              | truncated-normal distribution for the inefficiency term                                                             |
| <u>ufrom</u> ( <i>matrix</i> )                                                                                                      | specify untransformed log likelihood; only with <u>d</u> ( <u>tnormal</u> )                                         |
| <u>cm</u> ( <i>varlist</i> [, <u>noconstant</u> ])                                                                                  | fit conditional mean model; only with <u>d</u> ( <u>tnormal</u> ); use <u>noconstant</u> to suppress constant term  |
| Model 2                                                                                                                             |                                                                                                                     |
| <u>constraints</u> ( <i>constraints</i> )                                                                                           | apply specified linear constraints                                                                                  |
| <u>collinear</u>                                                                                                                    | keep collinear variables                                                                                            |
| <u>uhet</u> ( <i>varlist</i> [, <u>noconstant</u> ])                                                                                | explanatory variables for technical inefficiency variance function; use <u>noconstant</u> to suppress constant term |
| <u>vhet</u> ( <i>varlist</i> [, <u>noconstant</u> ])                                                                                | explanatory variables for idiosyncratic error variance function; use <u>noconstant</u> to suppress constant term    |
| <u>cost</u>                                                                                                                         | fit cost frontier model; default is production frontier model                                                       |
| SE                                                                                                                                  |                                                                                                                     |
| <u>vce</u> ( <i>vcetype</i> )                                                                                                       | <i>vcetype</i> may be <u>oim</u> , <u>opg</u> , <u>bootstrap</u> , or <u>jackknife</u>                              |
| Reporting                                                                                                                           |                                                                                                                     |
| <u>level</u> (#)                                                                                                                    | set confidence level; default is <u>level</u> (95)                                                                  |
| <u>nocnsreport</u>                                                                                                                  | do not display constraints                                                                                          |
| <u>display_options</u>                                                                                                              | control spacing and display of omitted variables and base and empty cells                                           |
| Maximization                                                                                                                        |                                                                                                                     |
| <u>maximize_options</u>                                                                                                             | control the maximization process; seldom used                                                                       |
| <sup>†</sup> <u>coeflegend</u>                                                                                                      | display coefficients' legend instead of coefficient table                                                           |
| <sup>†</sup> <u>coeflegend</u> does not appear in the dialog box.                                                                   |                                                                                                                     |
| <i>indepvars</i> and <i>varlist</i> may contain factor variables; see [U] 11.4.3 Factor variables.                                  |                                                                                                                     |
| <u>bootstrap</u> , <u>by</u> , <u>jackknife</u> , <u>rolling</u> , and <u>statsby</u> are allowed; see [U] 11.1.10 Prefix commands. |                                                                                                                     |
| Weights are not allowed with the <u>bootstrap</u> prefix.                                                                           |                                                                                                                     |
| <u>fweights</u> , <u>iweights</u> , and <u>pweights</u> are allowed; see [U] 11.1.6 weight.                                         |                                                                                                                     |
| See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.                                     |                                                                                                                     |

## Menu

Statistics > Linear models and related > Frontier models

## Description

`frontier` fits stochastic production or cost frontier models; the default is a production frontier model. It provides estimators for the parameters of a linear model with a disturbance that is assumed to be a mixture of two components, which have a strictly nonnegative and symmetric distribution, respectively. `frontier` can fit models in which the nonnegative distribution component (a measurement of inefficiency) is assumed to be from a half-normal, exponential, or truncated-normal distribution. See [Kumbhakar and Lovell \(2000\)](#) for a detailed introduction to frontier analysis.

## Options

### Model

`noconstant`; see [\[R\] estimation options](#).

`distribution(distname)` specifies the distribution for the inefficiency term as half-normal (`hnormal`), exponential, or truncated-normal (`tnormal`). The default is `hnormal`.

`ufrom(matrix)` specifies a  $1 \times K$  matrix of untransformed starting values when the distribution is truncated-normal (`tnormal`). `frontier` can estimate the parameters of the model by maximizing either the log likelihood or a transformed log likelihood (see [Methods and formulas](#)). `frontier` automatically transforms the starting values before passing them on to the transformed log likelihood. The matrix must have the same number of columns as there are parameters to estimate.

`cm(varlist [, noconstant])` may be used only with `distribution(tnormal)`. Here `frontier` will fit a conditional mean model in which the mean of the truncated-normal distribution is modeled as a linear function of the set of covariates specified in *varlist*. Specifying `noconstant` suppresses the constant in the mean function.

### Model 2

`constraints(constraints), collinear`; see [\[R\] estimation options](#).

By default, when fitting the truncated-normal model or the conditional mean model, `frontier` maximizes a transformed log likelihood. When constraints are applied, `frontier` will maximize the untransformed log likelihood with constraints defined in the untransformed metric.

`uhet(varlist [, noconstant])` specifies that the technical inefficiency component is heteroskedastic, with the variance function depending on a linear combination of *varlist*<sub>*u*</sub>. Specifying `noconstant` suppresses the constant term from the variance function. This option may not be specified with `distribution(tnormal)`.

`vhet(varlist [, noconstant])` specifies that the idiosyncratic error component is heteroskedastic, with the variance function depending on a linear combination of *varlist*<sub>*v*</sub>. Specifying `noconstant` suppresses the constant term from the variance function. This option may not be specified with `distribution(tnormal)`.

`cost` specifies that `frontier` fit a cost frontier model.

## SE

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

## Reporting

`level(#)`; see [R] [estimation options](#).

`nocnsreport`; see [R] [estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

## Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [R] [maximize](#). These options are seldom used.

Setting the optimization type to `technique(bhhh)` resets the default `vcetype` to `vce(opg)`.

The following option is available with `frontier` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

Stochastic production frontier models were introduced by [Aigner, Lovell, and Schmidt \(1977\)](#) and [Meeusen and van den Broeck \(1977\)](#). Since then, stochastic frontier models have become a popular subfield in econometrics. [Kumbhakar and Lovell \(2000\)](#) provide a good introduction.

`frontier` fits three stochastic frontier models with distinct parameterizations of the inefficiency term and can fit stochastic production or cost frontier models.

Let's review the nature of the stochastic frontier problem. Suppose that a producer has a production function  $f(\mathbf{z}_i, \beta)$ . In a world without error or inefficiency, the  $i$ th firm would produce

$$q_i = f(\mathbf{z}_i, \beta)$$

Stochastic frontier analysis assumes that each firm potentially produces less than it might due to a degree of inefficiency. Specifically,

$$q_i = f(\mathbf{z}_i, \beta)\xi_i$$

where  $\xi_i$  is the level of efficiency for firm  $i$ ;  $\xi_i$  must be in the interval  $(0, 1]$ . If  $\xi_i = 1$ , the firm is achieving the optimal output with the technology embodied in the production function  $f(\mathbf{z}_i, \beta)$ . When  $\xi_i < 1$ , the firm is not making the most of the inputs  $\mathbf{z}_i$  given the technology embodied in the production function  $f(\mathbf{z}_i, \beta)$ . Because the output is assumed to be strictly positive (i.e.,  $q_i > 0$ ), the degree of technical efficiency is assumed to be strictly positive (i.e.,  $\xi_i > 0$ ).

Output is also assumed to be subject to random shocks, implying that

$$q_i = f(\mathbf{z}_i, \beta)\xi_i \exp(v_i)$$

Taking the natural log of both sides yields

$$\ln(q_i) = \ln\{f(\mathbf{z}_i, \beta)\} + \ln(\xi_i) + v_i$$

Assuming that there are  $k$  inputs and that the production function is linear in logs, defining  $u_i = -\ln(\xi_i)$  yields

$$\ln(q_i) = \beta_0 + \sum_{j=1}^k \beta_j \ln(z_{ji}) + v_i - u_i \quad (1)$$

Because  $u_i$  is subtracted from  $\ln(q_i)$ , restricting  $u_i \geq 0$  implies that  $0 < \xi_i \leq 1$ , as specified above.

Kumbhakar and Lovell (2000) provide a detailed version of the above derivation, and they show that performing an analogous derivation in the dual cost function problem allows us to specify the problem as

$$\ln(c_i) = \beta_0 + \beta_q \ln(q_i) + \sum_{j=1}^k \beta_j \ln(p_{ji}) + v_i + u_i \quad (2)$$

where  $q_i$  is output,  $z_{ji}$  are input quantities,  $c_i$  is cost, and the  $p_{ji}$  are input prices.

Intuitively, the inefficiency effect is required to lower output or raise expenditure, depending on the specification.

## □ Technical note

The model that `frontier` actually fits is of the form

$$y_i = \beta_0 + \sum_{j=1}^k \beta_j x_{ji} + v_i - s u_i$$

where

$$s = \begin{cases} 1, & \text{for production functions} \\ -1, & \text{for cost functions} \end{cases}$$

so, in the context of the discussion above,  $y_i = \ln(q_i)$ , and  $x_{ji} = \ln(z_{ji})$  for a production function; and for a cost function,  $y_i = \ln(c_i)$ , and the  $x_{ji}$  are the  $\ln(p_{ji})$  and  $\ln(q_i)$ . You must take the natural logarithm of the data before fitting a stochastic frontier production or cost model. `frontier` performs no transformations on the data.

□

Different specifications of the  $u_i$  and the  $v_i$  terms give rise to distinct models. `frontier` provides estimators for the parameters of three basic models in which the idiosyncratic component,  $v_i$ , is assumed to be independently  $N(0, \sigma_v)$  distributed over the observations. The basic models differ in their specification of the inefficiency term,  $u_i$ , as follows:

**exponential**: the  $u_i$  are independently exponentially distributed with variance  $\sigma_u^2$

**hnormal**: the  $u_i$  are independently half-normally  $N^+(0, \sigma_u^2)$  distributed

**tnormal**: the  $u_i$  are independently  $N^+(\mu, \sigma_u^2)$  distributed with truncation point at 0

For half-normal or exponential distributions, `frontier` can fit models with heteroskedastic error components, conditional on a set of covariates. For a truncated-normal distribution, `frontier` can also fit a conditional mean model in which the mean is modeled as a linear function of a set of covariates.

## ► Example 1

For our first example, we demonstrate the half-normal and exponential models by reproducing a study found in [Greene \(2008\)](#), which uses data originally published in [Zellner and Revankar \(1969\)](#). In this study of the transportation-equipment manufacturing industry, observations on value added, capital, and labor are used to estimate a Cobb–Douglas production function. The variable `lnv` is the log-transformed value added, `lnk` is the log-transformed capital, and `lnl` is the log-transformed labor. OLS estimates are compared with those from stochastic frontier models using both the half-normal and exponential distribution for the inefficiency term.

```
. use http://www.stata-press.com/data/r11/greene9
```

```
. regress lnv lnk lnl
```

| Source   | SS         | df | MS         | Number of obs = 25 |        |
|----------|------------|----|------------|--------------------|--------|
| Model    | 44.1727741 | 2  | 22.086387  | F( 2, 22) =        | 397.54 |
| Residual | 1.22225984 | 22 | .055557265 | Prob > F =         | 0.0000 |
|          |            |    |            | R-squared =        | 0.9731 |
|          |            |    |            | Adj R-squared =    | 0.9706 |
| Total    | 45.3950339 | 24 | 1.89145975 | Root MSE =         | .23571 |

| lnv   | Coef.    | Std. Err. | t    | P> t  | [95% Conf. Interval] |          |
|-------|----------|-----------|------|-------|----------------------|----------|
| lnk   | .2454281 | .1068574  | 2.30 | 0.032 | .0238193             | .4670368 |
| lnl   | .805183  | .1263336  | 6.37 | 0.000 | .5431831             | 1.067183 |
| _cons | 1.844416 | .2335928  | 7.90 | 0.000 | 1.359974             | 2.328858 |

```
. frontier lnv lnk lnl
```

```
Iteration 0: log likelihood = 2.3357572
```

```
Iteration 1: log likelihood = 2.4673009
```

```
Iteration 2: log likelihood = 2.4695125
```

```
Iteration 3: log likelihood = 2.4695222
```

```
Iteration 4: log likelihood = 2.4695222
```

```
Stoc. frontier normal/half-normal model
```

```
Number of obs = 25
```

```
Wald chi2(2) = 743.71
```

```
Log likelihood = 2.4695222
```

```
Prob > chi2 = 0.0000
```

| lnv      | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| lnk      | .2585478  | .098764   | 2.62  | 0.009 | .0649738             | .4521218  |
| lnl      | .7802451  | .1199399  | 6.51  | 0.000 | .5451672             | 1.015323  |
| _cons    | 2.081135  | .281641   | 7.39  | 0.000 | 1.529128             | 2.633141  |
| /lnsig2v | -3.48401  | .6195353  | -5.62 | 0.000 | -4.698277            | -2.269743 |
| /lnsig2u | -3.014599 | 1.11694   | -2.70 | 0.007 | -5.203761            | -.8254368 |
| sigma_v  | .1751688  | .0542616  |       |       | .0954514             | .3214633  |
| sigma_u  | .2215073  | .1237052  |       |       | .074134              | .6618486  |
| sigma2   | .0797496  | .0426989  |       |       | -.0039388            | .163438   |
| lambda   | 1.264536  | .1678684  |       |       | .9355204             | 1.593552  |

```
Likelihood-ratio test of sigma_u=0: chibar2(01) = 0.43 Prob>=chibar2 = 0.256
```

```
. predict double u_h, u
```

```
. frontier lnv lnk ln1, distribution(exponential)
Iteration 0: log likelihood = 2.7270659
Iteration 1: log likelihood = 2.8551532
Iteration 2: log likelihood = 2.8604815
Iteration 3: log likelihood = 2.8604897
Iteration 4: log likelihood = 2.8604897

Stoc. frontier normal/exponential model Number of obs = 25
Wald chi2(2) = 845.68
Log likelihood = 2.8604897 Prob > chi2 = 0.0000
```

| lnv      | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| lnk      | .2624859  | .0919988  | 2.85  | 0.004 | .0821717             | .4428002  |
| ln1      | .7703795  | .1109569  | 6.94  | 0.000 | .5529079             | .9878511  |
| _cons    | 2.069242  | .2356159  | 8.78  | 0.000 | 1.607444             | 2.531041  |
| /lnsig2v | -3.527598 | .4486176  | -7.86 | 0.000 | -4.406873            | -2.648324 |
| /lnsig2u | -4.002457 | .9274575  | -4.32 | 0.000 | -5.820241            | -2.184674 |
| sigma_v  | .1713925  | .0384448  |       |       | .1104231             | .2660258  |
| sigma_u  | .1351691  | .0626818  |       |       | .0544692             | .3354317  |
| sigma2   | .0476461  | .0157921  |       |       | .016694              | .0785981  |
| lambda   | .7886525  | .087684   |       |       | .616795              | .9605101  |

Likelihood-ratio test of sigma\_u=0: chibar2(01) = 1.21    Prob>=chibar2 = 0.135

```
. predict double u_e, u
```

```
. list state u_h u_e
```

|     | state         | u_h       | u_e       |
|-----|---------------|-----------|-----------|
| 1.  | Alabama       | .2011338  | .14592865 |
| 2.  | California    | .14480966 | .0972165  |
| 3.  | Connecticut   | .1903485  | .13478797 |
| 4.  | Florida       | .51753139 | .5903303  |
| 5.  | Georgia       | .10397912 | .07140994 |
| 6.  | Illinois      | .12126696 | .0830415  |
| 7.  | Indiana       | .21128212 | .15450664 |
| 8.  | Iowa          | .24933153 | .20073081 |
| 9.  | Kansas        | .10099517 | .06857629 |
| 10. | Kentucky      | .05626919 | .04152443 |
| 11. | Louisiana     | .20332731 | .15066405 |
| 12. | Maine         | .22263164 | .17245793 |
| 13. | Maryland      | .13534062 | .09245501 |
| 14. | Massachusetts | .15636999 | .10932923 |
| 15. | Michigan      | .15809566 | .10756915 |
| 16. | Missouri      | .10288047 | .0704146  |
| 17. | NewJersey     | .09584337 | .06587986 |
| 18. | NewYork       | .27787793 | .22249416 |
| 19. | Ohio          | .22914231 | .16981857 |
| 20. | Pennsylvania  | .1500667  | .10302905 |
| 21. | Texas         | .20297875 | .14552218 |
| 22. | Virginia      | .14000132 | .09676078 |
| 23. | Washington    | .11047581 | .07533251 |
| 24. | WestVirginia  | .15561392 | .11236153 |
| 25. | Wisconsin     | .14067066 | .0970861  |

The parameter estimates and the estimates of the inefficiency terms closely match those published in [Greene \(2008\)](#), but the standard errors of the parameter estimates are estimated differently (see the technical note below).

The output from `frontier` includes estimates of the standard deviations of the two error components,  $\sigma_v$  and  $\sigma_u$ , which are labeled `sigma_v` and `sigma_u`, respectively. In the log likelihood, they are parameterized as  $\ln\sigma_v^2$  and  $\ln\sigma_u^2$ , and these estimates are labeled `/lnsig2v` and `/lnsig2u` in the output. `frontier` also reports two other useful parameterizations. The estimate of the total error variance,  $\sigma_S^2 = \sigma_v^2 + \sigma_u^2$ , is labeled `sigma2`, and the estimate of the ratio of the standard deviation of the inefficiency component to the standard deviation of the idiosyncratic component,  $\lambda = \sigma_u/\sigma_v$ , is labeled `lambda`.

At the bottom of the output, `frontier` reports the results of a test that there is no technical inefficiency component in the model. This is a test of the null hypothesis  $H_0 : \sigma_u^2 = 0$  against the alternative hypotheses  $H_1 : \sigma_u^2 > 0$ . If the null hypothesis is true, the stochastic frontier model reduces to an OLS model with normal errors. However, because the test lies on the boundary of the parameter space of  $\sigma_u^2$ , the standard likelihood-ratio test is not valid, and a one-sided generalized likelihood-ratio test must be constructed; see [Gutierrez, Carter, and Drukker \(2001\)](#). For this example, the output shows LR = 0.43 with a  $p$ -value of 0.256 for the half-normal model and LR = 1.21 with a  $p$ -value of 0.135 for the exponential model. There are several possible reasons for the failure to reject the null hypothesis, but the fact that the test is based on an asymptotic distribution and the sample size was 25 is certainly a leading candidate among those possibilities.

◀

## □ Technical note

`frontier` maximizes the log-likelihood function of a stochastic frontier model by using the Newton–Raphson method, and the estimated variance–covariance matrix is calculated as the inverse of the negative Hessian (matrix of second partial derivatives); see [\[R\] ml](#). When comparing the results with those published using other software, be aware of the difference in the optimization methods, which may result in different, yet asymptotically equivalent, variance estimates.

□

## ▷ Example 2

Often the error terms may not have constant variance. `frontier` allows you to model heteroskedasticity in either error term as a linear function of a set of covariates. The variance of either the technical inefficiency or the idiosyncratic component may be modeled as

$$\sigma_i^2 = \exp(\mathbf{w}_i\boldsymbol{\delta})$$

The default constant included in  $\mathbf{w}_i$  may be suppressed by appending a `noconstant` option to the list of covariates. Also, you can simultaneously specify covariates for both  $\sigma_{u_i}$  and  $\sigma_{v_i}$ .

In the example below, we use a sample of 756 observations of fictional firms producing a manufactured good by using capital and labor. The firms are hypothesized to use a constant returns-to-scale technology, but the sizes of the firms differ. Believing that this size variation will introduce heteroskedasticity into the idiosyncratic error term, we estimate the parameters of a Cobb–Douglas production function. To do this, we use a conditional heteroskedastic half-normal model, with the size of the firm as an explanatory variable in the variance function for the idiosyncratic error. We also perform a test of the hypothesis that the firms use a constant returns-to-scale technology.

```
. use http://www.stata-press.com/data/r11/frontier1, clear
. frontier lnoutput lnlabor lncapital, vhet(size)

Iteration 0: log likelihood = -1508.3692
Iteration 1: log likelihood = -1501.583
Iteration 2: log likelihood = -1500.3942
Iteration 3: log likelihood = -1500.3794
Iteration 4: log likelihood = -1500.3794

Stoc. frontier normal/half-normal model Number of obs = 756
 Wald chi2(2) = 9.68
Log likelihood = -1500.3794 Prob > chi2 = 0.0079
```

| lnoutput  | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-----------|-----------|-----------|-------|-------|----------------------|-----------|
| lnoutput  |           |           |       |       |                      |           |
| lnlabor   | .7090933  | .2349374  | 3.02  | 0.003 | .2486244             | 1.169562  |
| lncapital | .3931345  | .5422173  | 0.73  | 0.468 | -.6695919            | 1.455861  |
| _cons     | 1.252199  | 3.14656   | 0.40  | 0.691 | -4.914946            | 7.419344  |
| lnsig2v   |           |           |       |       |                      |           |
| size      | -.0016951 | .0004748  | -3.57 | 0.000 | -.0026256            | -.0007645 |
| _cons     | 3.156091  | .9265826  | 3.41  | 0.001 | 1.340023             | 4.97216   |
| lnsig2u   |           |           |       |       |                      |           |
| _cons     | 1.947487  | .1017653  | 19.14 | 0.000 | 1.748031             | 2.146943  |
| sigma_u   | 2.647838  | .134729   |       |       | 2.396514             | 2.925518  |

```
. test _b[lnlabor] + _b[lncapital] = 1
(1) [lnoutput]lnlabor + [lnoutput]lncapital = 1
 chi2(1) = 0.03
 Prob > chi2 = 0.8622
```

The output above indicates that the variance of the idiosyncratic error term is a function of firm size. Also, we failed to reject the hypothesis that the firms use a constant returns-to-scale technology.

❑ Technical note

In small samples, the conditional heteroskedastic estimators will lack precision for the variance parameters and may fail to converge altogether.

➤ Example 3

Let’s turn our attention to the truncated-normal model. Once again, we will use fictional data. For this example, we have 1,231 observations on the quantity of output, the total cost of production for each firm, the prices that each firm paid for labor and capital services, and a categorical variable measuring the quality of each firm’s management. After taking the natural logarithm of the costs (lncost), prices (lnp\_k and lnp\_l), and output (lnout), we fit a stochastic cost frontier model and specify the distribution for the inefficiency term to be truncated normal.



```
. use http://www.stata-press.com/data/r11/frontier2
. frontier lncost lnp_k lnp_l lnout, distribution(tnormal) cost

Iteration 0: log likelihood = -2386.9523
Iteration 1: log likelihood = -2386.5146
Iteration 2: log likelihood = -2386.2704
Iteration 3: log likelihood = -2386.2504
Iteration 4: log likelihood = -2386.2493
Iteration 5: log likelihood = -2386.2493

Stoc. frontier normal/truncated-normal model Number of obs = 1231
 Wald chi2(3) = 8.82
Log likelihood = -2386.2493 Prob > chi2 = 0.0318
```

| lncost     | Coef.    | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|------------|----------|-----------|------|-------|----------------------|----------|
| lnp_k      | .3410717 | .2363861  | 1.44 | 0.149 | -.1222366            | .80438   |
| lnp_l      | .6608628 | .4951499  | 1.33 | 0.182 | -.3096131            | 1.631339 |
| lnout      | .7528653 | .3468968  | 2.17 | 0.030 | .0729601             | 1.432771 |
| _cons      | 2.602609 | 1.083004  | 2.40 | 0.016 | .4799595             | 4.725259 |
| /mu        | 1.095705 | .881517   | 1.24 | 0.214 | -.632037             | 2.823446 |
| /lnsigma2  | 1.5534   | .1873464  | 8.29 | 0.000 | 1.186208             | 1.920592 |
| /ilgtgamma | 1.257862 | .2589522  | 4.86 | 0.000 | .7503255             | 1.765399 |
| sigma2     | 4.727518 | .8856833  |      |       | 3.274641             | 6.825001 |
| gamma      | .7786579 | .0446303  |      |       | .6792496             | .8538846 |
| sigma_u2   | 3.681119 | .7503408  |      |       | 2.210478             | 5.15176  |
| sigma_v2   | 1.046399 | .2660035  |      |       | .5250413             | 1.567756 |

H0: No inefficiency component:                      z =    5.595                      Prob>=z = 0.000

In addition to the coefficients, the output reports estimates for several parameters. `sigma_v2` is the estimate of  $\sigma_v^2$ . `sigma_u2` is the estimate of  $\sigma_u^2$ . `gamma` is the estimate of  $\gamma = \sigma_u^2 / \sigma_S^2$ . `sigma2` is the estimate of  $\sigma_S^2 = \sigma_v^2 + \sigma_u^2$ . Because  $\gamma$  must be between 0 and 1, the optimization is parameterized in terms of the inverse logit of  $\gamma$ , and this estimate is reported as `ilgtgamma`. Because  $\sigma_S^2$  must be positive, the optimization is parameterized in terms of  $\ln(\sigma_S^2)$ , whose estimate is reported as `lnsigma2`. Finally, `mu` is the estimate of  $\mu$ , the mean of the truncated-normal distribution.

In the output above, the generalized log-likelihood test for the presence of the inefficiency term has been replaced with a test based on the third moment of the OLS residuals. When  $\mu = 0$  and  $\sigma_u = 0$ , the truncated-normal model reduces to a linear regression model with normally distributed errors. However, the distribution of the test statistic under the null hypothesis is not well established, because it becomes impossible to evaluate the log likelihood as  $\sigma_u$  approaches zero, prohibiting the use of the likelihood-ratio test.

However, [Coelli \(1995\)](#) noted that the presence of an inefficiency term would negatively skew the residuals from an OLS regression. By identifying negative skewness in the residuals with the presence of an inefficiency term, Coelli derived a one-sided test for the presence of the inefficiency term. The results of this test are given at the bottom of the output. For this example, the null hypothesis of no inefficiency component is rejected.

In the example below, we fit a truncated model and detect a statistically significant inefficiency term in the model. We might question whether the inefficiency term is identically distributed over all firms or whether there might be heterogeneity across firms. `frontier` provides an extension to the truncated normal model by allowing the mean of the inefficiency term to be modeled as a linear function of a set of covariates. In our dataset, we have a categorical variable that measures the quality of a firm's management. We refit the model, including the `cm()` option, specifying a set of

binary indicator variables representing the different categories of the quality-measurement variable as covariates.

```
. frontier lncost lnp_k lnp_l lnout, distribution(tnormal) cm(i.quality) cost
Iteration 0: log likelihood = -2386.9523
Iteration 1: log likelihood = -2384.936
Iteration 2: log likelihood = -2382.3942
Iteration 3: log likelihood = -2382.324
Iteration 4: log likelihood = -2382.3233
Iteration 5: log likelihood = -2382.3233

Stoc. frontier normal/truncated-normal model Number of obs = 1231
 Wald chi2(3) = 9.31
Log likelihood = -2382.3233 Prob > chi2 = 0.0254
```

| lncost     | Coef.    | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|------------|----------|-----------|------|-------|----------------------|----------|
| lncost     |          |           |      |       |                      |          |
| lnp_k      | .3611204 | .2359749  | 1.53 | 0.126 | -.1013819            | .8236227 |
| lnp_l      | .680446  | .4934935  | 1.38 | 0.168 | -.2867835            | 1.647675 |
| lnout      | .7605533 | .3466102  | 2.19 | 0.028 | .0812098             | 1.439897 |
| _cons      | 2.550769 | 1.078911  | 2.36 | 0.018 | .4361417             | 4.665396 |
| mu         |          |           |      |       |                      |          |
| quality    |          |           |      |       |                      |          |
| 2          | .5056067 | .3382907  | 1.49 | 0.135 | -.1574309            | 1.168644 |
| 3          | .783223  | .376807   | 2.08 | 0.038 | .0446947             | 1.521751 |
| 4          | .5577511 | .3355061  | 1.66 | 0.096 | -.0998288            | 1.215331 |
| 5          | .6792882 | .3428073  | 1.98 | 0.048 | .0073981             | 1.351178 |
| _cons      | .6014025 | .990167   | 0.61 | 0.544 | -1.339289            | 2.542094 |
| /lnsigma2  | 1.541784 | .1790926  | 8.61 | 0.000 | 1.190769             | 1.892799 |
| /ilgtgamma | 1.242302 | .2588968  | 4.80 | 0.000 | .734874              | 1.749731 |
| sigma2     | 4.67292  | .8368852  |      |       | 3.289611             | 6.637923 |
| gamma      | .7759645 | .0450075  |      |       | .6758739             | .8519189 |
| sigma_u2   | 3.62602  | .7139576  |      |       | 2.226689             | 5.025351 |
| sigma_v2   | 1.0469   | .2583469  |      |       | .5405491             | 1.553251 |

The conditional mean model was developed in the context of panel-data estimators, and we can apply frontier’s conditional mean model to panel data.



## Saved results

`frontier` saves the following in `e()`:

### Scalars

|                            |                                                |
|----------------------------|------------------------------------------------|
| <code>e(N)</code>          | number of observations                         |
| <code>e(df_m)</code>       | model degrees of freedom                       |
| <code>e(k)</code>          | number of parameters                           |
| <code>e(k_eq)</code>       | number of equations                            |
| <code>e(k_eq_model)</code> | number of equations in model Wald test         |
| <code>e(k_dv)</code>       | number of dependent variables                  |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints |
| <code>e(chi2)</code>       | $\chi^2$                                       |
| <code>e(ll)</code>         | log likelihood                                 |
| <code>e(ll_c)</code>       | log likelihood for $H_0: \sigma_u=0$           |
| <code>e(z)</code>          | test for negative skewness of OLS residuals    |
| <code>e(sigma_u)</code>    | standard deviation of technical inefficiency   |
| <code>e(sigma_v)</code>    | standard deviation of $v_i$                    |
| <code>e(p)</code>          | significance                                   |
| <code>e(chi2_c)</code>     | LR test statistic                              |
| <code>e(p_z)</code>        | $p$ -value for $z$                             |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                      |
| <code>e(ic)</code>         | number of iterations                           |
| <code>e(rc)</code>         | return code                                    |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                    |

### Macros

|                                 |                                                                          |
|---------------------------------|--------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>frontier</code>                                                    |
| <code>e(cmdline)</code>         | command as typed                                                         |
| <code>e(depvar)</code>          | name of dependent variable                                               |
| <code>e(function)</code>        | production or cost                                                       |
| <code>e(wtype)</code>           | weight type                                                              |
| <code>e(wexp)</code>            | weight expression                                                        |
| <code>e(title)</code>           | title in estimation output                                               |
| <code>e(chi2type)</code>        | Wald; type of model $\chi^2$ test                                        |
| <code>e(dist)</code>            | distribution assumption for $u_i$                                        |
| <code>e(het)</code>             | heteroskedastic components                                               |
| <code>e(u_hetvar)</code>        | <i>varlist</i> in <code>uhet()</code>                                    |
| <code>e(v_hetvar)</code>        | <i>varlist</i> in <code>vhet()</code>                                    |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                           |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                            |
| <code>e(opt)</code>             | type of optimization                                                     |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>       | type of ml method                                                        |
| <code>e(user)</code>            | name of likelihood-evaluator program                                     |
| <code>e(technique)</code>       | maximization technique                                                   |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular              |
| <code>e(crittype)</code>        | optimization criterion                                                   |
| <code>e(properties)</code>      | b V                                                                      |
| <code>e(predict)</code>         | program used to implement predict                                        |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>           |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>           |

|                 |                                              |
|-----------------|----------------------------------------------|
| Matrices        |                                              |
| e(b)            | coefficient vector                           |
| e(Cns)          | constraints matrix                           |
| e(ilog)         | iteration log (up to 20 iterations)          |
| e(gradient)     | gradient vector                              |
| e(V)            | variance–covariance matrix of the estimators |
| e(V_modelbased) | model-based variance                         |
| Functions       |                                              |
| e(sample)       | marks estimation sample                      |

## Methods and formulas

frontier is implemented as an ado-file.

Consider an equation of the form

$$y_i = \mathbf{x}_i\boldsymbol{\beta} + v_i - su_i$$

where  $y_i$  is the dependent variable,  $\mathbf{x}_i$  is a  $1 \times k$  vector of observations on the independent variables included as indent covariates,  $\boldsymbol{\beta}$  is a  $k \times 1$  vector of coefficients, and

$$s = \begin{cases} 1, & \text{for production functions} \\ -1, & \text{for cost functions} \end{cases}$$

The log-likelihood functions are as follows.

Normal/half-normal model:

$$\ln L = \sum_{i=1}^N \left\{ \frac{1}{2} \ln \left( \frac{2}{\pi} \right) - \ln \sigma_S + \ln \Phi \left( -\frac{s\epsilon_i \lambda}{\sigma_S} \right) - \frac{\epsilon_i^2}{2\sigma_S^2} \right\}$$

Normal/exponential model:

$$\ln L = \sum_{i=1}^N \left\{ -\ln \sigma_u + \frac{\sigma_v^2}{2\sigma_u^2} + \ln \Phi \left( \frac{-s\epsilon_i - \frac{\sigma_v^2}{\sigma_u}}{\sigma_v} \right) + \frac{s\epsilon_i}{\sigma_u} \right\}$$

Normal/truncated-normal model:

$$\begin{aligned} \ln L = \sum_{i=1}^N \bigg\{ & -\frac{1}{2} \ln (2\pi) - \ln \sigma_S - \ln \Phi \left( \frac{\mu}{\sigma_S \sqrt{\gamma}} \right) \\ & + \ln \Phi \left[ \frac{(1-\gamma) \mu - s\gamma \epsilon_i}{\{\sigma_S^2 \gamma (1-\gamma)\}^{1/2}} \right] - \frac{1}{2} \left( \frac{\epsilon_i + s\mu}{\sigma_S} \right)^2 \bigg\} \end{aligned}$$

where  $\sigma_S = (\sigma_u^2 + \sigma_v^2)^{1/2}$ ,  $\lambda = \sigma_u/\sigma_v$ ,  $\gamma = \sigma_u^2/\sigma_S^2$ ,  $\epsilon_i = y_i - \mathbf{x}_i\beta$ , and  $\Phi()$  is the cumulative distribution function of the standard normal distribution.

To obtain estimation for  $u_i$ , you can use either the mean or the mode of the conditional distribution  $f(u|\epsilon)$ .

$$E(u_i | \epsilon_i) = \mu_{*i} + \sigma_* \left\{ \frac{\phi(-\mu_{*i}/\sigma_*)}{\Phi(\mu_{*i}/\sigma_*)} \right\}$$

$$M(u_i | \epsilon_i) = \begin{cases} \mu_{*i}, & \text{if } \mu_{*i} \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Then the technical efficiency ( $s = 1$ ) or cost efficiency ( $s = -1$ ) will be estimated by

$$E_i = E \{ \exp(-su_i) | \epsilon_i \}$$

$$= \left\{ \frac{1 - \Phi(s\sigma_* - \mu_{*i}/\sigma_*)}{1 - \Phi(-\mu_{*i}/\sigma_*)} \right\} \exp \left( -s\mu_{*i} + \frac{1}{2}\sigma_*^2 \right)$$

where  $\mu_{*i}$  and  $\sigma_*$  are defined for the normal/half-normal model as

$$\mu_{*i} = -s\epsilon_i\sigma_u^2/\sigma_S^2$$

$$\sigma_* = \sigma_u\sigma_v/\sigma_S$$

for the normal/exponential model as

$$\mu_{*i} = -s\epsilon_i - \sigma_v^2/\sigma_u$$

$$\sigma_* = \sigma_v$$

and for the normal/truncated-normal model as

$$\mu_{*i} = \frac{-s\epsilon_i\sigma_u^2 + \mu\sigma_v^2}{\sigma_S^2}$$

$$\sigma_* = \sigma_u\sigma_v/\sigma_S$$

In the half-normal and exponential models, when heteroskedasticity is assumed, the standard deviations,  $\sigma_u$  or  $\sigma_v$ , will be replaced in the above equations by

$$\sigma_i^2 = \exp(\mathbf{w}_i\boldsymbol{\delta})$$

where  $\mathbf{w}$  is the vector of explanatory variables in the variance function.

In the conditional mean model, the mean parameter of the truncated normal distribution,  $\mu$ , is modeled as a linear combination of the set of covariates,  $\mathbf{w}$ .

$$\mu = \mathbf{w}_i\boldsymbol{\delta}$$

Therefore, the log-likelihood function can be rewritten as

$$\ln L = \sum_{i=1}^N \left[ -\frac{1}{2} \ln(2\pi) - \ln \sigma_S - \ln \Phi \left( \frac{\mathbf{w}_i \boldsymbol{\delta}}{\sqrt{\sigma_S^2 \gamma}} \right) + \ln \Phi \left\{ \frac{(1-\gamma) \mathbf{w}_i \boldsymbol{\delta} - s \gamma \epsilon_i}{\sqrt{\sigma_S^2 \gamma (1-\gamma)}} \right\} - \frac{1}{2} \left( \frac{\epsilon_i + s \mathbf{w}_i \boldsymbol{\delta}}{\sigma_S} \right)^2 \right]$$

The  $z$  test reported in the output of the truncated-normal model is a third-moment test developed by Coelli (1995) as an extension of a test previously developed by Pagan and Hall (1983). Coelli shows that under the null of normally distributed errors, the statistic

$$z = \frac{m_3}{\left( \frac{6m_2^3}{N} \right)^{1/2}}$$

has a standard normal distribution, where  $m_3$  is the third moment from the OLS regression. Because the residuals are either negatively skewed (production function) or positively skewed (cost function), a one-sided  $p$ -value is used.

## References

- Aigner, D., C. A. K. Lovell, and P. Schmidt. 1977. Formulation and estimation of stochastic frontier production function models. *Journal of Econometrics* 6: 21–37.
- Caudill, S. B., J. M. Ford, and D. M. Gropner. 1995. Frontier estimation and firm-specific inefficiency measures in the presence of heteroscedasticity. *Journal of Business and Economic Statistics* 13: 105–111.
- Coelli, T. J. 1995. Estimators and hypothesis tests for a stochastic frontier function: A Monte Carlo analysis. *Journal of Productivity Analysis* 6: 247–268.
- Gould, W. W., J. S. Pitblado, and W. M. Sribney. 2006. *Maximum Likelihood Estimation with Stata*. 3rd ed. College Station, TX: Stata Press.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- Gutierrez, R. G., S. Carter, and D. M. Drukker. 2001. [sg160: On boundary-value likelihood-ratio tests](#). *Stata Technical Bulletin* 60: 15–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 269–273. College Station, TX: Stata Press.
- Kumbhakar, S. C., and C. A. K. Lovell. 2000. *Stochastic Frontier Analysis*. Cambridge: Cambridge University Press.
- Meeusen, W., and J. van den Broeck. 1977. Efficiency estimation from Cobb–Douglas production functions with composed error. *International Economic Review* 18: 435–444.
- Pagan, A. R., and A. D. Hall. 1983. Diagnostic tests as residual analysis. *Econometric Reviews* 2: 159–218.
- Petrin, A., B. P. Poi, and J. Levinsohn. 2004. [Production function estimation in Stata using inputs to control for unobservables](#). *Stata Journal* 4: 113–123.
- Stevenson, R. E. 1980. Likelihood functions for generalized stochastic frontier estimation. *Journal of Econometrics* 13: 57–66.
- Zellner, A., and N. S. Revankar. 1969. Generalized production functions. *Review of Economic Studies* 36: 241–250.

## Also see

- [R] [frontier postestimation](#) — Postestimation tools for frontier
- [R] [regress](#) — Linear regression
- [XT] [xtfrontier](#) — Stochastic frontier models for panel data
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `frontier`:

| command                | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>     | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estimates</code> | cataloging estimation results                                                                       |
| <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>  | link test for model specification                                                                   |
| <code>lrtest</code>    | likelihood-ratio test                                                                               |
| <code>margins</code>   | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>   | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code> | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic]

predict [type] { stub*|newvarxb newvarv newvaru } [if] [in] , scores
```

| statistic         | description                                                                                             |
|-------------------|---------------------------------------------------------------------------------------------------------|
| Main              |                                                                                                         |
| <code>xb</code>   | linear prediction; the default                                                                          |
| <code>stdp</code> | standard error of the prediction                                                                        |
| <code>u</code>    | estimates of minus the natural log of the technical efficiency via $E(u_i \mid \epsilon_i)$             |
| <code>m</code>    | estimates of minus the natural log of the technical efficiency via $M(u_i \mid \epsilon_i)$             |
| <code>te</code>   | estimates of the technical efficiency via $E\{\exp(-su_i) \mid \epsilon_i\}$                            |
|                   | $s = \begin{cases} 1, & \text{for production functions} \\ -1, & \text{for cost functions} \end{cases}$ |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

`xb`, the default, calculates the linear prediction.

`stdp` calculates the standard error of the linear prediction.

`u` produces estimates of minus the natural log of the technical efficiency via  $E(u_i \mid \epsilon_i)$ .

`m` produces estimates of minus the natural log of the technical efficiency via  $M(u_i \mid \epsilon_i)$ .

`te` produces estimates of the technical efficiency via  $E\{\exp(-su_i) \mid \epsilon_i\}$ .

`scores` calculates equation-level score variables.

The first new variable will contain  $\partial \ln L / \partial (\mathbf{x}_i \beta)$ .

The second new variable will contain  $\partial \ln L / \partial (\text{lnsig2v})$ .

The third new variable will contain  $\partial \ln L / \partial (\text{lnsig2u})$ .

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [frontier](#) — Stochastic frontier models

[U] [20 Estimation and postestimation commands](#)



# Title

**fvrevar** — Factor-variables operator programming command

## Syntax

```
fvrevar [varlist] [if] [in] [, substitute tsonly list]
```

You must `tsset` your data before using `fvrevar` if *varlist* contains time-series operators; see [TS] `tsset`.

## Description

`fvrevar` creates an equivalent, temporary variable list for a *varlist* that might contain factor variables, interactions, or time-series-operated variables so that the resulting variable list can be used by commands that do not otherwise support factor variables or time-series-operated variables. The resulting list also could be used in a program to speed execution at the cost of using more memory.

## Options

`substitute` specifies that equivalent, temporary variables be substituted for any factor variables, interactions, or time-series-operated variables in *varlist*. `substitute` is the default action taken by `fvrevar`; you do not need to specify the option.

`tsonly` specifies that equivalent, temporary variables be substituted for only the time-series-operated variables in *varlist*.

`list` specifies that all factor-variable operators and time-series operators be removed from *varlist* and the resulting list of base variables be returned in `r(varlist)`. No new variables are created with this option.

## Remarks

`fvrevar` might create no new variables, one new variable, or many new variables, depending on the number of factor variables, interactions, and time-series operators appearing in *varlist*. Any new variables created are temporary. The new, equivalent varlist is returned in `r(varlist)`. The new varlist corresponds one to one with the original *varlist*.

### ► Example 1

Typing

```
. use http://www.stata-press.com/data/r11/auto
. fvrevar i.rep78 mpg turn
```

creates five temporary variables corresponding to the levels of `rep78`. No new variables are created for variables `mpg` and `turn` because they do not contain factor-variable or time-series operators.

The resulting variable list is

```
. display "r(varlist)"
__000000 __000001 __000002 __000003 __000004 mpg turn
```

(Your temporary variable names may be different, but that is of no consequence.)

Temporary variables automatically vanish when the program concludes.

◀

► Example 2

Suppose we want to create temporary variables for specific levels of a factor variable. To do this, we can use the parenthesis notation of factor-variable syntax.

```
. fvrevar i(2,3)bn.rep78 mpg
```

creates two temporary variables corresponding to levels 2 and 3 of `rep78`. Notice that we specified that neither level 2 nor 3 be set as the base level by using the `bn` notation. If we did not specify `bn`, level 2 would have been treated as the base level.

The resulting variable list is

```
. display "r(varlist)"
__00000E __00000F mpg
```

We can see the results by listing the new variables alongside the original value of `rep78`.

```
. list rep78 'r(varlist)' in 1/5
```

|    | rep78 | __00000E | __00000F | mpg |
|----|-------|----------|----------|-----|
| 1. | 3     | 1        | 0        | 22  |
| 2. | 3     | 1        | 0        | 17  |
| 3. | .     | .        | .        | 22  |
| 4. | 3     | 1        | 0        | 20  |
| 5. | 4     | 0        | 1        | 15  |

If we had needed only the base-variable names, we could have specified

```
. fvrevar i(2,3)bn.rep78 mpg, list
. display "r(varlist)"
mpg rep78
```

The order of the list will probably differ from that of the original list; base variables are listed only once.

◀

► Example 3

Now let's assume we have a *varlist* containing both an interaction and time-series-operated variables. If we want to create temporary variables for the entire equivalent *varlist*, we can specify `fvrevar` with no options.

```
. generate t = _n
. tsset t
. fvrevar c.turn#i(2,3).rep78 L.mpg
```

The resulting variable list is

```
. display "r(varlist)"
__00000I __00000K __00000M
```

If we want to create temporary variables only for the time-series-operated variables, we can specify the `tsonly` option.

```
. fvrevar c.turn#i(2,3).rep78 L.mpg, tsonly
```

The resulting variable list is

```
. display "r(varlist)"
2b.rep78#c.turn 3.rep78#c.turn __00000M
```

Notice that `fvrevar` returned the expanded factor-variable list with the `tsonly` option.



## □ Technical note

`fvrevar`, `substitute` avoids creating duplicate variables. Consider

```
. fvrevar i.rep78 turn mpg i.rep78
```

`i.rep78` appears twice in the varlist. `fvrevar` will create only one set of new variables for the five levels of `rep78` and will use these new variables once in the resulting `r(varlist)`. Moreover, `fvrevar` will do this even across multiple calls:

```
. fvrevar i.rep78 turn mpg
. fvrevar i.rep78
```

`i.rep78` appears in two separate calls. At the first call, `fvrevar` creates five temporary variables corresponding to the five levels of `rep78`. At the second call, `fvrevar` remembers what it has done and uses the same temporary variables for `i.rep78`.



## Saved results

`fvrevar` saves the following in `r()`:

Macros

`r(varlist)` the modified variable list or list of base-variable names

## Also see

[P] [syntax](#) — Parse Stata syntax

[TS] [tsrevar](#) — Time-series operator programming command

[P] [unab](#) — Unabbreviate variable list

[U] [11 Language syntax](#)

[U] [11.4.4 Time-series varlists](#)

[U] [18 Programming Stata](#)

Syntax

Declare base settings

```
fvset base base_spec varlist
```

Declare design settings

```
fvset design design_spec varlist
```

Clear the current settings

```
fvset clear varlist
```

Report the current settings

```
fvset report [varlist] [, base(base_spec) design(design_spec)]
```

| <i>base_spec</i>       | description               |
|------------------------|---------------------------|
| <b>default</b>         | default base              |
| <b><u>first</u></b>    | lowest level value        |
| <b><u>last</u></b>     | highest level value       |
| <b><u>frequent</u></b> | most frequent level value |
| <b><u>none</u></b>     | no base                   |
| <b>#</b>               | nonnegative integer value |

| <i>design_spec</i>       | description                                     |
|--------------------------|-------------------------------------------------|
| <b>default</b>           | default base                                    |
| <b><u>asbalanced</u></b> | accumulate using $1/k$ , $k$ = number of levels |
| <b><u>asobserved</u></b> | accumulate using observed relative frequencies  |

Description

**fvset** declares factor-variable settings. Factor-variable settings identify the base level and how to accumulate statistics over levels.

**fvset base** specifies the base level for each variable in *varlist*. The default for factor variables without a declared base level is **first**.

**fvset design** specifies how to accumulate over the levels of a factor variable. The **margins** command is the only command aware of this setting; see [R] **margins**. By default, **margins** assumes that factor variables are **asobserved**, meaning that they are accumulated by weighting by the number of observations or the sum of the weights if weights have been specified.

`fvset clear` removes factor-variable settings for each variable in *varlist*. `fvset clear _all` removes all factor-variable settings from all variables.

`fvset report` reports the current factor-variable settings for each variable in *varlist*. `fvset` without arguments is a synonym for `fvset report`.

## Options

`base(base_spec)` restricts `fvset report` to report only the factor-variable settings for variables with the specified *base\_spec*.

`design(design_spec)` restricts `fvset report` to report only the factor-variable settings for variables with the specified *design\_spec*.

## Remarks

### ► Example 1

Using our `auto` dataset, we include factor variable `i.rep78` in a regression:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
```

```
. regress mpg i.rep78, baselevels
```

| Source   | SS         | df | MS         |                        |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | 549.415777 | 4  | 137.353944 | Number of obs = 69     |  |  |
| Residual | 1790.78712 | 64 | 27.9810488 | F( 4, 64) = 4.91       |  |  |
| Total    | 2340.2029  | 68 | 34.4147485 | Prob > F = 0.0016      |  |  |
|          |            |    |            | R-squared = 0.2348     |  |  |
|          |            |    |            | Adj R-squared = 0.1869 |  |  |
|          |            |    |            | Root MSE = 5.2897      |  |  |

| mpg   | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|-------|-----------|-----------|-------|-------|----------------------|----------|
| rep78 |           |           |       |       |                      |          |
| 1     | (base)    |           |       |       |                      |          |
| 2     | -1.875    | 4.181884  | -0.45 | 0.655 | -10.22927            | 6.479274 |
| 3     | -1.566667 | 3.863059  | -0.41 | 0.686 | -9.284014            | 6.150681 |
| 4     | .6666667  | 3.942718  | 0.17  | 0.866 | -7.209818            | 8.543152 |
| 5     | 6.363636  | 4.066234  | 1.56  | 0.123 | -1.759599            | 14.48687 |
| _cons | 21        | 3.740391  | 5.61  | 0.000 | 13.52771             | 28.47229 |

We specified the `baselevels` option so that the base level would be included in the output. By default, the first level is the base level. We can change the base level to 2:

(Continued on next page)

```
. fvset base 2 rep78
. regress mpg i.rep78, baselevels
```

| Source   | SS         | df | MS         |  |                 |        |
|----------|------------|----|------------|--|-----------------|--------|
| Model    | 549.415777 | 4  | 137.353944 |  | Number of obs = | 69     |
| Residual | 1790.78712 | 64 | 27.9810488 |  | F( 4, 64) =     | 4.91   |
|          |            |    |            |  | Prob > F =      | 0.0016 |
|          |            |    |            |  | R-squared =     | 0.2348 |
|          |            |    |            |  | Adj R-squared = | 0.1869 |
| Total    | 2340.2029  | 68 | 34.4147485 |  | Root MSE =      | 5.2897 |

| mpg   | Coef.    | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|-------|----------|-----------|-------|-------|----------------------|----------|
| rep78 |          |           |       |       |                      |          |
| 1     | 1.875    | 4.181884  | 0.45  | 0.655 | -6.479274            | 10.22927 |
| 2     | (base)   |           |       |       |                      |          |
| 3     | .3083333 | 2.104836  | 0.15  | 0.884 | -3.896559            | 4.513226 |
| 4     | 2.541667 | 2.247695  | 1.13  | 0.262 | -1.948621            | 7.031954 |
| 5     | 8.238636 | 2.457918  | 3.35  | 0.001 | 3.32838              | 13.14889 |
| _cons | 19.125   | 1.870195  | 10.23 | 0.000 | 15.38886             | 22.86114 |

Let’s set rep78 to have no base level and fit a cell-means regression:

```
. fvset base none rep78
. regress mpg i.rep78, noconstant
```

| Source   | SS         | df | MS         |  |                 |        |
|----------|------------|----|------------|--|-----------------|--------|
| Model    | 31824.2129 | 5  | 6364.84258 |  | Number of obs = | 69     |
| Residual | 1790.78712 | 64 | 27.9810488 |  | F( 5, 64) =     | 227.47 |
|          |            |    |            |  | Prob > F =      | 0.0000 |
|          |            |    |            |  | R-squared =     | 0.9467 |
|          |            |    |            |  | Adj R-squared = | 0.9426 |
| Total    | 33615      | 69 | 487.173913 |  | Root MSE =      | 5.2897 |

| mpg   | Coef.    | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|-------|----------|-----------|-------|-------|----------------------|----------|
| rep78 |          |           |       |       |                      |          |
| 1     | 21       | 3.740391  | 5.61  | 0.000 | 13.52771             | 28.47229 |
| 2     | 19.125   | 1.870195  | 10.23 | 0.000 | 15.38886             | 22.86114 |
| 3     | 19.43333 | .9657648  | 20.12 | 0.000 | 17.504               | 21.36267 |
| 4     | 21.66667 | 1.246797  | 17.38 | 0.000 | 19.1759              | 24.15743 |
| 5     | 27.36364 | 1.594908  | 17.16 | 0.000 | 24.17744             | 30.54983 |

➤ Example 2

By default, margins accumulates a margin by using the observed relative frequencies of the factor levels.

```
. regress mpg i.foreign
```

| Source   | SS         | df | MS         | Number of obs = 74     |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | 378.153515 | 1  | 378.153515 | F( 1, 72) = 13.18      |  |  |
| Residual | 2065.30594 | 72 | 28.6848048 | Prob > F = 0.0005      |  |  |
|          |            |    |            | R-squared = 0.1548     |  |  |
|          |            |    |            | Adj R-squared = 0.1430 |  |  |
| Total    | 2443.45946 | 73 | 33.4720474 | Root MSE = 5.3558      |  |  |

| mpg       | Coef.    | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|-----------|----------|-----------|-------|-------|----------------------|----------|
| 1.foreign | 4.945804 | 1.362162  | 3.63  | 0.001 | 2.230384             | 7.661225 |
| _cons     | 19.82692 | .7427186  | 26.70 | 0.000 | 18.34634             | 21.30751 |

```
. margins
```

```
Predictive margins
```

```
Number of obs = 74
```

```
Model VCE : OLS
```

```
Expression : Linear prediction, predict()
```

|       | Delta-method |           | z     | P> z  | [95% Conf. Interval] |          |
|-------|--------------|-----------|-------|-------|----------------------|----------|
|       | Margin       | Std. Err. |       |       |                      |          |
| _cons | 21.2973      | .6226014  | 34.21 | 0.000 | 20.07702             | 22.51757 |

Let's set foreign to always accumulate using equal relative frequencies:

```
. fvset design asbalanced foreign
```

```
. regress mpg i.foreign
```

| Source   | SS         | df | MS         | Number of obs = 74     |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | 378.153515 | 1  | 378.153515 | F( 1, 72) = 13.18      |  |  |
| Residual | 2065.30594 | 72 | 28.6848048 | Prob > F = 0.0005      |  |  |
|          |            |    |            | R-squared = 0.1548     |  |  |
|          |            |    |            | Adj R-squared = 0.1430 |  |  |
| Total    | 2443.45946 | 73 | 33.4720474 | Root MSE = 5.3558      |  |  |

| mpg       | Coef.    | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|-----------|----------|-----------|-------|-------|----------------------|----------|
| 1.foreign | 4.945804 | 1.362162  | 3.63  | 0.001 | 2.230384             | 7.661225 |
| _cons     | 19.82692 | .7427186  | 26.70 | 0.000 | 18.34634             | 21.30751 |

```
. margins
```

```
Adjusted predictions
```

```
Number of obs = 74
```

```
Model VCE : OLS
```

```
Expression : Linear prediction, predict()
```

```
at : foreign (asbalanced)
```

|       | Delta-method |           | z     | P> z  | [95% Conf. Interval] |          |
|-------|--------------|-----------|-------|-------|----------------------|----------|
|       | Margin       | Std. Err. |       |       |                      |          |
| _cons | 22.29983     | .6810811  | 32.74 | 0.000 | 20.96493             | 23.63472 |

Suppose that we issued the `fvset design` command earlier in our session and that we cannot remember which variables we set as `asbalanced`. We can retrieve this information by using the `fvset report` command:

```
. fvset report, design(asbalanced)
Variable Base Design

foreign asbalanced
```



❑ Technical note

margins is aware of a factor variable’s design setting only through the estimation results it is working with. The design setting is stored by the estimation command; thus changing the design setting between the estimation command and margins will have no effect. For example, the output from the following two calls to margins yields the same results:

```
. fvset clear foreign
. regress mpg i.foreign
```

| Source   | SS         | df | MS         |                 |        |  |
|----------|------------|----|------------|-----------------|--------|--|
| Model    | 378.153515 | 1  | 378.153515 | Number of obs = | 74     |  |
| Residual | 2065.30594 | 72 | 28.6848048 | F( 1, 72) =     | 13.18  |  |
| Total    | 2443.45946 | 73 | 33.4720474 | Prob > F =      | 0.0005 |  |
|          |            |    |            | R-squared =     | 0.1548 |  |
|          |            |    |            | Adj R-squared = | 0.1430 |  |
|          |            |    |            | Root MSE =      | 5.3558 |  |

| mpg       | Coef.    | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|-----------|----------|-----------|-------|-------|----------------------|----------|
| 1.foreign | 4.945804 | 1.362162  | 3.63  | 0.001 | 2.230384             | 7.661225 |
| _cons     | 19.82692 | .7427186  | 26.70 | 0.000 | 18.34634             | 21.30751 |

```
. margins
Predictive margins
Model VCE : OLS
Expression : Linear prediction, predict()
```

|       | Delta-method |           |       |       |                      |          |
|-------|--------------|-----------|-------|-------|----------------------|----------|
|       | Margin       | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
| _cons | 21.2973      | .6226014  | 34.21 | 0.000 | 20.07702             | 22.51757 |

```
. fvset design asbalanced foreign
. margins
Predictive margins
Model VCE : OLS
Expression : Linear prediction, predict()
```

|       | Delta-method |           |       |       |                      |          |
|-------|--------------|-----------|-------|-------|----------------------|----------|
|       | Margin       | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
| _cons | 21.2973      | .6226014  | 34.21 | 0.000 | 20.07702             | 22.51757 |





## Saved results

`fvset` saves the following in `r()`:

Macros

|                            |                                                    |
|----------------------------|----------------------------------------------------|
| <code>r(varlist)</code>    | <i>varlist</i>                                     |
| <code>r(baselist)</code>   | base setting for each variable in <i>varlist</i>   |
| <code>r(designlist)</code> | design setting for each variable in <i>varlist</i> |

## Methods and formulas

`fvset` is implemented as an ado-file.

# Title

**gllamm** — Generalized linear and latent mixed models

## Description

GLLAMM stands for generalized linear latent and mixed models, and **gllamm** is a Stata command for fitting such models written by Sophia Rabe-Hesketh (University of California–Berkeley) as part of joint work with Anders Skrondal (London School of Economics and Political Science) and Andrew Pickles (University of Manchester).

## Remarks

Generalized linear latent and mixed models are a class of multilevel latent variable models, where a latent variable is a factor or a random effect (intercept or coefficient), or a disturbance (residual). The **gllamm** command for fitting such models is not an official command of Stata; it has been independently developed by highly regarded authors and is itself highly regarded. You can learn more about **gllamm** by visiting <http://www.gllamm.org>.

**gllamm** is available from the Statistical Software Components (SSC) archive. To install, type

```
. ssc describe gllamm
. ssc install gllamm
```

If you later wish to uninstall **gllamm**, type `ado uninstall gllamm`.

## References

- Rabe-Hesketh, S., and B. S. Everitt. 2007. *A Handbook of Statistical Analyses Using Stata*. 4th ed. Boca Raton, FL: Chapman & Hall/CRC.
- Rabe-Hesketh, S., A. Pickles, and C. Taylor. 2000. [sg129: Generalized linear latent and mixed models](#). *Stata Technical Bulletin* 53: 47–57. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 293–307. College Station, TX: Stata Press.
- Rabe-Hesketh, S., and A. Skrondal. 2008. *Multilevel and Longitudinal Modeling Using Stata*. 2nd ed. College Station, TX: Stata Press.
- Rabe-Hesketh, S., A. Skrondal, and A. Pickles. 2002. [Reliable estimation of generalized linear mixed models using adaptive quadrature](#). *Stata Journal* 2: 1–21.
- . 2003. [Maximum likelihood estimation of generalized linear models with covariate measurement error](#). *Stata Journal* 3: 386–411.
- Skrondal, A., and S. Rabe-Hesketh. 2004. *Generalized Latent Variable Modeling: Multilevel, Longitudinal, and Structural Equation Models*. Boca Raton, FL: Chapman & Hall/CRC.

The references below are restricted to works by the primary authors of **gllamm**. There are many other books and articles that use or discuss **gllamm**; see <http://www.gllamm.org/pub.html> for a list.

## Also see

- [[XT](#)] [xtmelogit](#) — Multilevel mixed-effects logistic regression
- [[XT](#)] [xtmepoisson](#) — Multilevel mixed-effects Poisson regression
- [[XT](#)] [xtmixed](#) — Multilevel mixed-effects linear regression

Syntax

```
glm depvar [indepvars] [if] [in] [weight] [, options]
```

| <i>options</i>                            | description                                                                                                                                                                            |
|-------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                     |                                                                                                                                                                                        |
| <b>family</b> ( <i>familyname</i> )       | distribution of <i>depvar</i> ; default is <b>family</b> (gaussian)                                                                                                                    |
| <b>link</b> ( <i>linkname</i> )           | link function; default is canonical link for <b>family</b> () specified                                                                                                                |
| Model 2                                   |                                                                                                                                                                                        |
| <b>noconstant</b>                         | suppress constant term                                                                                                                                                                 |
| <b>exposure</b> ( <i>varname</i> )        | include ln( <i>varname</i> ) in model with coefficient constrained to 1                                                                                                                |
| <b>offset</b> ( <i>varname</i> )          | include <i>varname</i> in model with coefficient constrained to 1                                                                                                                      |
| <b>constraints</b> ( <i>constraints</i> ) | apply specified linear constraints                                                                                                                                                     |
| <b>collinear</b>                          | keep collinear variables                                                                                                                                                               |
| <b>mu</b> ( <i>varname</i> )              | use <i>varname</i> as the initial estimate for the mean of <i>depvar</i>                                                                                                               |
| <b>init</b> ( <i>varname</i> )            | synonym for <b>mu</b> ( <i>varname</i> )                                                                                                                                               |
| SE/Robust                                 |                                                                                                                                                                                        |
| <b>vce</b> ( <i>vcetype</i> )             | <i>vcetype</i> may be oim, <b>robust</b> , <b>cluster</b> <i>clustvar</i> , eim, opg, <b>bootstrap</b> , <b>jackknife</b> , <b>hac kernel</b> , <b>jackknife1</b> , or <b>unbiased</b> |
| <b>vfactor</b> (#)                        | multiply variance matrix by scalar #                                                                                                                                                   |
| <b>disp</b> (#)                           | quasilikelihood multiplier                                                                                                                                                             |
| <b>scale</b> (x2   dev   #)               | set the scale parameter                                                                                                                                                                |
| Reporting                                 |                                                                                                                                                                                        |
| <b>level</b> (#)                          | set confidence level; default is <b>level</b> (95)                                                                                                                                     |
| <b>eform</b>                              | report exponentiated coefficients                                                                                                                                                      |
| <b>nocnsreport</b>                        | do not display constraints                                                                                                                                                             |
| <b>display_options</b>                    | control spacing and display of omitted variables and base and empty cells                                                                                                              |
| Maximization                              |                                                                                                                                                                                        |
| <b>ml</b>                                 | use maximum likelihood optimization; the default                                                                                                                                       |
| <b>irls</b>                               | use iterated, reweighted least-squares optimization of the deviance                                                                                                                    |
| <b>maximize_options</b>                   | control the maximization process; seldom used                                                                                                                                          |
| <b>fisher</b> (#)                         | use the Fisher scoring Hessian or expected information matrix (EIM)                                                                                                                    |
| <b>search</b>                             | search for good starting values                                                                                                                                                        |
| <sup>†</sup> <b>noheader</b>              | suppress header table from above coefficient table                                                                                                                                     |
| <sup>†</sup> <b>notable</b>               | suppress coefficient table                                                                                                                                                             |
| <sup>†</sup> <b>nodisplay</b>             | suppress the output; iteration log is still displayed                                                                                                                                  |
| <sup>†</sup> <b>coeflegend</b>            | display coefficients' legend instead of coefficient table                                                                                                                              |

| <i>familyname</i>                                                         | description        |
|---------------------------------------------------------------------------|--------------------|
| <code>gaussian</code>                                                     | Gaussian (normal)  |
| <code>igaussian</code>                                                    | inverse Gaussian   |
| <code>binomial</code> [ <i>varname</i> <sub><i>N</i></sub>   # <i>N</i> ] | Bernoulli/binomial |
| <code>poisson</code>                                                      | Poisson            |
| <code>nbinomial</code> [ <i>#<sub>k</sub></i>   <i>ml</i> ]               | negative binomial  |
| <code>gamma</code>                                                        | gamma              |

| <i>linkname</i>        | description       |
|------------------------|-------------------|
| <code>identity</code>  | identity          |
| <code>log</code>       | log               |
| <code>logit</code>     | logit             |
| <code>probit</code>    | probit            |
| <code>cloglog</code>   | cloglog           |
| <code>power #</code>   | power             |
| <code>opower #</code>  | odds power        |
| <code>nbinomial</code> | negative binomial |
| <code>loglog</code>    | log-log           |
| <code>logc</code>      | log-complement    |

<sup>†</sup>`noheader`, `notable`, `nodisplay`, and `coeflegend` do not appear in the dialog box.

`indepvars` may contain factor variables; see [U] 11.4.3 Factor variables.

`devar` and `indepvars` may contain time-series operators; see [U] 11.4.4 Time-series varlists.

`bootstrap`, `by`, `fracpoly`, `jackknife`, `mfp`, `mi estimate`, `nestreg`, `rolling`, `statsby`, `stepwise`, and `svy` are allowed; see [U] 11.1.10 Prefix commands.

`vce(bootstrap)`, `vce(jackknife)`, and `vce(jackknife1)` are not allowed with the `mi estimate` prefix.

Weights are not allowed with the `bootstrap` prefix.

`aweight`s are not allowed with the `jackknife` prefix.

`vce()`, `vfactor()`, `disp()`, `scale()`, `irls`, `fisher()`, `noheader`, `notable`, `nodisplay`, and `weights` are not allowed with the `svy` prefix.

`fweights`, `aweight`s, `iweight`s, and `pweight`s are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

## Menu

Statistics > Generalized linear models > Generalized linear models (GLM)

## Description

`glm` fits generalized linear models. It can fit models by using either IRLS (maximum quasilielihood) or Newton–Raphson (maximum likelihood) optimization, which is the default. Previous versions of `glm` used only IRLS.

See [R] [logistic](#) and [R] [regress](#) for lists of related estimation commands.

## Options

### Model

`family(familyname)` specifies the distribution of *depvar*; `family(gaussian)` is the default.

`link(linkname)` specifies the link function; the default is the canonical link for the `family()` specified.

### Model 2

`noconstant`, `exposure(varname)`, `offset(varname)`, `constraints(constraints)`, `collinear`; see [R] [estimation options](#). `constraints(constraints)` and `collinear` are not allowed with `irls`.

`mu(varname)` specifies *varname* as the initial estimate for the mean of *depvar*. This option can be useful with models that experience convergence difficulties, such as `family(binomial)` models with power or odds-power links. `init(varname)` is a synonym.

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

In addition to the standard *vcetypes*, `glm` allows the following alternatives:

`vce(eim)` specifies that the EIM estimate of variance be used.

`vce(jackknife1)` specifies that the one-step jackknife estimate of variance be used.

`vce(hac kernel [#])` specifies that a heteroskedasticity- and autocorrelation-consistent (HAC) variance estimate be used. HAC refers to the general form for combining weighted matrices to form the variance estimate. There are three kernels built into `glm`. *kernel* is a user-written program or one of

`nwest` | `gallant` | `anderson`

`#` specifies the number of lags. If `#` is not specified,  $N - 2$  is assumed. If you wish to specify `vce(hac ...)`, you must `tsset` your data before calling `glm`.

`vce(unbiased)` specifies that the unbiased sandwich estimate of variance be used.

`vfactor(#)` specifies a scalar by which to multiply the resulting variance matrix. This option allows you to match output with other packages, which may apply degrees of freedom or other small-sample corrections to estimates of variance.

`disp(#)` multiplies the variance of *depvar* by `#` and divides the deviance by `#`. The resulting distributions are members of the quasilikelihood family.

`scale(x2|dev|#)` overrides the default scale parameter. This option is allowed only with Hessian (information matrix) variance estimates.

By default, `scale(1)` is assumed for the discrete distributions (binomial, Poisson, and negative binomial), and `scale(x2)` is assumed for the continuous distributions (Gaussian, gamma, and inverse Gaussian).

`scale(x2)` specifies that the scale parameter be set to the Pearson chi-squared (or generalized chi-squared) statistic divided by the residual degrees of freedom, which is recommended by [McCullagh and Nelder \(1989\)](#) as a good general choice for continuous distributions.

`scale(dev)` sets the scale parameter to the deviance divided by the residual degrees of freedom. This option provides an alternative to `scale(x2)` for continuous distributions and overdispersed or underdispersed discrete distributions.

`scale(#)` sets the scale parameter to `#`. For example, using `scale(1)` in `family(gamma)` models results in exponential-errors regression. Additional use of `link(log)` rather than the default `link(power -1)` for `family(gamma)` essentially reproduces Stata's `streg`, `dist(exp)` `nohr` command (see [ST] [streg](#)) if all the observations are uncensored.

#### Reporting

`level(#)`; see [R] [estimation options](#).

`eform` displays the exponentiated coefficients and corresponding standard errors and confidence intervals. For `family(binomial) link(logit)` (i.e., logistic regression), exponentiation results in odds ratios; for `family(poisson) link(log)` (i.e., Poisson regression), exponentiated coefficients are rate ratios.

`nocnsreport`; see [R] [estimation options](#).

`display_options`: [noomitted](#), [vsquish](#), [noemptycells](#), [baselevels](#), [allbaselevels](#); see [R] [estimation options](#).

#### Maximization

`ml` requests that optimization be carried out using Stata's `ml` commands and is the default.

`irls` requests iterated, reweighted least-squares (IRLS) optimization of the deviance instead of Newton–Raphson optimization of the log likelihood. If the `irls` option is not specified, the optimization is carried out using Stata's `ml` commands, in which case all options of `ml maximize` are also available.

`maximize_options`: [difficult](#), [technique\(algorithm\\_spec\)](#), [iterate\(#\)](#), [\[no\]log](#), [trace](#), [gradient](#), [showstep](#), [hessian](#), [showtolerance](#), [tolerance\(#\)](#), [ltolerance\(#\)](#), [nrtolerance\(#\)](#), [nonrtolerance](#), [from\(init\\_specs\)](#); see [R] [maximize](#). These options are seldom used.

Setting the optimization type to `technique(bhhh)` resets the default `vcetype` to `vce(opg)`.

`fisher(#)` specifies the number of Newton–Raphson steps that should use the Fisher scoring Hessian or EIM before switching to the observed information matrix (OIM). This option is useful only for Newton–Raphson optimization (and not when using `irls`).

`search` specifies that the command search for good starting values. This option is useful only for Newton–Raphson optimization (and not when using `irls`).

The following options are available with `glm` but are not shown in the dialog box:

`noheader` suppresses the header information from the output. The coefficient table is still displayed.

`notable` suppresses the table of coefficients from the output. The header information is still displayed.

`nodisplay` suppresses the output. The iteration log is still displayed.

`coeflegend`; see [R] [estimation options](#).

## Remarks

Remarks are presented under the following headings:

*General use*  
*Variance estimators*  
*User-defined functions*

## General use

`glm` fits generalized linear models of  $y$  with covariates  $\mathbf{x}$ :

$$g\{E(y)\} = \mathbf{x}\beta, \quad y \sim F$$

$g(\cdot)$  is called the link function, and  $F$  is the distributional family. Substituting various definitions for  $g(\cdot)$  and  $F$  results in a surprising array of models. For instance, if  $y$  is distributed as Gaussian (normal) and  $g(\cdot)$  is the identity function, we have

$$E(y) = \mathbf{x}\beta, \quad y \sim \text{Normal}$$

or linear regression. If  $g(\cdot)$  is the logit function and  $y$  is distributed as Bernoulli, we have

$$\text{logit}\{E(y)\} = \mathbf{x}\beta, \quad y \sim \text{Bernoulli}$$

or logistic regression. If  $g(\cdot)$  is the natural log function and  $y$  is distributed as Poisson, we have

$$\ln\{E(y)\} = \mathbf{x}\beta, \quad y \sim \text{Poisson}$$

or Poisson regression, also known as the log-linear model. Other combinations are possible.

Although `glm` can be used to perform linear regression (and, in fact, does so by default), this regression should be viewed as an instructional feature; `regress` produces such estimates more quickly, and many postestimation commands are available to explore the adequacy of the fit; see [\[R\] regress](#) and [\[R\] regress postestimation](#).

In any case, you specify the link function by using the `link()` option and specify the distributional family by using `family()`. The available link functions are

| Link function         | glm option                   |
|-----------------------|------------------------------|
| identity              | <code>link(identity)</code>  |
| log                   | <code>link(log)</code>       |
| logit                 | <code>link(logit)</code>     |
| probit                | <code>link(probit)</code>    |
| complementary log-log | <code>link(cloglog)</code>   |
| odds power            | <code>link(opower #)</code>  |
| power                 | <code>link(power #)</code>   |
| negative binomial     | <code>link(nbinomial)</code> |
| log-log               | <code>link(loglog)</code>    |
| log-complement        | <code>link(logc)</code>      |

Define  $\mu = E(y)$  and  $\eta = g(\mu)$ , meaning that  $g(\cdot)$  maps  $E(y)$  to  $\eta = \mathbf{x}\beta + \text{offset}$ .

Link functions are defined as follows:

- `identity` is defined as  $\eta = g(\mu) = \mu$ .
- `log` is defined as  $\eta = \ln(\mu)$ .
- `logit` is defined as  $\eta = \ln\{\mu/(1 - \mu)\}$ , the natural log of the odds.
- `probit` is defined as  $\eta = \Phi^{-1}(\mu)$ , where  $\Phi^{-1}()$  is the inverse Gaussian cumulative.
- `cloglog` is defined as  $\eta = \ln\{-\ln(1 - \mu)\}$ .
- `opower` is defined as  $\eta = [\{\mu/(1 - \mu)\}^n - 1]/n$ , the power of the odds. The function is generalized so that `link(opower 0)` is equivalent to `link(logit)`, the natural log of the odds.
- `power` is defined as  $\eta = \mu^n$ . Specifying `link(power 1)` is equivalent to specifying `link(identity)`. The power function is generalized so that  $\mu^0 \equiv \ln(\mu)$ . Thus `link(power 0)` is equivalent to `link(log)`. Negative powers are, of course, allowed.
- `nbinomial` is defined as  $\eta = \ln\{\mu/(\mu + k)\}$ , where  $k = 1$  if `family(nbinomial)` is specified,  $k = \#_k$  if `family(nbinomial \#_k)` is specified, and  $k$  is estimated via maximum likelihood if `family(nbinomial ml)` is specified.
- `loglog` is defined as  $\eta = -\ln\{-\ln(\mu)\}$ .
- `logc` is defined as  $\eta = \ln(1 - \mu)$ .

The available distributional families are

| Family             | glm option                     |
|--------------------|--------------------------------|
| Gaussian (normal)  | <code>family(gaussian)</code>  |
| inverse Gaussian   | <code>family(igaussian)</code> |
| Bernoulli/binomial | <code>family(binomial)</code>  |
| Poisson            | <code>family(poisson)</code>   |
| negative binomial  | <code>family(nbinomial)</code> |
| gamma              | <code>family(gamma)</code>     |

`family(normal)` is a synonym for `family(gaussian)`.

The binomial distribution can be specified as 1) `family(binomial)`, 2) `family(binomial \#_N)`, or 3) `family(binomial varname_N)`. In case 2,  $\#_N$  is the value of the binomial denominator  $N$ , the number of trials. Specifying `family(binomial 1)` is the same as specifying `family(binomial)`. In case 3, `varname_N` is the variable containing the binomial denominator, allowing the number of trials to vary across observations.

The negative binomial distribution can be specified as 1) `family(nbinomial)`, 2) `family(nbinomial \#_k)`, or 3) `family(nbinomial ml)`. Omitting  $\#_k$  is equivalent to specifying `family(nbinomial 1)`. In case 3, the value of  $\#_k$  is estimated via maximum likelihood. The value  $\#_k$  enters the variance and deviance functions. Typical values range between 0.01 and 2; see the technical note below.

You do not have to specify both `family()` and `link()`; the default `link()` is the canonical link for the specified `family()` (except for `nbinomial`):



| Family                         | Default link                |
|--------------------------------|-----------------------------|
| <code>family(gaussian)</code>  | <code>link(identity)</code> |
| <code>family(igaussian)</code> | <code>link(power -2)</code> |
| <code>family(binomial)</code>  | <code>link(logit)</code>    |
| <code>family(poisson)</code>   | <code>link(log)</code>      |
| <code>family(nbinomial)</code> | <code>link(log)</code>      |
| <code>family(gamma)</code>     | <code>link(power -1)</code> |

If you specify both `family()` and `link()`, not all combinations make sense. You may choose from the following combinations:

|                   | identity | log | logit | probit | cloglog | power | opower | nbinomial | loglog | logc |
|-------------------|----------|-----|-------|--------|---------|-------|--------|-----------|--------|------|
| Gaussian          | x        | x   |       |        |         | x     |        |           |        |      |
| inverse Gaussian  | x        | x   |       |        |         | x     |        |           |        |      |
| binomial          | x        | x   | x     | x      | x       | x     | x      |           | x      | x    |
| Poisson           | x        | x   |       |        |         | x     |        |           |        |      |
| negative binomial | x        | x   |       |        |         | x     |        | x         |        |      |
| gamma             | x        | x   |       |        |         | x     |        |           |        |      |

□ Technical note

Some `family()` and `link()` combinations result in models already fit by Stata. These are

| family()  | link()   | Options                                                                     | Equivalent Stata command                                 |
|-----------|----------|-----------------------------------------------------------------------------|----------------------------------------------------------|
| gaussian  | identity | <i>nothing</i>   <code>irls</code>   <code>irls vce(oim)</code>             | <code>regress</code>                                     |
| gaussian  | identity | <code>t(var) vce(hac nwest #)</code><br><code>vfactor(#<sub>v</sub>)</code> | <code>newey, t(var) lag(#)</code> (see note 1)           |
| binomial  | cloglog  | <i>nothing</i>   <code>irls vce(oim)</code>                                 | <code>cloglog</code> (see note 2)                        |
| binomial  | probit   | <i>nothing</i>   <code>irls vce(oim)</code>                                 | <code>probit</code> (see note 2)                         |
| binomial  | logit    | <i>nothing</i>   <code>irls</code>   <code>irls vce(oim)</code>             | <code>logit</code> or <code>logistic</code> (see note 3) |
| poisson   | log      | <i>nothing</i>   <code>irls</code>   <code>irls vce(oim)</code>             | <code>poisson</code> (see note 3)                        |
| nbinomial | log      | <i>nothing</i>   <code>irls vce(oim)</code>                                 | <code>nbreg</code> (see note 4)                          |
| gamma     | log      | <code>scale(1)</code>                                                       | <code>streg, dist(exp) nohr</code> (see note 5)          |

Notes:

1. The variance factor  $\#_v$  should be set to  $n/(n - k)$ , where  $n$  is the number of observations and  $k$  the number of regressors. If the number of regressors is not specified, the estimated standard errors will, as a result, differ by this factor.
2. Because the link is not the canonical link for the binomial family, you must specify the `vce(oim)` option if using `irls` to get equivalent standard errors. If `irls` is used without `vce(oim)`, the regression coefficients will be the same but the standard errors will be only asymptotically equivalent. If no options are specified (*nothing*), `glm` will optimize using Newton–Raphson, making it equivalent to the other Stata command.

See [\[R\] cloglog](#) and [\[R\] probit](#) for more details about these commands.

3. Because the canonical link is being used, the standard errors will be equivalent whether the EIM or the OIM estimator of variance is used.

4. Family negative binomial, log-link models—also known as negative binomial regression models—are used for data with an overdispersed Poisson distribution. Although `glm` can be used to fit such models, using Stata’s maximum likelihood `nbreg` command is probably better. In the GLM approach, you specify `family(nbinomial #k)` and then search for a  $\#_k$  that results in the deviance-based dispersion being 1. You can also specify `family(nbinomial ml)` to estimate  $\#_k$  via maximum likelihood, which will report the same value returned from `nbreg`. However, `nbreg` also reports a confidence interval for it; see [R] [nbreg](#) and [Rogers \(1993\)](#). Of course, `glm` allows links other than `log`, and for those links, including the canonical `nbinomial` link, you will need to use `glm`.
5. `glm` can be used to estimate parameters from exponential regressions, but this method requires specifying `scale(1)`. However, censoring is not available. Censored exponential regression may be modeled using `glm` with `family(poisson)`. The log of the original response is entered into a Poisson model as an offset, whereas the new response is the censor variable. The result of such modeling is identical to the log relative hazard parameterization of `streg`, `dist(exp) nohr`. See [ST] [streg](#) for details about the `streg` command.

In general, where there is overlap between a capability of `glm` and that of some other Stata command, we recommend using the other Stata command. Our recommendation is not because of some inferiority of the GLM approach. Rather, those other commands, by being specialized, provide options and ancillary commands that are missing in the broader `glm` framework. Nevertheless, `glm` does produce the same answers where it should.

*Special note.* When equivalence is expected, for some datasets, you may still see very slight differences in the results, most often only in the later digits of the standard errors. When you compare `glm` output to an equivalent Stata command, these tiny discrepancies arise for many reasons:

- a. `glm` uses a general methodology for starting values, whereas the equivalent Stata command may be more specialized in its treatment of starting values.
- b. When using a canonical link, `glm`, `irls` should be equivalent to the maximum likelihood method of the equivalent Stata command, yet the convergence criterion is different (one is for deviance, the other for log likelihood). These discrepancies are easily resolved by adjusting one convergence criterion to correspond to the other.
- c. When both `glm` and the equivalent Stata command use Newton–Raphson, small differences may still occur if the Stata command has a different default convergence criterion from that of `glm`. Adjusting the convergence criterion will resolve the difference. See [R] [ml](#) and [R] [maximize](#) for more details.

□

## ► Example 1

In [example 1](#) of [R] [logistic](#), we fit a model based on data from a study of risk factors associated with low birthweight ([Hosmer Jr. and Lemeshow 2000](#), 25). We can replicate the estimation by using `glm`:

```

. use http://www.stata-press.com/data/r11/lbw
(Hosmer & Lemeshow data)

. glm low age lwt i.race smoke ptl ht ui, family(binomial) link(logit)

Iteration 0: log likelihood = -101.0213
Iteration 1: log likelihood = -100.72519
Iteration 2: log likelihood = -100.724
Iteration 3: log likelihood = -100.724

Generalized linear models No. of obs = 189
Optimization : ML Residual df = 180
 Scale parameter = 1
Deviance = 201.4479911 (1/df) Deviance = 1.119156
Pearson = 182.0233425 (1/df) Pearson = 1.011241
Variance function: V(u) = u*(1-u) [Bernoulli]
Link function : g(u) = ln(u/(1-u)) [Logit]
 AIC = 1.1611
Log likelihood = -100.7239956 BIC = -742.0665

```

| low   | OIM       |           | z     | P> z  | [95% Conf. Interval] |           |
|-------|-----------|-----------|-------|-------|----------------------|-----------|
|       | Coef.     | Std. Err. |       |       |                      |           |
| age   | -.0271003 | .0364504  | -0.74 | 0.457 | -.0985418            | .0443412  |
| lwt   | -.0151508 | .0069259  | -2.19 | 0.029 | -.0287253            | -.0015763 |
| race  |           |           |       |       |                      |           |
| 2     | 1.262647  | .5264101  | 2.40  | 0.016 | .2309024             | 2.294392  |
| 3     | .8620792  | .4391532  | 1.96  | 0.050 | .0013548             | 1.722804  |
| smoke | .9233448  | .4008266  | 2.30  | 0.021 | .137739              | 1.708951  |
| ptl   | .5418366  | .346249   | 1.56  | 0.118 | -.136799             | 1.220472  |
| ht    | 1.832518  | .6916292  | 2.65  | 0.008 | .4769494             | 3.188086  |
| ui    | .7585135  | .4593768  | 1.65  | 0.099 | -.1418484            | 1.658875  |
| _cons | .4612239  | 1.20459   | 0.38  | 0.702 | -1.899729            | 2.822176  |

glm, by default, presents coefficient estimates, whereas logistic presents the exponentiated coefficients—the odds ratios. glm's eform option reports exponentiated coefficients, and glm, like Stata's other estimation commands, replays results.

(Continued on next page)

```
. glm, eform
Generalized linear models
Optimization : ML
Deviance = 201.4479911
Pearson = 182.0233425
Variance function: V(u) = u*(1-u)
Link function : g(u) = ln(u/(1-u))
Log likelihood = -100.7239956
```

```
No. of obs = 189
Residual df = 180
Scale parameter = 1
(1/df) Deviance = 1.119156
(1/df) Pearson = 1.011241
[Bernoulli]
[Logit]
AIC = 1.1611
BIC = -742.0665
```

|       | OIM        |           |       |       |                      |          |
|-------|------------|-----------|-------|-------|----------------------|----------|
| low   | Odds Ratio | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
| age   | .9732636   | .0354759  | -0.74 | 0.457 | .9061578             | 1.045339 |
| lwt   | .9849634   | .0068217  | -2.19 | 0.029 | .9716834             | .9984249 |
| race  |            |           |       |       |                      |          |
| 2     | 3.534767   | 1.860737  | 2.40  | 0.016 | 1.259736             | 9.918406 |
| 3     | 2.368079   | 1.039949  | 1.96  | 0.050 | 1.001356             | 5.600207 |
| smoke | 2.517698   | 1.00916   | 2.30  | 0.021 | 1.147676             | 5.523162 |
| ptl   | 1.719161   | .5952579  | 1.56  | 0.118 | .8721455             | 3.388787 |
| ht    | 6.249602   | 4.322408  | 2.65  | 0.008 | 1.611152             | 24.24199 |
| ui    | 2.1351     | .9808153  | 1.65  | 0.099 | .8677528             | 5.2534   |

These results are the same as reported in [\[R\] logistic](#).

Included in the output header are values for the [Akaike \(1973\)](#) information criterion (AIC) and the Bayesian information criterion (BIC) ([Raftery 1995](#)). Both are measures of model fit adjusted for the number of parameters that can be compared across models. In both cases, a smaller value generally indicates a better model fit. AIC is based on the log likelihood and thus is available only when Newton–Raphson optimization is used. BIC is based on the deviance and thus is always available. ◀

□ Technical note

The values for AIC and BIC reported in the output after `glm` are different from those reported by `estat ic`:

```
. estat ic
```

| Model | Obs | ll(null) | ll(model) | df | AIC     | BIC      |
|-------|-----|----------|-----------|----|---------|----------|
| .     | 189 | .        | -100.724  | 9  | 219.448 | 248.6237 |

Note: N=Obs used in calculating BIC; see [\[R\] BIC note](#)

There are various definitions of these information criteria (IC) in the literature; `glm` and `estat ic` use different definitions. `glm` bases its computation of the BIC on deviance, whereas `estat ic` uses the likelihood. Both `glm` and `estat ic` use the likelihood to compute the AIC; however, the AIC from `estat ic` is equal to  $N$ , the number of observations, times the AIC from `glm`. Refer to [Methods and formulas](#) in this entry and [\[R\] estat](#) for the references and formulas used by `glm` and `estat`, respectively, to compute AIC and BIC. Inferences based on comparison of IC values reported by `glm` for different GLM models will be equivalent to those based on comparison of IC values reported by `estat ic` after `glm`.



## ► Example 2

We use data from an early insecticide experiment, given in [Pregibon \(1980\)](#). The variables are `ldose`, the log dose of insecticide; `n`, the number of flour beetles subjected to each dose; and `r`, the number killed.

```
. use http://www.stata-press.com/data/r11/ldose
. list, sep(4)
```

|    | ldose  | n  | r  |
|----|--------|----|----|
| 1. | 1.6907 | 59 | 6  |
| 2. | 1.7242 | 60 | 13 |
| 3. | 1.7552 | 62 | 18 |
| 4. | 1.7842 | 56 | 28 |
|    |        |    |    |
| 5. | 1.8113 | 63 | 52 |
| 6. | 1.8369 | 59 | 53 |
| 7. | 1.861  | 62 | 61 |
| 8. | 1.8839 | 60 | 60 |

The aim of the analysis is to estimate a dose–response relationship between  $p$ , the proportion killed, and  $X$ , the log dose.

As a first attempt, we will formulate the model as a linear logistic regression of  $p$  on `ldose`; that is, we will take the logit of  $p$  and represent the dose–response curve as a straight line in  $X$ :

$$\ln\{p/(1-p)\} = \beta_0 + \beta_1 X$$

Because the data are grouped, we cannot use Stata’s `logistic` command to fit the model. Stata does, however, already have a command for performing logistic regression on data organized in this way, so we could type

```
. blogit r n ldose
(output omitted)
```

Instead, we will fit the model by using `glm`:

```
. glm r ldose, family(binomial n) link(logit)
Iteration 0: log likelihood = -18.824848
Iteration 1: log likelihood = -18.715271
Iteration 2: log likelihood = -18.715123
Iteration 3: log likelihood = -18.715123
```

```
Generalized linear models
Optimization : ML
```

```
Deviance = 11.23220702
Pearson = 10.0267936
```

```
No. of obs = 8
Residual df = 6
Scale parameter = 1
(1/df) Deviance = 1.872035
(1/df) Pearson = 1.671132
```

(Continued on next page)

Variance function:  $V(u) = u \cdot (1 - u/n)$   
Link function :  $g(u) = \ln(u/(n-u))$

[Binomial]  
[Logit]

Log likelihood = -18.71512262

AIC = 5.178781  
BIC = -1.244442

| r     | OIM       |           | z      | P> z  | [95% Conf. Interval] |           |
|-------|-----------|-----------|--------|-------|----------------------|-----------|
|       | Coef.     | Std. Err. |        |       |                      |           |
| ldose | 34.27034  | 2.912141  | 11.77  | 0.000 | 28.56265             | 39.97803  |
| _cons | -60.71747 | 5.180713  | -11.72 | 0.000 | -70.87149            | -50.56346 |

The only difference between `blogit` and `glm` here is how they obtain the answer. `blogit` expands the data to contain 481 observations (the sum of `n`) so that it can run Stata’s standard, individual-level logistic command. `glm`, on the other hand, uses the information on the binomial denominator directly. We specified `family(binomial n)`, meaning that variable `n` contains the denominator. Parameter estimates and standard errors from the two approaches do not differ.

An alternative model, which gives asymmetric sigmoid curves for  $p$ , involves the complementary log-log, or cloglog, function:

$$\ln\{-\ln(1 - p)\} = \beta_0 + \beta_1 X$$

We fit this model by using `glm`:

```
. glm r ldose, family(binomial n) link(cloglog)
Iteration 0: log likelihood = -14.883594
Iteration 1: log likelihood = -14.822264
Iteration 2: log likelihood = -14.822228
Iteration 3: log likelihood = -14.822228

Generalized linear models
Optimization : ML
Deviance = 3.446418004
Pearson = 3.294675153
Variance function: $V(u) = u \cdot (1 - u/n)$
Link function : $g(u) = \ln(-\ln(1 - u/n))$
Log likelihood = -14.82222811
```

No. of obs = 8  
Residual df = 6  
Scale parameter = 1  
(1/df) Deviance = .574403  
(1/df) Pearson = .5491125

[Binomial]  
[Complementary log-log]  
AIC = 4.205557  
BIC = -9.030231

| r     | OIM       |           | z      | P> z  | [95% Conf. Interval] |           |
|-------|-----------|-----------|--------|-------|----------------------|-----------|
|       | Coef.     | Std. Err. |        |       |                      |           |
| ldose | 22.04118  | 1.793089  | 12.29  | 0.000 | 18.52679             | 25.55557  |
| _cons | -39.57232 | 3.229047  | -12.26 | 0.000 | -45.90114            | -33.24351 |

The complementary log-log model is preferred; the deviance for the logistic model, 11.23, is much higher than the deviance for the cloglog model, 3.45. This change also is evident by comparing log likelihoods, or equivalently, AIC values.

This example also shows the advantage of the `glm` command—we can vary assumptions easily. Note the minor difference in what we typed to obtain the logistic and cloglog models:

```
. glm r ldose, family(binomial n) link(logit)
. glm r ldose, family(binomial n) link(cloglog)
```

If we were performing this work for ourselves, we would have typed the commands in a more abbreviated form:

```
. glm r ldose, f(b n) l(1)
. glm r ldose, f(b n) l(c1)
```



❑ Technical note

Factor variables may be used with `glm`. Say that, in the [example above](#), we had `ldose`, the log dose of insecticide; `n`, the number of flour beetles subjected to each dose; and `r`, the number killed—all as before—except that now we have results for three different kinds of beetles. Our hypothetical data include `beetle`, which contains the values 1, 2, and 3.

```
. use http://www.stata-press.com/data/r11/beetle
. list, sep(0)
```

|                  | beetle | ldose  | n  | r  |
|------------------|--------|--------|----|----|
| 1.               | 1      | 1.6907 | 59 | 6  |
| 2.               | 1      | 1.7242 | 60 | 13 |
| 3.               | 1      | 1.7552 | 62 | 18 |
| 4.               | 1      | 1.7842 | 56 | 28 |
| 5.               | 1      | 1.8113 | 63 | 52 |
| (output omitted) |        |        |    |    |
| 23.              | 3      | 1.861  | 64 | 23 |
| 24.              | 3      | 1.8839 | 58 | 22 |

Let’s assume that, at first, we wish merely to add a shift factor for the type of beetle. We could type

```
. glm r i.beetle ldose, f(bin n) l(cloglog)
Iteration 0: log likelihood = -79.012269
Iteration 1: log likelihood = -76.94951
Iteration 2: log likelihood = -76.945645
Iteration 3: log likelihood = -76.945645

Generalized linear models
Optimization : ML

Deviance = 73.76505595
Pearson = 71.8901173

Variance function: V(u) = u*(1-u/n)
Link function : g(u) = ln(-ln(1-u/n))

Log likelihood = -76.94564525
```

```
No. of obs = 24
Residual df = 20
Scale parameter = 1
(1/df) Deviance = 3.688253
(1/df) Pearson = 3.594506
[Binomial]
[Complementary log-log]
AIC = 6.74547
BIC = 10.20398
```

| r      | OIM       |           |        | P> z  | [95% Conf. Interval] |           |
|--------|-----------|-----------|--------|-------|----------------------|-----------|
|        | Coef.     | Std. Err. | z      |       |                      |           |
| beetle |           |           |        |       |                      |           |
| 2      | -.0910396 | .1076132  | -0.85  | 0.398 | -.3019576            | .1198783  |
| 3      | -1.836058 | .1307125  | -14.05 | 0.000 | -2.09225             | -1.579867 |
| ldose  | 19.41558  | .9954265  | 19.50  | 0.000 | 17.46458             | 21.36658  |
| _cons  | -34.84602 | 1.79333   | -19.43 | 0.000 | -38.36089            | -31.33116 |

We find strong evidence that the insecticide works differently on the third kind of beetle. We now check whether the curve is merely shifted or also differently sloped:

```
. glm r beetle##c.ldose, f(bin n) l(cloglog)
Iteration 0: log likelihood = -67.270188
Iteration 1: log likelihood = -65.149316
Iteration 2: log likelihood = -65.147978
Iteration 3: log likelihood = -65.147978

Generalized linear models
Optimization : ML
Deviance = 50.16972096
Pearson = 49.28422567
Variance function: V(u) = u*(1-u/n)
Link function : g(u) = ln(-ln(1-u/n))

Log likelihood = -65.14797776

No. of obs = 24
Residual df = 18
Scale parameter = 1
(1/df) Deviance = 2.787207
(1/df) Pearson = 2.738013
[Binomial]
[Complementary log-log]
AIC = 5.928998
BIC = -7.035248
```

| r       | OIM       |           |        |       |                      |           |
|---------|-----------|-----------|--------|-------|----------------------|-----------|
|         | Coef.     | Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
| beetle  |           |           |        |       |                      |           |
| 2       | -.79933   | 4.470882  | -0.18  | 0.858 | -9.562098            | 7.963438  |
| 3       | 17.78741  | 4.586429  | 3.88   | 0.000 | 8.798172             | 26.77664  |
| ldose   | 22.04118  | 1.793089  | 12.29  | 0.000 | 18.52679             | 25.55557  |
| beetle# |           |           |        |       |                      |           |
| c.ldose |           |           |        |       |                      |           |
| 2       | .3838708  | 2.478477  | 0.15   | 0.877 | -4.473855            | 5.241596  |
| 3       | -10.726   | 2.526412  | -4.25  | 0.000 | -15.67768            | -5.774321 |
| _cons   | -39.57232 | 3.229047  | -12.26 | 0.000 | -45.90114            | -33.24351 |

We find that the (complementary log-log) dose–response curve for the third kind of beetle has roughly half the slope of that for the first kind.

See [U] 25 Working with categorical data and factor variables; what is said there concerning linear regression is applicable to any GLM model.





## Variance estimators

glm offers many variance options and gives different types of standard errors when used in various combinations. We highlight some of them here, but for a full explanation, see [Hardin and Hilbe \(2007\)](#).

### ► Example 3

Continuing with our flour beetle data, we rerun the most recently displayed model, this time requesting estimation via IRLS.

```
. use http://www.stata-press.com/data/r11/beetle
. glm r beetle#c.ldose, f(bin n) l(cloglog) ltol(1e-15) irls

Iteration 1: deviance = 54.41414
Iteration 2: deviance = 50.19424
Iteration 3: deviance = 50.16973
(output omitted)
Iteration 256: deviance = 50.16972

Generalized linear models No. of obs = 24
Optimization : MQL Fisher scoring Residual df = 18
 (IRLS EIM) Scale parameter = 1
Deviance = 50.16972096 (1/df) Deviance = 2.787207
Pearson = 49.28422567 (1/df) Pearson = 2.738013

Variance function: V(u) = u*(1-u/n) [Binomial]
Link function : g(u) = ln(-ln(1-u/n)) [Complementary log-log]
BIC = -7.035248
```

| r       | EIM       |           | z      | P> z  | [95% Conf. Interval] |           |
|---------|-----------|-----------|--------|-------|----------------------|-----------|
|         | Coef.     | Std. Err. |        |       |                      |           |
| beetle  |           |           |        |       |                      |           |
| 2       | -.79933   | 4.586649  | -0.17  | 0.862 | -9.788997            | 8.190337  |
| 3       | 17.78741  | 4.624834  | 3.85   | 0.000 | 8.7229               | 26.85192  |
| ldose   | 22.04118  | 1.799356  | 12.25  | 0.000 | 18.5145              | 25.56785  |
| beetle# |           |           |        |       |                      |           |
| c.ldose |           |           |        |       |                      |           |
| 2       | .3838708  | 2.544068  | 0.15   | 0.880 | -4.602411            | 5.370152  |
| 3       | -10.726   | 2.548176  | -4.21  | 0.000 | -15.72033            | -5.731665 |
| _cons   | -39.57232 | 3.240274  | -12.21 | 0.000 | -45.92314            | -33.2215  |

Note our use of the `ltol()` option, which, although unrelated to our discussion on variance estimation, was used so that the regression coefficients would match those of the previous Newton–Raphson (NR) fit.

Because IRLS uses the EIM for optimization, the variance estimate is also based on EIM. If we want optimization via IRLS but the variance estimate based on OIM, we specify `glm, irls vce(oim)`:

(Continued on next page)

```
. glm r beetle##c.ldose, f(b n) l(cl) ltol(1e-15) irls vce(oim) noheader nolog
```

| r       | OIM       |           | z      | P> z  | [95% Conf. Interval] |           |
|---------|-----------|-----------|--------|-------|----------------------|-----------|
|         | Coef.     | Std. Err. |        |       |                      |           |
| beetle  |           |           |        |       |                      |           |
| 2       | -.79933   | 4.470882  | -0.18  | 0.858 | -9.562098            | 7.963438  |
| 3       | 17.78741  | 4.586429  | 3.88   | 0.000 | 8.798172             | 26.77664  |
| ldose   | 22.04118  | 1.793089  | 12.29  | 0.000 | 18.52679             | 25.55557  |
| beetle# |           |           |        |       |                      |           |
| c.ldose |           |           |        |       |                      |           |
| 2       | .3838708  | 2.478477  | 0.15   | 0.877 | -4.473855            | 5.241596  |
| 3       | -10.726   | 2.526412  | -4.25  | 0.000 | -15.67768            | -5.774321 |
| _cons   | -39.57232 | 3.229047  | -12.26 | 0.000 | -45.90114            | -33.24351 |

This approach is identical to NR except for the convergence path. Because the cloglog link is not the canonical link for the binomial family, EIM and OIM produce different results. Both estimators, however, are asymptotically equivalent.

Going back to NR, we can also specify `vce(robust)` to get the Huber/White/sandwich estimator of variance:

```
. glm r beetle##c.ldose, f(b n) l(cl) vce(robust) noheader nolog
```

| r       | Robust    |           | z      | P> z  | [95% Conf. Interval] |           |
|---------|-----------|-----------|--------|-------|----------------------|-----------|
|         | Coef.     | Std. Err. |        |       |                      |           |
| beetle  |           |           |        |       |                      |           |
| 2       | -.79933   | 5.733049  | -0.14  | 0.889 | -12.0359             | 10.43724  |
| 3       | 17.78741  | 5.158477  | 3.45   | 0.001 | 7.676977             | 27.89784  |
| ldose   | 22.04118  | .8998551  | 24.49  | 0.000 | 20.27749             | 23.80486  |
| beetle# |           |           |        |       |                      |           |
| c.ldose |           |           |        |       |                      |           |
| 2       | .3838708  | 3.174427  | 0.12   | 0.904 | -5.837892            | 6.605633  |
| 3       | -10.726   | 2.800606  | -3.83  | 0.000 | -16.21508            | -5.236912 |
| _cons   | -39.57232 | 1.621306  | -24.41 | 0.000 | -42.75003            | -36.39462 |

The sandwich estimator gets its name from the form of the calculation—it is the multiplication of three matrices, with the outer two matrices (the “bread”) set to the OIM variance matrix. When `irls` is used along with `vce(robust)`, the EIM variance matrix is instead used as the bread, and the resulting variance is labeled “Semirobust”.

```
. glm r beetle##c.ldose, f(b n) l(c1) irls ltol(1e-15) vce(robust) noheader
> nolog
```

| r              | Semirobust |           | z      | P> z  | [95% Conf. Interval] |           |
|----------------|------------|-----------|--------|-------|----------------------|-----------|
|                | Coef.      | Std. Err. |        |       |                      |           |
| beetle         |            |           |        |       |                      |           |
| 2              | -.79933    | 6.288963  | -0.13  | 0.899 | -13.12547            | 11.52681  |
| 3              | 17.78741   | 5.255307  | 3.38   | 0.001 | 7.487194             | 28.08762  |
| ldose          | 22.04118   | .9061566  | 24.32  | 0.000 | 20.26514             | 23.81721  |
| beetle#c.ldose |            |           |        |       |                      |           |
| 2              | .3838708   | 3.489723  | 0.11   | 0.912 | -6.455861            | 7.223603  |
| 3              | -10.726    | 2.855897  | -3.76  | 0.000 | -16.32345            | -5.128542 |
| _cons          | -39.57232  | 1.632544  | -24.24 | 0.000 | -42.77205            | -36.3726  |

The outer product of the gradient (OPG) estimate of variance is one that avoids the calculation of second derivatives. It is equivalent to the “middle” part of the sandwich estimate of variance and can be specified by using `glm`, `vce(opg)`, regardless of whether NR or IRLS optimization is used.

```
. glm r beetle##c.ldose, f(b n) l(c1) vce(opg) noheader nolog
```

| r              | OPG       |           | z     | P> z  | [95% Conf. Interval] |           |
|----------------|-----------|-----------|-------|-------|----------------------|-----------|
|                | Coef.     | Std. Err. |       |       |                      |           |
| beetle         |           |           |       |       |                      |           |
| 2              | -.79933   | 6.664045  | -0.12 | 0.905 | -13.86062            | 12.26196  |
| 3              | 17.78741  | 6.838505  | 2.60  | 0.009 | 4.384183             | 31.19063  |
| ldose          | 22.04118  | 3.572983  | 6.17  | 0.000 | 15.03826             | 29.0441   |
| beetle#c.ldose |           |           |       |       |                      |           |
| 2              | .3838708  | 3.700192  | 0.10  | 0.917 | -6.868372            | 7.636114  |
| 3              | -10.726   | 3.796448  | -2.83 | 0.005 | -18.1669             | -3.285097 |
| _cons          | -39.57232 | 6.433101  | -6.15 | 0.000 | -52.18097            | -26.96368 |

The OPG estimate of variance is a component of the BHHH (Berndt et al. 1974) optimization technique. This method of optimization is also available with `glm` with the `technique()` option; however, the `technique()` option is not allowed with the `irls` option.

◀

## ► Example 4

The Newey–West (1987) estimator of variance is a sandwich estimator with the “middle” of the sandwich modified to take into account possible autocorrelation between the observations. These estimators are a generalization of those given by the Stata command `newey` for linear regression. See [TS] [newey](#) for more details.

For example, consider the dataset given in [TS] [newey](#), which has time-series measurements on `usr` and `idle`. We want to perform a linear regression with Newey–West standard errors.

```
. use http://www.stata-press.com/data/r11/idle2
. list usr idle time
```

|                  | usr | idle | time |
|------------------|-----|------|------|
| 1.               | 0   | 100  | 1    |
| 2.               | 0   | 100  | 2    |
| 3.               | 0   | 97   | 3    |
| 4.               | 1   | 98   | 4    |
| 5.               | 2   | 94   | 5    |
| (output omitted) |     |      |      |
| 29.              | 1   | 98   | 29   |
| 30.              | 1   | 98   | 30   |

Examining *Methods and formulas* of [TS] [newey](#), we see that the variance estimate is multiplied by a correction factor of  $n/(n - k)$ , where  $k$  is the number of regressors. `glm`, `vce(hac ...)` does not make this correction, so to get the same standard errors, we must use the `vfactor()` option within `glm` to make the correction manually.

```
. display 30/28
1.0714286
. tsset time
 time variable: time, 1 to 30
 delta: 1 unit
. glm usr idle, vce(hac nwest 3) vfactor(1.0714286)
Iteration 0: log likelihood = -71.743396
Generalized linear models No. of obs = 30
Optimization : ML Residual df = 28
Scale parameter = 7.493297
Deviance = 209.8123165 (1/df) Deviance = 7.493297
Pearson = 209.8123165 (1/df) Pearson = 7.493297
Variance function: V(u) = 1 [Gaussian]
Link function : g(u) = u [Identity]
HAC kernel (lags): Newey-West (3)
Log likelihood = -71.74339627
AIC = 4.916226
BIC = 114.5788
```

| usr   | HAC       |           | z     | P> z  | [95% Conf. Interval] |           |
|-------|-----------|-----------|-------|-------|----------------------|-----------|
|       | Coef.     | Std. Err. |       |       |                      |           |
| idle  | -.2281501 | .0690928  | -3.30 | 0.001 | -.3635694            | -.0927307 |
| _cons | 23.13483  | 6.327033  | 3.66  | 0.000 | 10.73407             | 35.53558  |

The `glm` command above reproduces the results given in [TS] [newey](#). We may now generalize this output to models other than simple linear regression and to different kernel weights.

```
. glm usr idle, fam(gamma) link(log) vce(hac gallant 3)
Iteration 0: log likelihood = -61.76593
Iteration 1: log likelihood = -60.963233
Iteration 2: log likelihood = -60.95097
Iteration 3: log likelihood = -60.950965

Generalized linear models No. of obs = 30
Optimization : ML Residual df = 28
 Scale parameter = .431296
 (1/df) Deviance = .3538752
 (1/df) Pearson = .431296
Deviance = 9.908506707 [Gamma]
Pearson = 12.07628677 [Log]
Variance function: V(u) = u^2
Link function : g(u) = ln(u)
HAC kernel (lags): Gallant (3)

Log likelihood = -60.95096484 AIC = 4.196731
 BIC = -85.32502
```

| usr   | Coef.     | HAC<br>Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-------|-----------|------------------|-------|-------|----------------------|-----------|
| idle  | -.0796609 | .0184647         | -4.31 | 0.000 | -.115851             | -.0434708 |
| _cons | 7.771011  | 1.510198         | 5.15  | 0.000 | 4.811078             | 10.73094  |

glm also offers variance estimators based on the bootstrap (resampling your data with replacement) and the jackknife (refitting the model with each observation left out in succession). Also included is the one-step jackknife estimate, which, instead of performing full reestimation when each observation is omitted, calculates a one-step NR estimate, with the full data regression coefficients as starting values.

```
. set seed 1
. glm usr idle, fam(gamma) link(log) vce(bootstrap, reps(100) nodots)
Generalized linear models No. of obs = 30
Optimization : ML Residual df = 28
 Scale parameter = .431296
 (1/df) Deviance = .3538752
 (1/df) Pearson = .431296
Deviance = 9.908506707 [Gamma]
Pearson = 12.07628677 [Log]
Variance function: V(u) = u^2
Link function : g(u) = ln(u)
 AIC = 4.196731
Log likelihood = -60.95096484 BIC = -85.32502
```

| usr   | Observed<br>Coef. | Bootstrap<br>Std. Err. | z     | P> z  | Normal-based<br>[95% Conf. Interval] |           |
|-------|-------------------|------------------------|-------|-------|--------------------------------------|-----------|
| idle  | -.0796609         | .0216591               | -3.68 | 0.000 | -.1221119                            | -.0372099 |
| _cons | 7.771011          | 1.80278                | 4.31  | 0.000 | 4.237627                             | 11.3044   |

4

See [Hardin and Hilbe \(2007\)](#) for a full discussion of the variance options that go with glm and, in particular, of how the different variance estimators are modified when `vce(cluster clustvar)` is specified. Finally, not all variance options are supported with all types of weights. See `help glm` for a current table of the variance options that are supported with the different weights.

## User-defined functions

glm may be called with a user-written link function, variance (family) function, Newey–West kernel-weight function, or any combination of the three.

### Syntax of link functions

```

program programe
 version 11
 args todo eta mu return
 if 'todo' == -1 {
 /* Set global macros for output */
 global SGLM_lt " title for link function "
 global SGLM_lf " subtitle showing link definition "
 exit
 }
 if 'todo' == 0 {
 /* set $\eta = g(\mu)$ */
 /* Intermediate calculations go here */
 generate double 'eta' = ...
 exit
 }
 if 'todo' == 1 {
 /* set $\mu = g^{-1}(\eta)$ */
 /* Intermediate calculations go here */
 generate double 'mu' = ...
 exit
 }
 if 'todo' == 2 {
 /* set return = $\partial\mu/\partial\eta$ */
 /* Intermediate calculations go here */
 generate double 'return' = ...
 exit
 }
 if 'todo' == 3 {
 /* set return = $\partial^2\mu/\partial\eta^2$ */
 /* Intermediate calculations go here */
 generate double 'return' = ...
 exit
 }
 display as error "Unknown call to glm link function"
 exit 198
end

```

## Syntax of variance functions

```

program progname
 version 11
 args todo eta mu return
 if 'todo' == -1 {
 /* Set global macros for output */
 /* Also check that depvar is in proper range */
 /* Note: For this call, eta contains indicator for whether each obs. is in est. sample */
 global SGLM_vt " title for variance function "
 global SGLM_vf " subtitle showing function definition "
 global SGLM_mu " program to call to enforce boundary conditions on μ "
 exit
 }
 if 'todo' == 0 {
 /* set η to initial value. */
 /* Intermediate calculations go here */
 generate double 'eta' = ...
 exit
 }
 if 'todo' == 1 {
 /* set return = $V(\mu)$ */
 /* Intermediate calculations go here */
 generate double 'return' = ...
 exit
 }
 if 'todo' == 2 {
 /* set return = $\partial V(\mu)/\partial \mu$ */
 /* Intermediate calculations go here */
 generate double 'return' = ...
 exit
 }
 if 'todo' == 3 {
 /* set return = squared deviance (per observation) */
 /* Intermediate calculations go here */
 generate double 'return' = ...
 exit
 }
 if 'todo' == 4 {
 /* set return = Anscombe residual */
 /* Intermediate calculations go here */
 generate double 'return' = ...
 exit
 }
 if 'todo' == 5 {
 /* set return = log likelihood */
 /* Intermediate calculations go here */
 generate double 'return' = ...
 exit
 }
 if 'todo' == 6 {
 /* set return = adjustment for deviance residuals */
 /* Intermediate calculations go here */
 generate double 'return' = ...
 exit
 }
 display as error "Unknown call to glm variance function"
 exit 198
end

```

### Syntax of Newey–West kernel-weight functions

```
program progrname, rclass
 version 11
 args G j
 /* G is the maximum lag */
 /* j is the current lag */
 /* Intermediate calculations go here */
 return scalar wt = computed weight
 return local setype "Newey–West"
 return local sewtype " name of kernel "
end
```

### Global macros available for user-written programs

| Global macro | Description                                                                                                           |
|--------------|-----------------------------------------------------------------------------------------------------------------------|
| SGLM_V       | program name of variance (family) evaluator                                                                           |
| SGLM_L       | program name of link evaluator                                                                                        |
| SGLM_y       | dependent variable name                                                                                               |
| SGLM_m       | binomial denominator                                                                                                  |
| SGLM_a       | negative binomial $k$                                                                                                 |
| SGLM_p       | power if <code>power()</code> or <code>opower()</code> is used, or<br>an argument from a user-specified link function |
| SGLM_s1      | indicator; set to one if scale is equal to one                                                                        |
| SGLM_ph      | value of scale parameter                                                                                              |

### ► Example 5

Suppose that we wish to perform Poisson regression with a log-link function. Although this regression is already possible with standard `glm`, we will write our own version for illustrative purposes.

Because we want a log link,  $\eta = g(\mu) = \ln(\mu)$ , and for a Poisson family the variance function is  $V(\mu) = \mu$ .

The Poisson density is given by

$$f(y_i) = \frac{e^{-\exp(\mu_i)} e^{\mu_i y_i}}{y_i!}$$

resulting in a log likelihood of

$$L = \sum_{i=1}^n \{-e^{\mu_i} + \mu_i y_i - \ln(y_i!)\}$$

The squared deviance of the  $i$ th observation for the Poisson family is given by

$$d_i^2 = \begin{cases} 2\hat{\mu}_i & \text{if } y_i = 0 \\ 2\{y_i \ln(y_i/\hat{\mu}_i) - (y_i - \hat{\mu}_i)\} & \text{otherwise} \end{cases}$$



We now have enough information to write our own Poisson-log glm module. We create the file `mylog.ado`, which contains

```

program mylog
 version 11
 args todo eta mu return
 if 'todo' == -1 {
 global SGLM_lt "My Log" // Titles for output
 global SGLM_lf "ln(u)"
 exit
 }
 if 'todo' == 0 {
 gen double 'eta' = ln('mu') // $\eta = \ln(\mu)$
 exit
 }
 if 'todo' == 1 {
 gen double 'mu' = exp('eta') // $\mu = \exp(\eta)$
 exit
 }
 if 'todo' == 2 {
 gen double 'return' = 'mu' // $\partial\mu/\partial\eta = \exp(\eta) = \mu$
 exit
 }
 if 'todo' == 3 {
 gen double 'return' = 'mu' // $\partial^2\mu/\partial\eta^2 = \exp(\eta) = \mu$
 exit
 }
 di as error "Unknown call to glm link function"
 exit 198
end

```

and we create the file `mypoiss.ado`, which contains

```

program mypoiss
 version 11
 args todo eta mu return
 if 'todo' == -1 {
 local y "$SGLM_y"
 local touse "'eta'" // 'eta' marks estimation sample here
 capture assert 'y'>=0 if 'touse' // check range of y
 if _rc {
 di as error "'dependent variable 'y' has negative values'"
 exit 499
 }
 global SGLM_vt "My Poisson" // Titles for output
 global SGLM_vf "u"
 global SGLM_mu "glim_mu 0 ." // see note 1
 exit
 }
 if 'todo' == 0 {
 gen double 'eta' = ln('mu') // Initialization of η ; see note 2
 exit
 }
}

```

*(Continued on next page)*

```

if `todo' == 1 {
 gen double `return' = `mu' // $V(\mu) = \mu$
 exit
}
if `todo' == 2 {
 gen byte `return' = 1 // $\partial V(\mu)/\partial \mu$
 exit
}
if `todo' == 3 {
 local y "$SGLM_y" // squared deviance, defined above
 if "`y'" == "" {
 local y "e(depvar)"
 }
 gen double `return' = cond(`y'==0, 2*`mu', /
 / 2(`y'*ln(`y'/'mu')-(`y'-`mu'))))
 exit
}
if `todo' == 4 {
 local y "$SGLM_y" // Anscombe residual; see note 3
 if "`y'" == "" {
 local y "e(depvar)"
 }
 gen double `return' = 1.5*(`y'^(2/3)-`mu'^(2/3)) / `mu'^(1/6)
 exit
}
if `todo' == 5 {
 local y "$SGLM_y" // log likelihood; see note 4
 if "`y'" == "" {
 local y "e(depvar)"
 }
 gen double `return' = -`mu'+`y'*ln(`mu')-lgamma(`y'+1)
 exit
}
if `todo' == 6 {
 gen double `return' = 1/(6*sqrt(`mu')) // adjustment to residual; see note 5
 exit
}
di as error "Unknown call to glm variance function"
error 198
end

```

## Notes:

1. `glm_mu` is a Stata program that will, at each iteration, bring  $\hat{\mu}$  back into its plausible range, should it stray out of it. Here `glm_mu` is called with the arguments zero and missing, meaning that zero is the lower bound of  $\hat{\mu}$  and there exists no upper bound—such is the case for Poisson models.
2. Here the initial value of  $\eta$  is easy because we intend to fit this model with our user-defined log link. In general, however, the initialization may need to vary according to the link to obtain convergence. If so, the global macro `SGLM_L` is used to determine which link is being utilized.
3. The Anscombe formula is given here because we know it. If we were not interested in Anscombe residuals, we could merely set ``return'` to missing. Also, the local macro `y` is set either to `SGLM_y` if it is in current estimation or to `e(depvar)` if this function is being accessed by `predict`.
4. If we were not interested in ML estimation, we could omit this code entirely and just leave an `exit` statement in its place. Similarly, if we were not interested in deviance or IRLS optimization, we could set ``return'` in the deviance portion of the code (``todo'==3`) to missing.

5. This code defines the term to be added to the predicted residuals if the `adjusted` option is specified. Again, if we were not interested, we could set `'return'` to missing.

We can now test our Poisson-log module by running it on the airline data presented in [R] [poisson](#).

```
. use http://www.stata-press.com/data/r11/airline
. list airline injuries n XYZowned
```

|    | airline | injuries | n      | XYZowned |
|----|---------|----------|--------|----------|
| 1. | 1       | 11       | 0.0950 | 1        |
| 2. | 2       | 7        | 0.1920 | 0        |
| 3. | 3       | 7        | 0.0750 | 0        |
| 4. | 4       | 19       | 0.2078 | 0        |
| 5. | 5       | 9        | 0.1382 | 0        |
| 6. | 6       | 4        | 0.0540 | 1        |
| 7. | 7       | 3        | 0.1292 | 0        |
| 8. | 8       | 1        | 0.0503 | 0        |
| 9. | 9       | 3        | 0.0629 | 1        |

```
. gen lnN=ln(n)
. glm injuries XYZowned lnN, fam(mypois) link(mylog) scale(1)
```

```
Iteration 0: log likelihood = -22.557572
Iteration 1: log likelihood = -22.332861
Iteration 2: log likelihood = -22.332276
Iteration 3: log likelihood = -22.332276
```

Generalized linear models

Optimization : ML

Deviance = 12.70432823

Pearson = 12.7695081

Variance function:  $V(u) = u$

Link function :  $g(u) = \ln(u)$

Log likelihood = -22.33227605

No. of obs = 9

Residual df = 6

Scale parameter = 1

(1/df) Deviance = 2.117388

(1/df) Pearson = 2.128251

[My Poisson]

[My Log]

AIC = 5.629395

BIC = -.4790192

| injuries | OIM      |           | z    | P> z  | [95% Conf. Interval] |          |
|----------|----------|-----------|------|-------|----------------------|----------|
|          | Coef.    | Std. Err. |      |       |                      |          |
| XYZowned | .6840668 | .3895877  | 1.76 | 0.079 | -.0795111            | 1.447645 |
| lnN      | 1.424169 | .3725155  | 3.82 | 0.000 | .6940517             | 2.154286 |
| _cons    | 4.863891 | .7090501  | 6.86 | 0.000 | 3.474178             | 6.253603 |

(Standard errors scaled using dispersion equal to square root of 1.)

These are precisely the results given in [R] [poisson](#) and are those that would have been given had we run `glm, family(poisson) link(log)`. The only minor adjustment we needed to make was to specify the `scale(1)` option. If `scale()` is left unspecified, `glm` assumes `scale(1)` for discrete distributions and `scale(x2)` for continuous ones. By default, `glm` assumes that any user-defined family is continuous because it has no way of checking. Thus we needed to specify `scale(1)` because our model is discrete.

Because we were careful in defining the squared deviance, we could have fit this model with IRLS. Because `log` is the canonical link for the Poisson family, we would not only get the same regression coefficients but also the same standard errors.

## ► Example 6

Suppose now that we wish to use our log link (`mylog.ado`) with `glm`'s binomial family. This task requires some modification because our current function is not equipped to deal with the binomial denominator, which we are allowed to specify. This denominator is accessible to our link function through the global macro `SGLM_m`. We now make the modifications and store them in `mylog2.ado`.

```

program mylog2 // <-- changed
 version 11
 args todo eta mu return
 if 'todo' == -1 {
 global SGLM_lt "My Log, Version 2" // <-- changed
 if "$SGLM_m" == "1" { // <-- changed
 global SGLM_lf "ln(u)" // <-- changed
 } // <-- changed
 else global SGLM_lf "ln(u/$SGLM_m)" // <-- changed
 exit
 }
 if 'todo' == 0 {
 gen double 'eta' = ln('mu'/$SGLM_m) // <-- changed
 exit
 }
 if 'todo' == 1 {
 gen double 'mu' = $SGLM_m*exp('eta') // <-- changed
 exit
 }
 if 'todo' == 2 {
 gen double 'return' = 'mu'
 exit
 }
 if 'todo' == 3 {
 gen double 'return' = 'mu'
 exit
 }
 di as error "Unknown call to glm link function"
 exit 198
end

```

We can now run our new log link with `glm`'s binomial family. Using the flour beetle data from earlier, we have

```
. use http://www.stata-press.com/data/r11/beetle, clear
. glm r ldose, fam(bin n) link(mylog2) irls

Iteration 1: deviance = 2212.108
Iteration 2: deviance = 452.9352
Iteration 3: deviance = 429.95
Iteration 4: deviance = 429.2745
Iteration 5: deviance = 429.2192
Iteration 6: deviance = 429.2082
Iteration 7: deviance = 429.2061
Iteration 8: deviance = 429.2057
Iteration 9: deviance = 429.2056
Iteration 10: deviance = 429.2056
Iteration 11: deviance = 429.2056
Iteration 12: deviance = 429.2056

Generalized linear models No. of obs = 24
Optimization : MQL Fisher scoring Residual df = 22
 (IRLS EIM) Scale parameter = 1
Deviance = 429.205599 (1/df) Deviance = 19.50935
Pearson = 413.088142 (1/df) Pearson = 18.77673

Variance function: V(u) = u*(1-u/n) [Binomial]
Link function : g(u) = ln(u/n) [My Log, Version 2]
 BIC = 359.2884
```

| r     | EIM       |           | z      | P> z  | [95% Conf. Interval] |           |
|-------|-----------|-----------|--------|-------|----------------------|-----------|
|       | Coef.     | Std. Err. |        |       |                      |           |
| ldose | 8.478908  | .4702808  | 18.03  | 0.000 | 7.557175             | 9.400642  |
| _cons | -16.11006 | .8723167  | -18.47 | 0.000 | -17.81977            | -14.40035 |



For a more detailed discussion on user-defined functions, and for an example of a user-defined Newey–West kernel weight, see [Hardin and Hilbe \(2007\)](#).

John Ashworth Nelder was born in Somerset, England, in 1924. He studied mathematics and statistics at Cambridge and worked as a statistician at the National Vegetable Research Station and then Rothamsted Experimental Station. He is especially well known for his contributions to the theory of linear models and statistical computing, as the principal architect of both Genstat and Glim.

Robert William MacLagan Wedderburn (1947–1975) was born in Edinburgh and studied mathematics and statistics at Cambridge. At Rothamsted Experimental Station, he developed the theory of generalized linear models with Nelder and originated the concept of quasilikelihood. He died of anaphylactic shock from an insect bite on a canal holiday.

(Continued on next page)

## Saved results

`glm`, `ml` saves the following in `e()`:

### Scalars

|                             |                                                                |
|-----------------------------|----------------------------------------------------------------|
| <code>e(N)</code>           | number of observations                                         |
| <code>e(k)</code>           | number of parameters                                           |
| <code>e(k_eq)</code>        | number of equations in <code>e(b)</code>                       |
| <code>e(k_eq_model)</code>  | number of equations in model Wald test                         |
| <code>e(k_dv)</code>        | number of dependent variables                                  |
| <code>e(k_autoCns)</code>   | number of base, empty, and omitted constraints                 |
| <code>e(df_m)</code>        | model degrees of freedom                                       |
| <code>e(df)</code>          | residual degrees of freedom                                    |
| <code>e(phi)</code>         | scale parameter                                                |
| <code>e(aic)</code>         | model AIC                                                      |
| <code>e(bic)</code>         | model BIC                                                      |
| <code>e(ll)</code>          | log likelihood, if NR                                          |
| <code>e(N_clust)</code>     | number of clusters                                             |
| <code>e(chi2)</code>        | $\chi^2$                                                       |
| <code>e(p)</code>           | significance                                                   |
| <code>e(deviance)</code>    | deviance                                                       |
| <code>e(deviance_s)</code>  | scaled deviance                                                |
| <code>e(deviance_p)</code>  | Pearson deviance                                               |
| <code>e(deviance_ps)</code> | scaled Pearson deviance                                        |
| <code>e(dispers)</code>     | dispersion                                                     |
| <code>e(dispers_s)</code>   | scaled dispersion                                              |
| <code>e(dispers_p)</code>   | Pearson dispersion                                             |
| <code>e(dispers_ps)</code>  | scaled Pearson dispersion                                      |
| <code>e(nbml)</code>        | 1 if negative binomial parameter estimated via ML, 0 otherwise |
| <code>e(vf)</code>          | factor set by <code>vfactor()</code> , 1 if not set            |
| <code>e(power)</code>       | power set by <code>power()</code> , <code>opower()</code>      |
| <code>e(rank)</code>        | rank of <code>e(V)</code>                                      |
| <code>e(ic)</code>          | number of iterations                                           |
| <code>e(rc)</code>          | return code                                                    |
| <code>e(converged)</code>   | 1 if converged, 0 otherwise                                    |

## Macros

|                                 |                                                                                                     |
|---------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>glm</code>                                                                                    |
| <code>e(cmdline)</code>         | command as typed                                                                                    |
| <code>e(depvar)</code>          | name of dependent variable                                                                          |
| <code>e(varfunc)</code>         | name of variance function used                                                                      |
| <code>e(varfunct)</code>        | Gaussian, Inverse Gaussian, Binomial, Poisson, Neg. Binomial,<br>Bernoulli, Power, or Gamma         |
| <code>e(varfunctf)</code>       | variance function                                                                                   |
| <code>e(link)</code>            | name of link function used                                                                          |
| <code>e(linkt)</code>           | link title                                                                                          |
| <code>e(linkf)</code>           | link form                                                                                           |
| <code>e(m)</code>               | number of binomial trials                                                                           |
| <code>e(wtype)</code>           | weight type                                                                                         |
| <code>e(wexp)</code>            | weight expression                                                                                   |
| <code>e(title)</code>           | title in estimation output                                                                          |
| <code>e(clustvar)</code>        | name of cluster variable                                                                            |
| <code>e(offset)</code>          | offset                                                                                              |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                                             |
| <code>e(cons)</code>            | set if <code>noconstant</code> specified                                                            |
| <code>e(hac_kernel)</code>      | HAC kernel                                                                                          |
| <code>e(hac_lag)</code>         | HAC lag                                                                                             |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                                                      |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                                                       |
| <code>e(opt)</code>             | <code>ml</code> or <code>irls</code>                                                                |
| <code>e(opt1)</code>            | optimization title, line 1                                                                          |
| <code>e(which)</code>           | <code>max</code> or <code>min</code> ; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>       | type of <code>ml</code> method                                                                      |
| <code>e(user)</code>            | name of likelihood-evaluator program                                                                |
| <code>e(technique)</code>       | maximization technique                                                                              |
| <code>e(singularHmethod)</code> | <code>m-marquardt</code> or <code>hybrid</code> ; method used when Hessian is singular              |
| <code>e(crittype)</code>        | optimization criterion                                                                              |
| <code>e(properties)</code>      | <code>b</code> <code>V</code>                                                                       |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                                                      |
| <code>e(marginsnotok)</code>    | predictions disallowed by <code>margins</code>                                                      |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>                                      |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>                                      |

## Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(gradient)</code>     | gradient vector                              |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

`glm`, `irls` saves the following in `e()`:

Scalars

|                             |                                                                |
|-----------------------------|----------------------------------------------------------------|
| <code>e(N)</code>           | number of observations                                         |
| <code>e(k)</code>           | number of parameters                                           |
| <code>e(k_eq_model)</code>  | number of equations in model Wald test                         |
| <code>e(df_m)</code>        | model degrees of freedom                                       |
| <code>e(df)</code>          | residual degrees of freedom                                    |
| <code>e(phi)</code>         | scale parameter                                                |
| <code>e(disps)</code>       | dispersion parameter                                           |
| <code>e(bic)</code>         | model BIC                                                      |
| <code>e(N_clust)</code>     | number of clusters                                             |
| <code>e(deviance)</code>    | deviance                                                       |
| <code>e(deviance_s)</code>  | scaled deviance                                                |
| <code>e(deviance_p)</code>  | Pearson deviance                                               |
| <code>e(deviance_ps)</code> | scaled Pearson deviance                                        |
| <code>e(dispers)</code>     | dispersion                                                     |
| <code>e(dispers_s)</code>   | scaled dispersion                                              |
| <code>e(dispers_p)</code>   | Pearson dispersion                                             |
| <code>e(dispers_ps)</code>  | scaled Pearson dispersion                                      |
| <code>e(nbml)</code>        | 1 if negative binomial parameter estimated via ML, 0 otherwise |
| <code>e(vf)</code>          | factor set by <code>vfactor()</code> , 1 if not set            |
| <code>e(power)</code>       | power set by <code>power()</code> , <code>opower()</code>      |
| <code>e(rank)</code>        | rank of <code>e(V)</code>                                      |
| <code>e(rc)</code>          | return code                                                    |



## Macros

|                              |                                                                                          |
|------------------------------|------------------------------------------------------------------------------------------|
| <code>e(cmd)</code>          | <code>glm</code>                                                                         |
| <code>e(cmdline)</code>      | command as typed                                                                         |
| <code>e(depvar)</code>       | name of dependent variable                                                               |
| <code>e(varfunc)</code>      | name of variance function used                                                           |
| <code>e(varfunct)</code>     | Gaussian, Inverse Gaussian, Binomial, Poisson, Neg. Binomial, Bernoulli, Power, or Gamma |
| <code>e(varfuncf)</code>     | variance function                                                                        |
| <code>e(link)</code>         | name of link function used                                                               |
| <code>e(linkt)</code>        | link title                                                                               |
| <code>e(linkf)</code>        | link form                                                                                |
| <code>e(m)</code>            | number of binomial trials                                                                |
| <code>e(wtype)</code>        | weight type                                                                              |
| <code>e(wexp)</code>         | weight expression                                                                        |
| <code>e(clustvar)</code>     | name of cluster variable                                                                 |
| <code>e(offset)</code>       | offset                                                                                   |
| <code>e(cons)</code>         | set if <code>noconstant</code> specified                                                 |
| <code>e(hac_kernel)</code>   | HAC kernel                                                                               |
| <code>e(hac_lag)</code>      | HAC lag                                                                                  |
| <code>e(vce)</code>          | <i>vcetype</i> specified in <code>vce()</code>                                           |
| <code>e(vcetype)</code>      | title used to label Std. Err.                                                            |
| <code>e(opt)</code>          | ml or irls                                                                               |
| <code>e(opt1)</code>         | optimization title, line 1                                                               |
| <code>e(opt2)</code>         | optimization title, line 2                                                               |
| <code>e(crittype)</code>     | optimization criterion                                                                   |
| <code>e(properties)</code>   | <code>b V</code>                                                                         |
| <code>e(predict)</code>      | program used to implement <code>predict</code>                                           |
| <code>e(marginsnotok)</code> | predictions disallowed by <code>margins</code>                                           |
| <code>e(asbalanced)</code>   | factor variables <code>fvset</code> as <code>asbalanced</code>                           |
| <code>e(asobserved)</code>   | factor variables <code>fvset</code> as <code>asobserved</code>                           |

## Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`glm` is implemented as an ado-file.

The canonical reference on GLM is [McCullagh and Nelder \(1989\)](#). The term “generalized linear model” is from [Nelder and Wedderburn \(1972\)](#). Many people use the acronym GLIM for GLM models because of the classic GLM software tool GLIM, by [Baker and Nelder \(1985\)](#). See [Dobson and Barnett \(2008\)](#) for a concise introduction and overview. See [Rabe-Hesketh and Everitt \(2007\)](#) for more examples of GLM using Stata. [Hoffmann \(2004\)](#) focuses on applying generalized linear models, using real-world datasets, along with interpreting computer output, which for the most part is obtained using Stata.

This discussion highlights the details of parameter estimation and predicted statistics. For a more detailed treatment, and for information on variance estimation, see [Hardin and Hilbe \(2007\)](#). `glm` supports estimation with survey data. For details on VCEs with survey data, see [\[SVY\] variance estimation](#).

`glm` obtains results by IRLS, as described in [McCullagh and Nelder \(1989\)](#), or by maximum likelihood using Newton–Raphson. The implementation here, however, allows user-specified weights, which we denote as  $v_j$  for the  $j$ th observation. Let  $M$  be the number of “observations” ignoring weights. Define

$$w_j = \begin{cases} 1 & \text{if no weights are specified} \\ v_j & \text{if `fweights` or `weights` are specified} \\ Mv_j/(\sum_k v_k) & \text{if `aweight` or `pweight` are specified} \end{cases}$$

The number of observations is then  $N = \sum_j w_j$  if `fweights` are specified and  $N = M$  otherwise. Each IRLS step is performed by `regress` using  $w_j$  as the weights.

Let  $d_j^2$  denote the squared deviance residual for the  $j$ th observation:

For the Gaussian family,  $d_j^2 = (y_j - \hat{\mu}_j)^2$ .

For the Bernoulli family (binomial with denominator 1),

$$d_j^2 = \begin{cases} -2\ln(1 - \hat{\mu}_j) & \text{if } y_j = 0 \\ -2\ln(\hat{\mu}_j) & \text{otherwise} \end{cases}$$

For the binomial family with denominator  $m_j$ ,

$$d_j^2 = \begin{cases} 2y_j\ln(y_j/\hat{\mu}_j) + 2(m_j - y_j)\ln\{(m_j - y_j)/(m_j - \hat{\mu}_j)\} & \text{if } 0 < y_j < m_j \\ 2m_j\ln\{m_j/(m_j - \hat{\mu}_j)\} & \text{if } y_j = 0 \\ 2y_j\ln(y_j/\hat{\mu}_j) & \text{if } y_j = m_j \end{cases}$$

For the Poisson family,

$$d_j^2 = \begin{cases} 2\hat{\mu}_j & \text{if } y_j = 0 \\ 2\{y_j\ln(y_j/\hat{\mu}_j) - (y_j - \hat{\mu}_j)\} & \text{otherwise} \end{cases}$$

For the gamma family,  $d_j^2 = -2\{\ln(y_j/\hat{\mu}_j) - (y_j - \hat{\mu}_j)/\hat{\mu}_j\}$ .

For the inverse Gaussian,  $d_j^2 = (y_j - \hat{\mu}_j)^2/(\hat{\mu}_j^2 y_j)$ .

For the negative binomial,

$$d_j^2 = \begin{cases} 2\ln(1 + k\hat{\mu}_j)/k & \text{if } y_j = 0 \\ 2y_j\ln(y_j/\hat{\mu}_j) - 2\{(1 + ky_j)/k\}\ln\{(1 + ky_j)/(1 + k\hat{\mu}_j)\} & \text{otherwise} \end{cases}$$

Let  $\phi = 1$  if the scale parameter is set to one; otherwise, define  $\phi = \hat{\phi}_0(n - k)/n$ , where  $\hat{\phi}_0$  is the estimated scale parameter and  $k$  is the number of covariates in the model (including intercept).

Let  $\ln L_j$  denote the log likelihood for the  $j$ th observation:

For the Gaussian family,

$$\ln L_j = -\frac{1}{2} \left[ \left\{ \frac{(y_j - \hat{\mu}_j)^2}{\phi} \right\} + \ln(2\pi\phi) \right]$$

For the binomial family with denominator  $m_j$  (Bernoulli if all  $m_j = 1$ ),

$$\ln L_j = \phi \times \begin{cases} \ln\{\Gamma(m_j + 1)\} - \ln\{\Gamma(y_j + 1)\} + \ln\{\Gamma(m_j - y_j + 1)\} & \text{if } 0 < y_j < m_j \\ \quad + (m_j - y_j) \ln(1 - \hat{\mu}_j/m_j) + y_j \ln(\hat{\mu}_j/m_j) & \\ m_j \ln(1 - \hat{\mu}_j/m_j) & \text{if } y_j = 0 \\ m_j \ln(\hat{\mu}_j/m_j) & \text{if } y_j = m_j \end{cases}$$

For the Poisson family,

$$\ln L_j = \phi [y_j \ln(\hat{\mu}_j) - \hat{\mu}_j - \ln\{\Gamma(y_j + 1)\}]$$

For the gamma family,  $\ln L_j = -y_j/\hat{\mu}_j + \ln(1/\hat{\mu}_j)$ .

For the inverse Gaussian,

$$\ln L_j = -\frac{1}{2} \left\{ \frac{(y_j - \hat{\mu}_j)^2}{y_j \hat{\mu}_j^2} + 3 \ln(y_j) + \ln(2\pi) \right\}$$

For the negative binomial (let  $m = 1/k$ ),

$$\begin{aligned} \ln L_j = & \phi [\ln\{\Gamma(m + y_j)\} - \ln\{\Gamma(y_j + 1)\} - \ln\{\Gamma(m)\} \\ & - m \ln(1 + \hat{\mu}_j/m) + y_j \ln\{\hat{\mu}_j/(\hat{\mu}_j + m)\}] \end{aligned}$$

The overall deviance reported by **glm** is  $D^2 = \sum_j w_j d_j^2$ . The dispersion of the deviance is  $D^2$  divided by the residual degrees of freedom.

The Akaike information criterion (AIC) and Bayesian information criterion (BIC) are given by

$$\begin{aligned} \text{AIC} &= \frac{-2 \ln L + 2k}{N} \\ \text{BIC} &= D^2 - (N - k) \ln(N) \end{aligned}$$

where  $\ln L = \sum_j w_j \ln L_j$  is the overall log likelihood.

The Pearson deviance reported by **glm** is  $\sum_j w_j r_j^2$ . The corresponding Pearson dispersion is the Pearson deviance divided by the residual degrees of freedom. **glm** also calculates the scaled versions of all these quantities by dividing by the estimated scale parameter.

## Acknowledgments

glm was written by James Hardin, University of South Carolina, and Joseph Hilbe, Arizona State University. The previous version of this routine was written by Patrick Royston, MRC Clinical Trials Unit, London. The original version of this routine was published in [Royston \(1994\)](#). Royston's work, in turn, was based on a prior implementation by Joseph Hilbe, first published in [Hilbe \(1993\)](#). Roger Newson wrote an early implementation ([Newson 1999](#)) of robust variance estimates for GLM. Parts of this entry are excerpts from [Hardin and Hilbe \(2007\)](#).

## References

- Akaike, H. 1973. Information theory and an extension of the maximum likelihood principle. In *Second International Symposium on Information Theory*, ed. B. N. Petrov and F. Csaki, 267–281. Budapest: Akailseoniai–Kiudo.
- Anscombe, F. J. 1953. Contribution of discussion paper by H. Hotelling “New light on the correlation coefficient and its transforms”. *Journal of the Royal Statistical Society, Series B* 15: 229–230.
- Baker, R. J., and J. A. Nelder. 1985. *The Generalized Linear Interactive Modelling System, Release 3.77*. Oxford: Numerical Algorithms Group.
- Basu, A. 2005. [Extended generalized linear models: Simultaneous estimation of flexible link and variance functions](#). *Stata Journal* 5: 501–516.
- Berndt, E. K., B. H. Hall, R. E. Hall, and J. A. Hausman. 1974. Estimation and inference in nonlinear structural models. *Annals of Economic and Social Measurement* 3/4: 653–665.
- Dobson, A. J., and A. G. Barnett. 2008. *An Introduction to Generalized Linear Models*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC.
- Hardin, J. W., and J. M. Hilbe. 2007. *Generalized Linear Models and Extensions*. 2nd ed. College Station, TX: Stata Press.
- Hilbe, J. M. 1993. [sg16: Generalized linear models](#). *Stata Technical Bulletin* 11: 20–28. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 149–159. College Station, TX: Stata Press.
- . 2000. [sg126: Two-parameter log-gamma and log-inverse Gaussian models](#). *Stata Technical Bulletin* 53: 31–32. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 273–275. College Station, TX: Stata Press.
- . 2009. *Logistic Regression Models*. Boca Raton, FL: Chapman & Hill/CRC.
- Hoffmann, J. P. 2004. *Generalized Linear Models: An Applied Approach*. Boston: Pearson.
- Hosmer Jr., D. W., and S. Lemeshow. 2000. *Applied Logistic Regression*. 2nd ed. New York: Wiley.
- McCullagh, P., and J. A. Nelder. 1989. *Generalized Linear Models*. 2nd ed. London: Chapman & Hall/CRC.
- Nelder, J. A. 1975. Robert William MacLagan Wedderburn, 1947–1975. *Journal of the Royal Statistical Society, Series A* 138: 587.
- Nelder, J. A., and R. W. M. Wedderburn. 1972. Generalized linear models. *Journal of the Royal Statistical Society, Series A* 135: 370–384.
- Newey, W. K., and K. D. West. 1987. A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica* 55: 703–708.
- Newson, R. 1999. [sg114: rglim—Robust variance estimates for generalized linear models](#). *Stata Technical Bulletin* 50: 27–33. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 181–190. College Station, TX: Stata Press.
- . 2004. [Generalized power calculations for generalized linear models and more](#). *Stata Journal* 4: 379–401.
- Pregibon, D. 1980. Goodness of link tests for generalized linear models. *Applied Statistics* 29: 15–24.
- Rabe-Hesketh, S., and B. S. Everitt. 2007. *A Handbook of Statistical Analyses Using Stata*. 4th ed. Boca Raton, FL: Chapman & Hall/CRC.
- Rabe-Hesketh, S., A. Pickles, and C. Taylor. 2000. [sg129: Generalized linear latent and mixed models](#). *Stata Technical Bulletin* 53: 47–57. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 293–307. College Station, TX: Stata Press.
- Rabe-Hesketh, S., A. Skrondal, and A. Pickles. 2004. [st0005: Reliable estimation of generalized linear mixed models using adaptive quadrature](#). *Stata Journal* 2: 1–21.

- Raftery, A. 1995. Bayesian model selection in social research. In Vol. 25 of *Sociological Methodology*, ed. P. V. Marsden, 111–163. Oxford: Blackwell.
- Rogers, W. H. 1993. [sg16.4: Comparison of nbreg and glm for negative binomial](#). *Stata Technical Bulletin* 16: 7. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 82–84. College Station, TX: Stata Press.
- Royston, P. 1994. [sg22: Generalized linear models: Revision of glm](#). *Stata Technical Bulletin* 18: 6–11. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 112–121. College Station, TX: Stata Press.
- Schonlau, M. 2005. [Boosted regression \(boosting\): An introductory tutorial and a Stata plugin](#). *Stata Journal* 5: 330–354.
- Senn, S. J. 2003. A conversation with John Nelder. *Statistical Science* 18: 118–131.

## Also see

- [R] [glm postestimation](#) — Postestimation tools for glm
- [R] [cloglog](#) — Complementary log-log regression
- [R] [logistic](#) — Logistic regression, reporting odds ratios
- [R] [nbreg](#) — Negative binomial regression
- [R] [poisson](#) — Poisson regression
- [R] [regress](#) — Linear regression
- [SVY] [svy estimation](#) — Estimation commands for survey data
- [XT] [xtgee](#) — Fit population-averaged panel-data models by using GEE
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `glm`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>            | link test for model specification                                                                   |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

# Syntax for predict

```
predict [type] newvar [if] [in] [, statistic options]
```

| statistic             | description                                                                    |
|-----------------------|--------------------------------------------------------------------------------|
| Main                  |                                                                                |
| <u>mu</u>             | expected value of $y$ ; the default                                            |
| <u>xb</u>             | linear prediction $\eta = \mathbf{x}\hat{\beta}$                               |
| <u>eta</u>            | synonym of <u>xb</u>                                                           |
| <u>stdp</u>           | standard error of the linear prediction                                        |
| <u>anscombe</u>       | <a href="#">Anscombe (1953)</a> residuals                                      |
| <u>cooks</u> <u>d</u> | Cook's distance                                                                |
| <u>deviance</u>       | deviance residuals                                                             |
| <u>hat</u>            | diagonals of the “hat” matrix                                                  |
| <u>likelihood</u>     | a weighted average of standardized deviance and standardized Pearson residuals |
| <u>pearson</u>        | Pearson residuals                                                              |
| <u>response</u>       | differences between the observed and fitted outcomes                           |
| <u>score</u>          | first derivative of the log likelihood with respect to $\mathbf{x}_j\beta$     |
| <u>working</u>        | working residuals                                                              |

| options             | description                                                                            |
|---------------------|----------------------------------------------------------------------------------------|
| Options             |                                                                                        |
| <u>nooffset</u>     | modify calculations to ignore offset variable                                          |
| <u>adjusted</u>     | adjust deviance residual to speed up convergence                                       |
| <u>standardized</u> | multiply residual by the factor $(1 - h)^{-1/2}$                                       |
| <u>studentized</u>  | multiply residual by one over the square root of the estimated scale parameter         |
| <u>modified</u>     | modify denominator of residual to be a reasonable estimate of the variance of $depvar$ |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

`mu`, `xb`, `stdp`, and `score` are the only statistics allowed with `svy` estimation results.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

`mu`, the default, specifies that `predict` calculate the expected value of  $y$ , equal to  $g^{-1}(\mathbf{x}\hat{\beta})$  [ $ng^{-1}(\mathbf{x}\hat{\beta})$  for the binomial family].

`xb` calculates the linear prediction  $\eta = \mathbf{x}\hat{\beta}$ .

`eta` is a synonym for `xb`.

`stdp` calculates the standard error of the linear prediction.

`anscombe` calculates the [Anscombe \(1953\)](#) residuals to produce residuals that closely follow a normal distribution.

**cooks** calculates Cook’s distance, which measures the aggregate change in the estimated coefficients when each observation is left out of the estimation.

**deviance** calculates the deviance residuals. Deviance residuals are recommended by [McCullagh and Nelder \(1989\)](#) and by others as having the best properties for examining the goodness of fit of a GLM. They are approximately normally distributed if the model is correct. They may be plotted against the fitted values or against a covariate to inspect the model’s fit. Also see the **pearson** option below.

**hat** calculates the diagonals of the “hat” matrix as an analog to simple linear regression.

**likelihood** calculates a weighted average of standardized deviance and standardized Pearson residuals.

**pearson** calculates the Pearson residuals. Pearson residuals often have markedly skewed distributions for nonnormal family distributions. Also see the **deviance** option above.

**response** calculates the differences between the observed and fitted outcomes.

**score** calculates the equation-level score,  $\partial \ln L / \partial (\mathbf{x}_j \beta)$ .

**working** calculates the working residuals, which are response residuals weighted according to the derivative of the link function.

#### Options

**nooffset** is relevant only if you specified **offset(varname)** for **glm**. It modifies the calculations made by **predict** so that they ignore the offset variable; the linear prediction is treated as  $\mathbf{x}_j \mathbf{b}$  rather than as  $\mathbf{x}_j \mathbf{b} + \text{offset}_j$ .

**adjusted** adjusts the deviance residual to speed up the convergence to the limiting normal distribution. The adjustment deals with adding to the deviance residual a higher-order term that depends on the variance function family. This option is allowed only when **deviance** is specified.

**standardized** requests that the residual be multiplied by the factor  $(1 - h)^{-1/2}$ , where  $h$  is the diagonal of the hat matrix. This operation is done to account for the correlation between *depvar* and its predicted value.

**studentized** requests that the residual be multiplied by one over the square root of the estimated scale parameter.

**modified** requests that the denominator of the residual be modified to be a reasonable estimate of the variance of *depvar*. The base residual is multiplied by the factor  $(k/w)^{-1/2}$ , where  $k$  is either one or the user-specified dispersion parameter and  $w$  is the specified weight (or one if left unspecified).

## Remarks

Remarks are presented under the following headings:

*Predictions*

*Other postestimation commands*

## Predictions

### ► Example 1

After **glm** estimation, **predict** may be used to obtain various predictions based on the model. In [example 2](#) of [\[R\] glm](#), we mentioned that the complementary log-log link seemed to fit the data better than the logit link. Now we go back and obtain the fitted values and deviance residuals:



```

. use http://www.stata-press.com/data/r11/ldose
. glm r ldose, f(binomial n) l(logit)
 (output omitted)
. predict mu_logit
(option mu assumed; predicted mean r)
. predict dr_logit, deviance
. quietly glm r ldose, f(binomial n) l(cloglog)
. predict mu_cl
(option mu assumed; predicted mean r)
. predict dr_cl, d
. format mu_logit dr_logit mu_cl dr_cl %9.5f
. list r mu_logit dr_logit mu_cl dr_cl, sep(4)

```

|    | r  | mu_logit | dr_logit | mu_cl    | dr_cl    |
|----|----|----------|----------|----------|----------|
| 1. | 6  | 3.45746  | 1.28368  | 5.58945  | 0.18057  |
| 2. | 13 | 9.84167  | 1.05969  | 11.28067 | 0.55773  |
| 3. | 18 | 22.45139 | -1.19611 | 20.95422 | -0.80330 |
| 4. | 28 | 33.89761 | -1.59412 | 30.36942 | -0.63439 |
| 5. | 52 | 50.09584 | 0.60614  | 47.77644 | 1.28883  |
| 6. | 53 | 53.29092 | -0.12716 | 54.14273 | -0.52366 |
| 7. | 61 | 59.22216 | 1.25107  | 61.11331 | -0.11878 |
| 8. | 60 | 58.74297 | 1.59398  | 59.94723 | 0.32495  |

In six of the eight cases,  $|\text{dr\_logit}| > |\text{dr\_cl}|$ . The above represents only one of the many available options for `predict`. See [Hardin and Hilbe \(2007\)](#) for a more in-depth examination.



## Other postestimation commands

### □ Technical note

After `glm` estimation, you may perform any of the postestimation commands that you would perform after any other kind of estimation in Stata; see [\[U\] 20 Estimation and postestimation commands](#). Below we test the joint significance of all the interaction terms.

```

. use http://www.stata-press.com/data/r11/beetle, clear
. glm r beetle#c.ldose, f(bin n) l(cloglog)
 (output omitted)
. testparm i.beetle beetle#c.ldose
(1) [r]2.beetle = 0
(2) [r]3.beetle = 0
(3) [r]2.beetle#c.ldose = 0
(4) [r]3.beetle#c.ldose = 0

 chi2(4) = 249.69
 Prob > chi2 = 0.0000

```

If you wanted to print the variance–covariance matrix of the estimators, you would type `estat vce`.

If you use the `linktest` postestimation command, you must also specify the `family()` and `link()` options; see [\[R\] linktest](#).



## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

We follow the terminology used in [Methods and formulas](#) of [R] **glm**.

The deviance residual calculated by `predict` following **glm** is  $d_j = \text{sign}(y_j - \hat{\mu}_j) \sqrt{d_j^2}$ .

The Pearson residual calculated by `predict` following **glm** is

$$r_j = \frac{y_j - \hat{\mu}_j}{\sqrt{V(\hat{\mu}_j)}}$$

where  $V(\hat{\mu}_j)$  is the family-specific variance function.

$$V(\hat{\mu}_j) = \begin{cases} \hat{\mu}_j(1 - \hat{\mu}_j/m_j) & \text{if binomial or Bernoulli } (m_j = 1) \\ \hat{\mu}_j^2 & \text{if gamma} \\ 1 & \text{if Gaussian} \\ \hat{\mu}_j^3 & \text{if inverse Gaussian} \\ \hat{\mu}_j + k\hat{\mu}_j^2 & \text{if negative binomial} \\ \hat{\mu}_j & \text{if Poisson} \end{cases}$$

The response residuals are given by  $r_i^R = y_i - \mu_i$ . The working residuals are

$$r_i^W = (y_i - \hat{\mu}_i) \left( \frac{\partial \eta}{\partial \mu} \right)_i$$

and the score residuals are

$$r_i^S = \frac{y_i - \hat{\mu}_i}{V(\hat{\mu}_i)} \left( \frac{\partial \eta}{\partial \mu} \right)_i^{-1}$$

Define  $\widehat{W} = V(\widehat{\mu})$  and  $X$  to be the covariate matrix.  $h_i$ , then, is the  $i$ th diagonal of the hat matrix given by

$$\widehat{H} = \widehat{W}^{1/2} X (X^T \widehat{W} X)^{-1} X^T \widehat{W}^{1/2}$$

As a result, the likelihood residuals are given by

$$r_i^L = \text{sign}(y_i - \hat{\mu}_i) \left\{ h_i (r_i')^2 + (1 - h_i) (d_i')^2 \right\}^{1/2}$$

where  $r_i'$  and  $d_i'$  are the standardized Pearson and standardized deviance residuals, respectively. By *standardized*, we mean that the residual is divided by  $\{1 - h_i\}^{1/2}$ .

Cook's distance is an overall measure of the change in the regression coefficients caused by omitting the  $i$ th observation from the analysis. Computationally, Cook's distance is obtained as

$$C_i = \frac{(r_i')^2 h_i}{k(1 - h_i)}$$

where  $k$  is the number of regressors, including the constant.

Anscombe residuals are given by

$$r_i^A = \frac{A(y_i) - A(\hat{\mu}_i)}{A'(\hat{\mu}_i)\{V(\hat{\mu}_i)\}^{1/2}}$$

where

$$A(\cdot) = \int \frac{d\mu}{V^{1/3}(\mu)}$$

Deviance residuals may be adjusted (`predict, adjusted`) to make the following correction:

$$d_i^a = d_i + \frac{1}{6}\rho_3(\theta)$$

where  $\rho_3(\theta)$  is a family-specific correction. See [Hardin and Hilbe \(2007\)](#) for the exact forms of  $\rho_3(\theta)$  for each family.

## References

- Anscombe, F. J. 1953. Contribution of discussion paper by H. Hotelling “New light on the correlation coefficient and its transforms”. *Journal of the Royal Statistical Society, Series B* 15: 229–230.
- Hardin, J. W., and J. M. Hilbe. 2007. *Generalized Linear Models and Extensions*. 2nd ed. College Station, TX: Stata Press.
- McCullagh, P., and J. A. Nelder. 1989. *Generalized Linear Models*. 2nd ed. London: Chapman & Hall/CRC.

## Also see

- [R] [glm](#) — Generalized linear models
- [R] [regress postestimation](#) — Postestimation tools for regress
- [U] [20 Estimation and postestimation commands](#)

Syntax

Logistic regression for grouped data

```
blogit pos_var pop_var [rhsvars] [if] [in] [, blogit_options]
```

Probit regression for grouped data

```
bprobit pos_var pop_var [rhsvars] [if] [in] [, bprobit_options]
```

Weighted least-squares logistic regression for grouped data

```
glogit pos_var pop_var [rhsvars] [if] [in] [, glogit_options]
```

Weighted least-squares probit regression for grouped data

```
gprobit pos_var pop_var [rhsvars] [if] [in] [, gprobit_options]
```

| blogit_options                   | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| Model                            |                                                                                                     |
| <u>noconstant</u>                | suppress constant term                                                                              |
| <u>asis</u>                      | retain perfect predictor variables                                                                  |
| <u>offset</u> (varname)          | include varname in model with coefficient constrained to 1                                          |
| <u>constraints</u> (constraints) | apply specified linear constraints                                                                  |
| <u>collinear</u>                 | keep collinear variables                                                                            |
| SE/Robust                        |                                                                                                     |
| <u>vce</u> (vcetype)             | vcetype may be oim, <u>robust</u> , <u>cluster</u> clustvar, <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting                        |                                                                                                     |
| <u>level</u> (#)                 | set confidence level; default is level(95)                                                          |
| <u>or</u>                        | report odds ratios                                                                                  |
| <u>nocnsreport</u>               | do not display constraints                                                                          |
| <u>display_options</u>           | control spacing and display of omitted variables and base and empty cells                           |
| Maximization                     |                                                                                                     |
| <u>maximize_options</u>          | control the maximization process; seldom used                                                       |
| <sup>†</sup> <u>nocoef</u>       | do not display coefficient table; seldom used                                                       |
| <sup>†</sup> <u>coeflegend</u>   | display coefficients' legend instead of coefficient table                                           |

| <i>bprobit_options</i>                        | description                                                                                                                |
|-----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| Model                                         |                                                                                                                            |
| <b><u>noconstant</u></b>                      | suppress constant term                                                                                                     |
| <b><u>asis</u></b>                            | retain perfect predictor variables                                                                                         |
| <b><u>offset</u>(<i>varname</i>)</b>          | include <i>varname</i> in model with coefficient constrained to 1                                                          |
| <b><u>constraints</u>(<i>constraints</i>)</b> | apply specified linear constraints                                                                                         |
| <b><u>collinear</u></b>                       | keep collinear variables                                                                                                   |
| SE/Robust                                     |                                                                                                                            |
| <b><u>vce</u>(<i>vcetype</i>)</b>             | <i>vcetype</i> may be <u>oim</u> , <u>robust</u> , <u>cluster</u> <i>clustvar</i> , <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting                                     |                                                                                                                            |
| <b><u>level</u>(#)</b>                        | set confidence level; default is <u>level</u> (95)                                                                         |
| <b><u>nocnsreport</u></b>                     | do not display constraints                                                                                                 |
| <b><u>display_options</u></b>                 | control spacing and display of omitted variables and base and empty cells                                                  |
| Maximization                                  |                                                                                                                            |
| <b><u>maximize_options</u></b>                | control the maximization process; seldom used                                                                              |
| <sup>†</sup> <b><u>nocoef</u></b>             | do not display coefficient table; seldom used                                                                              |
| <sup>†</sup> <b><u>coeflegend</u></b>         | display coefficients' legend instead of coefficient table                                                                  |
| <i>glogit_options</i>                         | description                                                                                                                |
| SE                                            |                                                                                                                            |
| <b><u>vce</u>(<i>vcetype</i>)</b>             | <i>vcetype</i> may be <u>ols</u> , <u>bootstrap</u> , or <u>jackknife</u>                                                  |
| Reporting                                     |                                                                                                                            |
| <b><u>level</u>(#)</b>                        | set confidence level; default is <u>level</u> (95)                                                                         |
| <b><u>or</u></b>                              | report odds ratios                                                                                                         |
| <b><u>display_options</u></b>                 | control spacing and display of omitted variables and base and empty cells                                                  |
| <sup>†</sup> <b><u>coeflegend</u></b>         | display coefficients' legend instead of coefficient table                                                                  |
| <i>gprobit_options</i>                        | description                                                                                                                |
| SE                                            |                                                                                                                            |
| <b><u>vce</u>(<i>vcetype</i>)</b>             | <i>vcetype</i> may be <u>ols</u> , <u>bootstrap</u> , or <u>jackknife</u>                                                  |
| Reporting                                     |                                                                                                                            |
| <b><u>level</u>(#)</b>                        | set confidence level; default is <u>level</u> (95)                                                                         |
| <b><u>display_options</u></b>                 | control spacing and display of omitted variables and base and empty cells                                                  |
| <sup>†</sup> <b><u>coeflegend</u></b>         | display coefficients' legend instead of coefficient table                                                                  |

<sup>†</sup>`_nocoef` and `_coeflegend` do not appear in the dialog box.

`rhsvars` may contain factor variables; see [U] 11.4.3 [Factor variables](#).

`bootstrap`, `by`, `jackknife`, `rolling`, and `statsby` are allowed; see [U] 11.1.10 [Prefix commands](#).

See [U] 20 [Estimation and postestimation commands](#) for more capabilities of estimation commands.

## Menu

### **blogit**

Statistics > Binary outcomes > Grouped data > Logit regression for grouped data

### **bprobit**

Statistics > Binary outcomes > Grouped data > Probit regression for grouped data

### **glogit**

Statistics > Binary outcomes > Grouped data > Weighted least-squares logit regression

### **gprobit**

Statistics > Binary outcomes > Grouped data > Weighted least-squares probit regression

## Description

`blogit` and `bprobit` produce maximum-likelihood logit and probit estimates on grouped (“blocked”) data; `glogit` and `gprobit` produce weighted least-squares estimates. In the syntax diagrams above, *pos\_var* and *pop\_var* refer to variables containing the total number of positive responses and the total population.

See [R] [logistic](#) for a list of related estimation commands.

## Options for `blogit` and `bprobit`

### Model

`noconstant`; see [R] [estimation options](#).

`asis` forces retention of perfect predictor variables and their associated perfectly predicted observations and may produce instabilities in maximization; see [R] [probit](#).

`offset(varname)`, `constraints(constraints)`, `collinear`; see [R] [estimation options](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

### Reporting

`level(#)`; see [R] [estimation options](#).

or (b`logit` only) reports the estimated coefficients transformed to odds ratios, i.e.,  $e^b$  rather than  $b$ . Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated. `or` may be specified at estimation or when replaying previously estimated results.

`nocnsreport`; see [R] [estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

#### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [R] [maximize](#). These options are seldom used.

The following options are available with `blogit` and `bprobit` but are not shown in the dialog box: `nocoeff` specifies that the coefficient table not be displayed. This option is sometimes used by program writers but is useless interactively.

`coeflegend`; see [R] [estimation options](#).

## Options for glogit and gprobit

#### SE

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

`vce(ols)`, the default, uses the standard variance estimator for ordinary least-squares regression.

#### Reporting

`level(#)`; see [R] [estimation options](#).

or (g`logit` only) reports the estimated coefficients transformed to odds ratios, i.e.,  $e^b$  rather than  $b$ . Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated. `or` may be specified at estimation or when replaying previously estimated results.

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

The following option is available with `glogit` and `gprobit` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

Remarks are presented under the following headings:

*Maximum likelihood estimates*

*Weighted least-squares estimates*

Maximum likelihood estimates

blogit produces the same results as logit and logistic, and bprobit produces the same results as probit, but the “blocked” commands accept data in a slightly different “shape”. Consider the following two datasets:

```
. use http://www.stata-press.com/data/r11/xmpl1
. list, sepby(agecat)
```

|    | agecat | exposed | died | pop |
|----|--------|---------|------|-----|
| 1. | 0      | 0       | 0    | 115 |
| 2. | 0      | 0       | 1    | 5   |
| 3. | 0      | 1       | 0    | 98  |
| 4. | 0      | 1       | 1    | 8   |
| 5. | 1      | 0       | 0    | 69  |
| 6. | 1      | 0       | 1    | 16  |
| 7. | 1      | 1       | 0    | 76  |
| 8. | 1      | 1       | 1    | 22  |

```
. use http://www.stata-press.com/data/r11/xmpl2
. list
```

|    | agecat | exposed | deaths | pop |
|----|--------|---------|--------|-----|
| 1. | 0      | 0       | 5      | 120 |
| 2. | 0      | 1       | 8      | 106 |
| 3. | 1      | 0       | 16     | 85  |
| 4. | 1      | 1       | 22     | 98  |

These two datasets contain the same information; observations 1 and 2 of xmpl1 correspond to observation 1 of xmpl2, observations 3 and 4 of xmpl1 correspond to observation 2 of xmpl2, and so on.

The first observation of xmpl1 says that for agecat==0 and exposed==0, 115 subjects did not die (died==0). The second observation says that for the same agecat and exposed groups, five subjects did die (died==1). In xmpl2, the first observation says that there were five deaths of a population of 120 in agecat==0 and exposed==0. These are two different ways of saying the same thing. Both datasets are transcriptions from the following table, reprinted in Rothman, Greenland, and Lash (2008, 260), for age-specific deaths from all causes for tolbutamide and placebo treatment groups (University Group Diabetes Program 1970):

|           | Age through 54 |         | Age 55 and above |         |
|-----------|----------------|---------|------------------|---------|
|           | Tolbutamide    | Placebo | Tolbutamide      | Placebo |
| Dead      | 8              | 5       | 22               | 16      |
| Surviving | 98             | 115     | 76               | 79      |

The data in xmpl1 are said to be “fully relational”, which is computer jargon meaning that each observation corresponds to one cell of the table. Stata typically prefers data in this format. The second form of storing these data in xmpl2 is said to be “folded”, which is computer jargon for something less than fully relational.

blogit and bprobit deal with “folded” data and produce the same results that logit and probit would have if the data had been stored in the “fully relational” representation.



## ► Example 1

For the tolbutamide data, the fully relational representation is preferred. We could then use `logistic`, `logit`, and any of the epidemiological table commands; see [R] [logistic](#), [R] [logit](#), and [ST] [epitab](#). Nevertheless, there are occasions when the folded representation seems more natural. With `blogit` and `bprobit`, we avoid the tedium of having to unfold the data:

```
. use http://www.stata-press.com/data/r11/xmpl12
. blogit deaths pop agecat exposed, or
Logistic regression for grouped data
```

|               |   |        |
|---------------|---|--------|
| Number of obs | = | 409    |
| LR chi2(2)    | = | 22.47  |
| Prob > chi2   | = | 0.0000 |
| Pseudo R2     | = | 0.0730 |

```
Log likelihood = -142.6212
```

| _outcome | Odds Ratio | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|----------|------------|-----------|------|-------|----------------------|----------|
| agecat   | 4.216299   | 1.431519  | 4.24 | 0.000 | 2.167361             | 8.202223 |
| exposed  | 1.404674   | .4374454  | 1.09 | 0.275 | .7629451             | 2.586175 |

If we had not specified the `or` option, results would have been presented as coefficients instead of as odds ratios. The estimated odds ratio of death for tolbutamide exposure is 1.40, although the 95% confidence interval includes 1. (By comparison, these data, in fully relational form and analyzed using the `cs` command [see [ST] [epitab](#)], produce a Mantel–Haenszel weighted odds ratio of 1.40 with a 95% confidence interval of 0.76 to 2.59.)

We can see the underlying coefficients by replaying the estimation results and not specifying the `or` option:

```
. blogit
Logistic regression for grouped data
```

|               |   |        |
|---------------|---|--------|
| Number of obs | = | 409    |
| LR chi2(2)    | = | 22.47  |
| Prob > chi2   | = | 0.0000 |
| Pseudo R2     | = | 0.0730 |

```
Log likelihood = -142.6212
```

| _outcome | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| agecat   | 1.438958  | .3395203  | 4.24  | 0.000 | .7735101             | 2.104405  |
| exposed  | .3398053  | .3114213  | 1.09  | 0.275 | -.2705692            | .9501798  |
| _cons    | -2.968471 | .33234    | -8.93 | 0.000 | -3.619846            | -2.317097 |

◀

## ► Example 2

`bprobit` works like `blogit`, substituting the probit for the logit-likelihood function.

```
. bprobit deaths pop agecat exposed
Probit regression for grouped data
```

|               |   |        |
|---------------|---|--------|
| Number of obs | = | 409    |
| LR chi2(2)    | = | 22.58  |
| Prob > chi2   | = | 0.0000 |
| Pseudo R2     | = | 0.0734 |

```
Log likelihood = -142.56478
```

| _outcome | Coef.     | Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|----------|-----------|-----------|--------|-------|----------------------|-----------|
| agecat   | .7542049  | .1709692  | 4.41   | 0.000 | .4191114             | 1.089298  |
| exposed  | .1906236  | .1666059  | 1.14   | 0.253 | -.1359179            | .5171651  |
| _cons    | -1.673973 | .1619594  | -10.34 | 0.000 | -1.991408            | -1.356539 |

◀

Weighted least-squares estimates

Example 3

We have state data for the United States on the number of marriages (`marriage`), the total population aged 18 years or more (`pop18p`), and the median age (`medage`). The dataset excludes Nevada, so it has 49 observations. We now wish to estimate a logit equation for the marriage rate. We will include age squared by specifying the term `c.medage#c.medage`:

```
. use http://www.stata-press.com/data/r11/census7
(1980 Census data by state)

. glogit marriage pop18p medage c.medage#c.medage
```

Weighted LS logistic regression for grouped data

| Source   | SS         | df | MS         | Number of obs = | 49     |
|----------|------------|----|------------|-----------------|--------|
| Model    | .71598314  | 2  | .35799157  | F( 2, 46) =     | 12.89  |
| Residual | 1.27772858 | 46 | .027776708 | Prob > F =      | 0.0000 |
|          |            |    |            | R-squared =     | 0.3591 |
|          |            |    |            | Adj R-squared = | 0.3313 |
| Total    | 1.99371172 | 48 | .041535661 | Root MSE =      | .16666 |

|                   | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|-------------------|-----------|-----------|-------|-------|----------------------|-----------|
| medage            | -.6459349 | .2828381  | -2.28 | 0.027 | -1.215258            | -.0766114 |
| c.medage#c.medage | .0095414  | .0046608  | 2.05  | 0.046 | .0001598             | .0189231  |
| _cons             | 6.503833  | 4.288977  | 1.52  | 0.136 | -2.129431            | 15.1371   |



Example 4

We could just as easily have fit a grouped-probit model by typing `gprobit` rather than `glogit`:

```
. gprobit marriage pop18p medage c.medage#c.medage
```

Weighted LS probit regression for grouped data

| Source   | SS         | df | MS         | Number of obs = | 49     |
|----------|------------|----|------------|-----------------|--------|
| Model    | .108222962 | 2  | .054111481 | F( 2, 46) =     | 12.94  |
| Residual | .192322476 | 46 | .004180923 | Prob > F =      | 0.0000 |
|          |            |    |            | R-squared =     | 0.3601 |
|          |            |    |            | Adj R-squared = | 0.3323 |
| Total    | .300545438 | 48 | .006261363 | Root MSE =      | .06466 |

|                   | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|-------------------|-----------|-----------|-------|-------|----------------------|-----------|
| medage            | -.2755007 | .1121042  | -2.46 | 0.018 | -.5011548            | -.0498466 |
| c.medage#c.medage | .0041082  | .0018422  | 2.23  | 0.031 | .0004001             | .0078163  |
| _cons             | 2.357708  | 1.704446  | 1.38  | 0.173 | -1.073164            | 5.788579  |



## Saved results

`blogit` and `bprobit` save the following in `e()`:

### Scalars

|                            |                                                |
|----------------------------|------------------------------------------------|
| <code>e(N)</code>          | number of observations                         |
| <code>e(N_cds)</code>      | number of completely determined successes      |
| <code>e(N_cdf)</code>      | number of completely determined failures       |
| <code>e(k)</code>          | number of parameters                           |
| <code>e(k_eq)</code>       | number of equations                            |
| <code>e(k_eq_model)</code> | number of equations in model Wald test         |
| <code>e(k_dv)</code>       | number of dependent variables                  |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints |
| <code>e(df_m)</code>       | model degrees of freedom                       |
| <code>e(r2_p)</code>       | pseudo- <i>R</i> -squared                      |
| <code>e(ll)</code>         | log likelihood                                 |
| <code>e(ll_0)</code>       | log likelihood, constant-only model            |
| <code>e(N_clust)</code>    | number of clusters                             |
| <code>e(chi2)</code>       | $\chi^2$                                       |
| <code>e(p)</code>          | significance of model test                     |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                      |
| <code>e(ic)</code>         | number of iterations                           |
| <code>e(rc)</code>         | return code                                    |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                    |

### Macros

|                                 |                                                                          |
|---------------------------------|--------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>blogit</code> or <code>bprobit</code>                              |
| <code>e(cmdline)</code>         | command as typed                                                         |
| <code>e(depvar)</code>          | name of dependent variable                                               |
| <code>e(wtype)</code>           | weight type                                                              |
| <code>e(wexp)</code>            | weight expression                                                        |
| <code>e(title)</code>           | title in estimation output                                               |
| <code>e(clustvar)</code>        | name of cluster variable                                                 |
| <code>e(offset)</code>          | offset                                                                   |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                  |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                           |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                            |
| <code>e(opt)</code>             | type of optimization                                                     |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>       | type of ml method                                                        |
| <code>e(user)</code>            | name of likelihood-evaluator program                                     |
| <code>e(technique)</code>       | maximization technique                                                   |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular              |
| <code>e(crittype)</code>        | optimization criterion                                                   |
| <code>e(properties)</code>      | b V                                                                      |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                           |
| <code>e(marginsok)</code>       | predictions allowed by <code>margins</code>                              |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>           |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>           |

## Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(gradient)</code>     | gradient vector                              |
| <code>e(mns)</code>          | vector of means of the independent variables |
| <code>e(rules)</code>        | information about perfect predictors         |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

`glogit` and `gprobit` save the following in `e()`:

## Scalars

|                      |                             |
|----------------------|-----------------------------|
| <code>e(N)</code>    | number of observations      |
| <code>e(mss)</code>  | model sum of squares        |
| <code>e(df_m)</code> | model degrees of freedom    |
| <code>e(rss)</code>  | residual sum of squares     |
| <code>e(df_r)</code> | residual degrees of freedom |
| <code>e(r2)</code>   | <i>R</i> -squared           |
| <code>e(r2_a)</code> | adjusted <i>R</i> -squared  |
| <code>e(F)</code>    | <i>F</i> statistic          |
| <code>e(rmse)</code> | root mean squared error     |
| <code>e(rank)</code> | rank of <code>e(V)</code>   |

## Macros

|                            |                                                                |
|----------------------------|----------------------------------------------------------------|
| <code>e(cmd)</code>        | <code>glogit</code> or <code>gprobit</code>                    |
| <code>e(cmdline)</code>    | command as typed                                               |
| <code>e(depvar)</code>     | name of dependent variable                                     |
| <code>e(model)</code>      | ols                                                            |
| <code>e(title)</code>      | title in estimation output                                     |
| <code>e(vce)</code>        | <i>vcetype</i> specified in <code>vce()</code>                 |
| <code>e(vcetype)</code>    | title used to label Std. Err.                                  |
| <code>e(properties)</code> | <code>b V</code>                                               |
| <code>e(predict)</code>    | program used to implement <code>predict</code>                 |
| <code>e(marginsok)</code>  | predictions allowed by <code>margins</code>                    |
| <code>e(asbalanced)</code> | factor variables <code>fvset</code> as <code>asbalanced</code> |
| <code>e(asobserved)</code> | factor variables <code>fvset</code> as <code>asobserved</code> |

## Matrices

|                   |                                              |
|-------------------|----------------------------------------------|
| <code>e(b)</code> | coefficient vector                           |
| <code>e(V)</code> | variance–covariance matrix of the estimators |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`blogit`, `bprobit`, `glogit`, and `gprobit` are implemented as ado-files.

Methods and formulas are presented under the following headings:

*Maximum likelihood estimates*

*Weighted least-squares estimates*

## Maximum likelihood estimates

The results reported by `blogit` and `bprobit` are obtained by maximizing a weighted logit- or probit-likelihood function. Let  $F(\cdot)$  denote the normal- or logistic-likelihood function. The likelihood of observing each observation in the data is then

$$F(\beta x)^s \{1 - F(\beta x)\}^{t-s}$$

where  $s$  is the number of successes and  $t$  is the population. The term above is counted as contributing  $s + (t - s) = t$  degrees of freedom. All of this follows directly from the definition of logit and probit.

`blogit` and `bprobit` support the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

## Weighted least-squares estimates

The logit function is defined as the log of the odds ratio. If there is one explanatory variable, the model can be written as

$$\log\left(\frac{p_j}{1 - p_j}\right) = \beta_0 + \beta_1 x_j + \epsilon_j \quad (1)$$

where  $p_j$  represents successes divided by population for the  $j$ th observation. (If there is more than one explanatory variable, we simply interpret  $\beta_1$  as a row vector and  $x_j$  as a column vector.) The large-sample expectation of  $\epsilon_j$  is zero, and its variance is

$$\sigma_j^2 = \frac{1}{n_j p_j (1 - p_j)}$$

where  $n_j$  represents the population for observation  $j$ . We can thus apply weighted least-squares to the observations, with weights proportional to  $n_j p_j (1 - p_j)$ .

As in any feasible generalized least-squares problem, estimation proceeds in two steps. First, we fit (1) by OLS and compute the predicted probabilities as

$$\hat{p}_j = \frac{\exp(\widehat{\beta}_0 + \widehat{\beta}_1 x_j)}{1 + \exp(\widehat{\beta}_0 + \widehat{\beta}_1 x_j)}$$

In the second step, we fit (1) by using analytic weights equal to  $n_j \hat{p}_j (1 - \hat{p}_j)$ .

For `gprobit`, write  $\Phi(\cdot)$  for the cumulative normal distribution, and define  $z_j$  implicitly by  $\Phi(z_j) = p_j$ , where  $p_j$  is the fraction of successes for observation  $j$ . The probit model for one explanatory variable can be written as

$$\Phi^{-1}(p_j) = \beta_0 + \beta_1 x_j + \epsilon_j$$

(If there is more than one explanatory variable, we simply interpret  $\beta_1$  as a row vector and  $x_j$  as a column vector.)

The expectation of  $\epsilon_j$  is zero, and its variance is given by

$$\sigma_j^2 = \frac{p_j(1 - p_j)}{n_j \phi^2\{\Phi^{-1}(p_j)\}}$$

where  $\phi(\cdot)$  represents the normal density (Amemiya 1981, 1498). We can thus apply weighted least squares to the observations with weights proportional to  $1/\sigma_j^2$ . As for grouped logit, we use a two-step estimator to obtain the weighted least-squares estimates.

## References

- Amemiya, T. 1981. Qualitative response models: A survey. *Journal of Economic Literature* 19: 1483–1536.
- Hosmer Jr., D. W., and S. Lemeshow. 2000. *Applied Logistic Regression*. 2nd ed. New York: Wiley.
- Judge, G. G., W. E. Griffiths, R. C. Hill, H. Lütkepohl, and T.-C. Lee. 1985. *The Theory and Practice of Econometrics*. 2nd ed. New York: Wiley.
- Rothman, K. J., S. Greenland, and T. L. Lash. 2008. *Modern Epidemiology*. 3rd ed. Philadelphia: Lippincott Williams & Wilkins.
- University Group Diabetes Program. 1970. A study of the effects of hypoglycemic agents on vascular complications in patients with adult-onset diabetes, II: Mortality results. *Diabetes* 19, supplement 2: 789–830.

## Also see

- [R] **glogit postestimation** — Postestimation tools for glogit, gprobit, blogit, and bprobit
- [R] **logistic** — Logistic regression, reporting odds ratios
- [R] **logit** — Logistic regression, reporting coefficients
- [R] **probit** — Probit regression
- [R] **scobit** — Skewed logistic regression
- [U] **20 Estimation and postestimation commands**

Description

The following postestimation commands are available for `glogit`, `gprobit`, `blogit`, and `bprobit`:

| command                | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| * <code>estat</code>   | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estimates</code> | cataloging estimation results                                                                       |
| <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| * <code>lrtest</code>  | likelihood-ratio test                                                                               |
| <code>margins</code>   | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>   | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code> | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

\* `estat ic` and `lrtest` are not appropriate after `glogit` and `gprobit`.

See the corresponding entries in the *Base Reference Manual* for details.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic]
```

| statistic         | description                             |
|-------------------|-----------------------------------------|
| Main              |                                         |
| <code>n</code>    | predicted count; the default            |
| <code>pr</code>   | probability of a positive outcome       |
| <code>xb</code>   | linear prediction                       |
| <code>stdp</code> | standard error of the linear prediction |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

**n**, the default, calculates the expected count, that is, the estimated probability times *pop\_var*, which is the total population.

**pr** calculates the predicted probability of a positive outcome.

**xb** calculates the linear prediction.

**stdp** calculates the standard error of the linear prediction.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [glogit](#) — Logit and probit regression for grouped data

[U] [20 Estimation and postestimation commands](#)



## Syntax

### Interactive version

```
gmm ([eqname1:]<mexp1>) ([eqname2:]<mexp2>) ... [if] [in] [weight] [, options]
```

### Moment-evaluator program version

```
gmm moment_prog [if] [in] [weight] , { equations(namelist) | nequations(#) }
 { parameters(namelist) | nparameters(#) } [options] [program_options]
```

where

*mexp<sub>j</sub>* is the substitutable expression for the *j*th moment equation and

*moment\_prog* is a moment-evaluator program.

| <i>options</i> | description |
|----------------|-------------|
|----------------|-------------|

#### Model

derivative(<*dexp<sub>mn</sub>*>) specify derivative of *mexp<sub>m</sub>* with respect to parameter *n*; can be specified more than once (interactive version only)

\* twostep

use two-step GMM estimator; the default

\* onestep

use one-step GMM estimator

\* igmm

use iterative GMM estimator

#### Instruments

instruments([<*eqlist*>:] *varlist* [, noconstant])  
specify instruments; can be specified more than once

xtinstruments([<*eqlist*>:] *varlist*, lags(#<sub>1</sub>/#<sub>2</sub>))  
specify panel-style instruments; can be specified more than once

#### Weight matrix

wmatrix(*wmtype* [, independent])  
specify weight matrix; *wmtype* may be robust, cluster *clustvar*,  
*hac kernel* [*lags*], or unadjusted

center center moments in weight-matrix computation

winitial(*iwtype* [, independent])  
specify initial weight matrix; *iwtype* may be identity,  
unadjusted, *xt xtspec*, or the name of a Stata matrix

---

|                                                         |                                                                                                                                                                             |
|---------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Options                                                 |                                                                                                                                                                             |
| <code>variables(<i>varlist</i>)</code>                  | specify variables in model                                                                                                                                                  |
| <code>nocommonesample</code>                            | do not restrict estimation sample to be the same for all equations                                                                                                          |
| SE/Robust                                               |                                                                                                                                                                             |
| <code>vce(<i>vcetype</i> [, <u>independent</u>])</code> | <i>vcetype</i> may be <u>robust</u> , <u>cluster</u> <i>clustvar</i> , <u>bootstrap</u> ,<br><u>jackknife</u> , <i>hac</i> <i>kernel</i> <i>lags</i> , or <u>unadjusted</u> |
| Reporting                                               |                                                                                                                                                                             |
| <code>level(#)</code>                                   | set confidence level; default is <code>level(95)</code>                                                                                                                     |
| <code>title(<i>string</i>)</code>                       | display <i>string</i> as title above the table of parameter estimates                                                                                                       |
| <code>title2(<i>string</i>)</code>                      | display <i>string</i> as subtitle                                                                                                                                           |
| Optimization                                            |                                                                                                                                                                             |
| <code>from(<i>initial_values</i>)</code>                | specify initial values for parameters                                                                                                                                       |
| <code>† igmmiterate(#)</code>                           | specify maximum number of iterations for iterated GMM estimator                                                                                                             |
| <code>† igmmeps(#)</code>                               | specify # for iterated GMM parameter convergence criterion;<br>default is <code>igmmeps(1e-6)</code>                                                                        |
| <code>† igmmweps(#)</code>                              | specify # for iterated GMM weight-matrix convergence criterion;<br>default is <code>igmmweps(1e-6)</code>                                                                   |
| <code>optimization_options</code>                       | control the optimization process; seldom used                                                                                                                               |
| <code>† <u>coeflegend</u></code>                        | display coefficients' legend instead of coefficient table                                                                                                                   |

---

\* You can specify at most one of these options.

† These options may be specified only when `igmm` is specified.

† `coeflegend` does not appear in the dialog box.

---

| <i>program_options</i> | description |
|------------------------|-------------|
|------------------------|-------------|

---

|                                            |                                                                 |
|--------------------------------------------|-----------------------------------------------------------------|
| Model                                      |                                                                 |
| <code>evaluator_options</code>             | additional options to be passed to the moment-evaluator program |
| <code>hasderivatives</code>                | moment-evaluator program can calculate derivatives              |
| * <code>equations(<i>namelist</i>)</code>  | specify moment-equation names                                   |
| * <code>nequations(#)</code>               | specify number of moment equations                              |
| † <code>parameters(<i>namelist</i>)</code> | specify parameter names                                         |
| † <code>nparameters(#)</code>              | specify number of parameters                                    |

---

\* You must specify `equations(namelist)` or `nequations(#)`; you may specify both.

† You must specify `parameters(namelist)` or `nparameters(#)`; you may specify both.

`bootstrap`, `by`, `jackknife`, `rolling`, `statsby`, and `xi` are allowed; see [U] 11.1.10 Prefix commands. Weights are not allowed with the `bootstrap` prefix.

`aweight`s are not allowed with the `jackknife` prefix.

`aweight`s, `fweight`s, and `pweight`s are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for additional capabilities of estimation commands.

$\langle mexp_j \rangle$  and  $\langle dexp_{mn} \rangle$  are extensions of valid Stata expressions that also contain parameters to be estimated. The parameters are enclosed in curly braces and must otherwise satisfy the naming

requirements for variables; {beta} is an example of a parameter. Also allowed is a notation of the form {<eqname>:varlist} for linear combinations of multiple covariates and their parameters. For example, {xb: mpg price turn} defines a linear combination of the variables mpg, price, and turn. See [Substitutable expressions](#) under *Remarks* below.

## Menu

Statistics > Endogenous covariates > Generalized method of moments estimation

## Description

**gmm** performs generalized method of moments (GMM) estimation. With the interactive version of the command, you enter the moment equations directly into the dialog box or on the command line using substitutable expressions. The moment-evaluator program version gives you greater flexibility in exchange for increased complexity; with this version, you write a program in an ado-file that calculates the moments based on a vector of parameters passed to it.

**gmm** can fit both single- and multiple-equation models, and it allows moment conditions of the form  $E\{\mathbf{z}_i u_i(\beta)\} = \mathbf{0}$ , where  $\mathbf{z}_i$  is a vector of instruments and  $u_i(\beta)$  is often an additive regression error term, as well as more general moment conditions of the form  $E\{\mathbf{h}_i(\mathbf{z}_i; \beta)\} = \mathbf{0}$ . **gmm** works with cross-sectional, time-series, and longitudinal (panel) data.

## Options

### Model

**derivative**([eqname | #]/name = <dexp<sub>mn</sub>>) specifies the derivative of moment equation eqname or # with respect to parameter name. If eqname or # is not specified, **gmm** assumes that the derivative applies to the first moment equation.

For a moment equation of the form  $E\{\mathbf{z}_{mi} u_{mi}(\beta)\} = \mathbf{0}$ , **derivative**(m/β<sub>j</sub> = <dexp<sub>mn</sub>>) is to contain a substitutable expression for  $\partial u_{mi} / \partial \beta_j$ .

For a moment equation of the form  $E\{h_{mi}(\mathbf{z}_i; \beta)\} = \mathbf{0}$ , **derivative**(m/β<sub>j</sub> = <dexp<sub>mn</sub>>) is to contain a substitutable expression for  $\partial h_{mi} / \partial \beta_j$ .

<dexp<sub>mn</sub>> uses the same substitutable expression syntax as is used to specify moment equations. If you declare a linear combination in a moment equation, you provide the derivative for the linear combination; **gmm** then applies the chain rule for you. See [Specifying derivatives](#) under *Remarks* below for examples.

If you do not specify the **derivative**() option, **gmm** calculates derivatives numerically. You must either specify no derivatives or specify all the derivatives that are not identically zero; you cannot specify some analytic derivatives and have **gmm** compute the rest numerically.

**twostep**, **onestep**, and **igmm** specify which estimator is to be used. You can specify at most one of these options. **twostep** is the default.

**twostep** requests the two-step GMM estimator. **gmm** obtains parameter estimates based on the initial weight matrix, computes a new weight matrix based on those estimates, and then reestimates the parameters based on that weight matrix.

**onestep** requests the one-step GMM estimator. The parameters are estimated based on an initial weight matrix, and no updating of the weight matrix is performed except when calculating the appropriate variance–covariance (VCE) matrix.

`igmm` requests the iterative GMM estimator. `gmm` obtains parameter estimates based on the initial weight matrix, computes a new weight matrix based on those estimates, reestimates the parameters based on that weight matrix, computes a new weight matrix, and so on, to convergence. Convergence is declared when the relative change in the parameter vector is less than `igmmeps()`, the relative change in the weight matrix is less than `igmmweps()`, or `igmmiterate()` iterations have been completed. Hall (2005, sec. 2.4 and 3.6) mentions that there may be gains to finite-sample efficiency from using the iterative estimator.

#### Instruments

`instruments([<eqlist>:]varlist[, noconstant])` specifies a list of instrumental variables to be used. If you specify a single moment equation, then you do not need to specify the equations to which the instruments apply; you can omit the `eqlist` and simply specify `instruments(varlist)`. By default, a constant term is included in `varlist`; to omit the constant term, use the `noconstant` suboption: `instruments(varlist, noconstant)`.

If you specify a model with multiple moment conditions of the form

$$E \begin{Bmatrix} \mathbf{z}_{1i}'\mathbf{u}_{1i}(\beta) \\ \dots \\ \mathbf{z}_{qi}'\mathbf{u}_{qi}(\beta) \end{Bmatrix} = \mathbf{0}$$

then you can specify the equations to indicate the moment equations for which the list of variables is to be used as instruments if you do not want that list applied to all the moment equations. For example, you might type

```
gmm (main:<mexp1>) (<mexp2>) (<mexp3>), instruments(z1 z2) ///
 instruments(2: z3) instruments(main 3: z4)
```

Variables `z1` and `z2` will be used as instruments for all three equations, `z3` will be used as an instrument for the second equation, and `z4` will be used as an instrument for the first and third equations. Notice that we chose to supply a name for the first moment equation but not the second two.

`xtinstruments([<eqlist>:]varlist, lags(#1/#2))` is for use with panel-data models in which the set of available instruments depends on the time period. As with `instruments()`, you can prefix the list of variables with equation names or numbers to target instruments to specific equations. Unlike with `instruments()`, a constant term is not included in `varlist`. You must `xtset` your data before using this option; see [XT] `xtset`.

If you specify

```
gmm ..., xtinstruments(x, lags(1/.)) ...
```

then for panel  $i$  and period  $t$ , `gmm` uses as instruments  $x_{i,t-1}, x_{i,t-2}, \dots, x_{i1}$ . More generally, specifying `xtinstruments(x, lags(#1, #2))` uses as instruments  $x_{i,t-\#1}, \dots, x_{i,t-\#2}$ ; setting `#2 = .` requests all available lags. `#1` and `#2` must be zero or positive integers.

`gmm` automatically excludes observations for which no valid instruments are available. It does, however, include observations for which only a subset of the lags is available. For example, if you request that lags one through three be used, then `gmm` will include the observations for the second and third time periods even though fewer than three lags are available as instruments.

#### Weight matrix

`wmatrix(wmtype[, independent])` specifies the type of weight matrix to be used in conjunction with the two-step and iterated GMM estimators.

Specifying `wmatrix(robust)` requests a weight matrix that is appropriate when the errors are independent but not necessarily identically distributed. `wmatrix(robust)` is the default.

Specifying `wmatrix(cluster clustvar)` requests a weight matrix that accounts for arbitrary correlation among observations within clusters identified by *clustvar*.

Specifying `wmatrix(hac kernel #)` requests a heteroskedasticity- and autocorrelation-consistent (HAC) weight matrix using the specified kernel (see below) with *#* lags. The bandwidth of a kernel is equal to the number of lags plus one.

Specifying `wmatrix(hac kernel opt)` requests an HAC weight matrix using the specified kernel, and the lag order is selected using Newey and West's (1994) optimal lag-selection algorithm.

Specifying `wmatrix(hac kernel)` requests an HAC weight matrix using the specified kernel and  $N - 2$  lags, where  $N$  is the sample size.

There are three kernels available for HAC weight matrices, and you may request each one by using the name used by statisticians or the name perhaps more familiar to economists:

`bartlett` or `nwest` requests the Bartlett (Newey–West) kernel;

`parzen` or `gallant` requests the Parzen (Gallant) kernel; and

`quadraticspectral` or `andrews` requests the quadratic spectral (Andrews) kernel.

Specifying `wmatrix(unadjusted)` requests a weight matrix that is suitable when the errors are homoskedastic. In some applications, the GMM estimator so constructed is known as the (nonlinear) two-stage least-squares (2SLS) estimator.

Including the `independent` suboption creates a weight matrix that assumes moment equations are independent. This suboption is often used to replicate other models that can be motivated outside the GMM framework, such as the estimation of a system of equations by system-wide 2SLS. This suboption has no effect if only one moment equation is specified.

`wmatrix()` has no effect if `onestep` is also specified.

`center` requests that the sample moments be centered (demeaned) when computing GMM weight matrices. By default, centering is not done.

`winitial(wmtype[, independent])` specifies the weight matrix to use to obtain the first-step parameter estimates.

Specifying `winitial(unadjusted)` requests a weighting matrix that assumes the moment equations are independent and identically distributed. This matrix is of the form  $(\mathbf{Z}'\mathbf{Z})^{-1}$ , where  $\mathbf{Z}$  represents all the instruments specified in the `instruments()` option. To avoid a singular weight matrix, you should specify at least  $q - 1$  moment equations of the form  $E\{\mathbf{z}_{hi}u_{hi}(\beta)\} = \mathbf{0}$ , where  $q$  is the number of moment equations, or you should specify the `independent` suboption.

Including the `independent` suboption creates a weight matrix that assumes moment equations are independent. Elements of the weight matrix corresponding to covariances between two moment equations are set equal to zero. This suboption has no effect if only one moment equation is specified.

`winitial(unadjusted)` is the default.

`winitial(xt xtspec)` is for use with dynamic panel-data models in which one of the moment equations is specified in first-differences form. *xtspec* is a string consisting of the letters “L” and “D”, the length of which is equal to the number of moment equations in the model. You specify “L” for a moment equation if that moment equation is written in levels, and you specify “D” for a moment equation if it is written in first-differences; *xtspec* is not case sensitive. When you specify

this option, you can specify at most one moment equation in levels and one moment equation in first-differences. See the examples listed in [Dynamic panel-data models](#) under the *Remarks* below.

`winitial(identity)` requests that the identity matrix be used.

`winitial(matname)` requests that Stata matrix *matname* be used. You cannot specify the `independent` suboption if you specify `winitial(matname)`.

#### Options

`variables(varlist)` specifies the variables in the model. `gmm` ignores observations for which any of these variables has a missing value. If you do not specify `variables()`, then `gmm` assumes all the observations are valid and issues an error message with return code 480 if any moment equations evaluate to missing for any observations at the initial value of the parameter vector.

`nocommonesample` requests that `gmm` not restrict the estimation sample to be the same for all equations. By default, `gmm` will restrict the estimation sample to observations that are available for all equations in the model, mirroring the behavior of other multiple-equation estimators such as `nlsur`, `suresg`, or `reg3`. For certain models, however, different equations can have different numbers of observations. For these models, you should specify `nocommonesample`. See [Dynamic panel-data models](#) below for one application of this option. You cannot specify weights if you specify `nocommonesample`.

#### SE/Robust

`vce(vcetype[, independent])` specifies the type of standard error reported, which includes types that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [\[R\] vce\\_option](#).

`vce(unadjusted)` specifies that an unadjusted (nonrobust) VCE matrix be used; this, along with the `twostep` option, results in the “optimal two-step GMM” estimates often discussed in textbooks.

The default `vcetype` is based on the `wmtype` specified in the `wmatrix()` option. If `wmatrix()` is specified but `vce()` is not, then `vcetype` is set equal to `wmtype`. To override this behavior and obtain an unadjusted (nonrobust) VCE matrix, specify `vce(unadjusted)`.

Specifying `vce(bootstrap)` or `vce(jackknife)` results in standard errors based on the bootstrap or jackknife, respectively. See [\[R\] vce\\_option](#), [\[R\] bootstrap](#), and [\[R\] jackknife](#) for more information on these VCEs.

The syntax for `vcetypes` other than `bootstrap` and `jackknife` are identical to those for `wmatrix()`.

#### Reporting

`level(#)`; see [\[R\] estimation options](#).

`title(string)` specifies an optional title that will be displayed just above the table of parameter estimates.

`title2(string)` specifies an optional subtitle that will be displayed between the title specified in `title()` and the table of parameter estimates. If `title2()` is specified but `title()` is not, `title2()` has the same effect as `title()`.

#### Optimization

`from(initial_values)` specifies the initial values to begin the estimation. You can specify a  $1 \times k$  matrix, where  $k$  is the number of parameters in the model, or you can specify a parameter name, its initial value, another parameter name, its initial value, and so on. For example, to initialize `alpha` to 1.23 and `delta` to 4.57, you would type

```
gmm ..., from(alpha 1.23 delta 4.57) ...
```

Initial values declared using this option override any that are declared within substitutable expressions. If you specify a parameter that does not appear in your model, `gmm` exits with error code 480. If you specify a matrix, the values must be in the same order in which the parameters are declared in your model. `gmm` ignores the row and column names of the matrix.

`igmmiterate(#)`, `igmmeps(#)`, and `igmmweps(#)` control the iterative process for the iterative GMM estimator. These options can be specified only if you also specify `igmm`.

`igmmiterate(#)` specifies the maximum number of iterations to perform with the iterative GMM estimator. The default is the number set using `set maxiter`, which is 16,000 by default. To use this option, you must also specify the `igmm` option.

`igmmeps(#)` specifies the convergence criterion used for successive parameter estimates when the iterative GMM estimator is used. The default is `igmmeps(1e-6)`. Convergence is declared when the relative difference between successive parameter estimates is less than `igmmeps()` and the relative difference between successive estimates of the weight matrix is less than `igmmweps()`. To use this option, you must also specify the `igmm` option.

`igmmweps(#)` specifies the convergence criterion used for successive estimates of the weight matrix when the iterative GMM estimator is used. The default is `igmmweps(1e-6)`. To use this option, you must also specify the `igmm` option. Convergence is declared when the relative difference between successive parameter estimates is less than `igmmeps()` and the relative difference between successive estimates of the weight matrix is less than `igmmweps()`. To use this option, you must also specify the `igmm` option.

*optimization\_options*: `technique()`, `conv_maxiter()`, `conv_ptol()`, `conv_vtol()`, `conv_nrtol()`, `tracelevel()`. `technique()` specifies the optimization technique to use; `gn` (the default), `nr`, `dfp`, and `bfgs` are allowed. `conv_maxiter()` specifies the maximum number of iterations; `conv_ptol()`, `conv_vtol()`, and `conv_nrtol()` specify the convergence criteria for the parameters, gradient, and scaled Hessian, respectively. `tracelevel()` allows you to obtain additional details during the iterative process. See [M-5] [optimize\(\)](#).

The following options pertain only to the moment-evaluator program version of `gmm`.

#### Model

*evaluator\_options* refer to any options allowed by your *moment\_prog*.

`hasderivatives` indicates that you have written your moment-evaluator program to compute derivatives. If you do not specify this option, derivatives are computed numerically. If your moment-evaluator program does compute derivatives but you wish to use numerical derivatives instead (perhaps during debugging), do not specify this option.

`equations(namelist)` specifies the names of the moment equations in the model. If you specify both `equations()` and `nequations()`, the number of names in the former must match the number specified in the latter.

`nequations(#)` specifies the number of moment equations in the model. If you do not specify names with the `equations()` option, `gmm` numbers the moment equations 1, 2, 3, .... If you specify both `equations()` and `nequations()`, the number of names in the former must match the number specified in the latter.

`parameters(namelist)` specifies the names of the parameters in the model. The names of the parameters must adhere to the naming conventions of Stata's variables;

see [U] [11.3 Naming conventions](#). If you specify both `parameters()` and `nparameters()`, the number of names in the former must match the number specified in the latter.

`nparameters(#)` specifies the number of parameters in the model. If you do not specify names with the `parameters()` option, `gmm` names them `b1`, `b2`, ..., `b#`. If you specify both `parameters()` and `nparameters()`, the number of names in the former must match the number specified in the latter.

The following option is available with `gmm` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

Remarks are presented under the following headings:

- [Introduction](#)
- [Substitutable expressions](#)
- [The weight matrix and two-step estimation](#)
- [Obtaining standard errors](#)
- [Exponential \(Poisson\) regression models](#)
- [Specifying derivatives](#)
- [Exponential regression models with panel data](#)
- [Rational-expectations models](#)
- [System estimators](#)
- [Dynamic panel-data models](#)
- [Details of moment-evaluator programs](#)

## Introduction

The generalized method of moments (GMM) estimator is a workhorse of modern econometrics and is discussed in all the leading textbooks, including [Cameron and Trivedi \(2005, 2009\)](#), [Davidson and MacKinnon \(1993, 2004\)](#), [Greene \(2008\)](#), [Ruud \(2000\)](#), [Hayashi \(2000\)](#), [Wooldridge \(2002\)](#), [Hamilton \(1994\)](#), and [Baum \(2006\)](#). An excellent treatise on GMM with a focus on time-series applications is [Hall \(2005\)](#). The collection of papers by [Mátyás \(1999\)](#) provides both theoretical and applied aspects of GMM. Here we give a brief introduction to the methodology and emphasize how the various options of `gmm` are used.

The starting point for the generalized method of moments (GMM) estimator is the analogy principle, which says we can estimate a parameter by replacing a population moment condition with its sample analogue. For example, the mean of an independent and identically distributed (i.i.d.) population is defined as the value  $\mu$  such that the first (central) population moment is zero; that is,  $\mu$  solves  $E(y - \mu) = 0$  where  $y$  is a random draw from the population. The analogy principle tells us that to obtain an estimate,  $\hat{\mu}$ , of  $\mu$ , we replace the population-expectations operator with its sample analogue ([Manski 1988](#); [Wooldridge 2002](#)):

$$E(y - \mu) = 0 \longrightarrow \frac{1}{N} \sum_{i=1}^N (y_i - \hat{\mu}) = 0 \longrightarrow \hat{\mu} = \frac{1}{N} \sum_{i=1}^N y_i$$

where  $N$  denotes sample size and  $y_i$  represents the  $i$ th observation of  $y$  in our dataset. The estimator  $\hat{\mu}$  is known as the method of moments (MM) estimator, because we started with a population moment condition and then applied the analogy principle to obtain an estimator that depends on the observed data.



Ordinary least-squares (OLS) regression can also be viewed as an MM estimator. In the model

$$y = \mathbf{x}'\beta + u$$

we assume that  $u$  has mean zero conditional on  $\mathbf{x}$ :  $E(u|\mathbf{x}) = 0$ . This conditional expectation implies the unconditional expectation  $E(\mathbf{x}u) = \mathbf{0}$  because, using the law of iterated expectations,

$$E(\mathbf{x}u) = E_{\mathbf{x}} \{E(\mathbf{x}u|\mathbf{x})\} = E_{\mathbf{x}} \{\mathbf{x} E(u|\mathbf{x})\} = \mathbf{0}$$

(Using the law of iterated expectations to derive unconditional expectations based on conditional expectations, perhaps motivated by subject theory, is extremely common in GMM estimation.) Continuing,

$$E(\mathbf{x}u) = E \{\mathbf{x}(y - \mathbf{x}'\beta)\} = \mathbf{0}$$

Applying the analogy principle,

$$E \{\mathbf{x}(y - \mathbf{x}'\beta)\} \longrightarrow \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i(y_i - \mathbf{x}_i'\beta) = \mathbf{0}$$

so that

$$\hat{\beta} = \left( \sum_i \mathbf{x}_i \mathbf{x}_i' \right)^{-1} \sum_i \mathbf{x}_i y_i$$

which is just the more familiar formula  $\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1} \mathbf{X}'\mathbf{y}$  written using summation notation.

In both the previous examples, the number of parameters we were estimating equaled the number of moment conditions. In the first example, we estimated one parameter,  $\mu$ , and had one moment condition  $E(y - \mu) = 0$ . In the second example, the parameter vector  $\beta$  had  $k$  elements, as did the vector of regressors  $\mathbf{x}$ , yielding  $k$  moment conditions. Ignoring peculiar cases, a model of  $m$  equations in  $m$  unknowns has a unique solution, and because the moment equations in these examples were linear, we were able to solve for the parameters analytically. Had the moment conditions been nonlinear, we would have had to use numerical techniques to solve for the parameters, but that is not a significant limitation with modern computers.

What if we have more moment conditions than parameters? Say we have  $l$  moment conditions and  $k$  parameters. A model of  $l > k$  equations in  $k$  unknowns does not have a unique solution. Any size- $k$  subset of the moment conditions would yield a consistent parameter estimate, though the parameter estimate so obtained would in general be different based on which  $k$  moment conditions we used.

For concreteness, let's return to our regression model,

$$y = \mathbf{x}'\beta + u$$

but we no longer wish to assume that  $E(\mathbf{x}u) = \mathbf{0}$ ; we suspect that the error term  $u$  affects one or more elements of  $\mathbf{x}$ . As a result, we can no longer use the OLS estimator. Suppose we have a vector  $\mathbf{z}$  with the properties that  $E(\mathbf{z}u) = \mathbf{0}$ , that the rank of  $E(\mathbf{z}'\mathbf{z})$  equals  $l$ , and that the rank of  $E(\mathbf{z}'\mathbf{x}) = k$ . The first assumption simply states that  $\mathbf{z}$  is not correlated with the error term. The second assumption rules out perfect collinearity among the elements of  $\mathbf{z}$ . The third assumption, known as the *rank condition* in econometrics, ensures that  $\mathbf{z}$  is sufficiently correlated with  $\mathbf{x}$  and that the estimator is feasible. If some elements of  $\mathbf{x}$  are not correlated with  $u$ , then they should also appear in  $\mathbf{z}$ .

If  $l < k$ , then the rank of  $E(\mathbf{z}'\mathbf{x}) < k$ , violating the rank condition.

If  $l = k$ , then we can use the simpler MM estimator we already discussed; we would obtain what is sometimes called the simple instrumental-variables estimator  $\hat{\beta} = (\sum_i \mathbf{z}_i \mathbf{x}'_i)^{-1} \sum_i \mathbf{z}_i y_i$ . The rank condition ensures that  $\sum_i \mathbf{z}_i \mathbf{x}'_i$  is invertible, at least in the population.

If  $l > k$ , the GMM estimator chooses the value,  $\hat{\beta}$ , that minimizes a quadratic function of the moment conditions. We could define

$$\hat{\beta} \equiv \arg \min_{\beta} \left\{ \frac{1}{N} \sum_i \mathbf{z}_i u_i(\beta) \right\}' \left\{ \frac{1}{N} \sum_i \mathbf{z}_i u_i(\beta) \right\} \quad (1)$$

where for our linear regression example  $u_i(\beta) = y_i - \mathbf{x}'_i \beta$ . This estimator tries to make the moment conditions as close to zero as possible. This simple estimator, however, applies equal weight to each of the moment conditions; and as we shall see later, we can obtain more efficient estimators by choosing to weight some moment conditions more highly than others.

Consider the quadratic function

$$Q(\beta) = \left\{ \frac{1}{N} \sum_i \mathbf{z}_i u_i(\beta) \right\}' \mathbf{W} \left\{ \frac{1}{N} \sum_i \mathbf{z}_i u_i(\beta) \right\}$$

where  $\mathbf{W}$  is a symmetric positive-definite matrix known as a weight matrix. Then we define the GMM estimator as

$$\hat{\beta} \equiv \arg \min_{\beta} Q(\beta) \quad (2)$$

Continuing with our regression model example, if we choose

$$\mathbf{W} = \left( \frac{1}{N} \sum_i \mathbf{z}_i \mathbf{z}'_i \right)^{-1} \quad (3)$$

then we obtain

$$\hat{\beta} = \left\{ \left( \frac{1}{N} \sum_i \mathbf{x}_i \mathbf{z}'_i \right) \left( \frac{1}{N} \sum_i \mathbf{z}_i \mathbf{z}'_i \right)^{-1} \left( \frac{1}{N} \sum_i \mathbf{z}_i \mathbf{x}'_i \right) \right\}^{-1} \times \left( \frac{1}{N} \sum_i \mathbf{x}_i \mathbf{z}'_i \right) \left( \frac{1}{N} \sum_i \mathbf{z}_i \mathbf{z}'_i \right)^{-1} \left( \frac{1}{N} \sum_i \mathbf{z}_i y_i \right)$$

which is the well-known two-stage least-squares (2SLS) estimator. Our choice of weight matrix here was based on the assumption that  $u$  was homoskedastic. A feature of GMM estimation is that by selecting different weight matrices, we can obtain estimators that can tolerate heteroskedasticity, clustering, autocorrelation, and other features of  $u$ . See [R] [ivregress](#) for more information about the 2SLS and linear GMM estimators.

Returning to the case where the model is “just identified”, meaning that  $l = k$ , if we apply the GMM estimator, we will obtain the same estimate,  $\hat{\beta}$ , regardless of our choice of  $\mathbf{W}$ . Because  $l = k$ , if a unique solution exists, it will set all the sample moment conditions to zero jointly, so  $\mathbf{W}$  has no impact on the value of  $\beta$  that minimizes the objective function.

We will highlight other features of the GMM estimator and the `gmm` command as we proceed through examples. First, though, we discuss how to specify moment equations by using substitutable expressions.

## Substitutable expressions

To use the interactive version of `gmm`, you define the moment equations by using substitutable expressions. In most applications, your moment conditions are of the form  $E\{z_i u_i(\beta)\}$ , where  $u_i(\beta)$  is a residual term that depends on the parameter vector  $\beta$  as well as variables in your dataset, though we suppress expressing the variables for notational simplicity; we refer to  $u_i(\beta)$  as the moment equation to differentiate it from the moment conditions  $E\{z_i' u_i(\beta)\} = 0$ .

Substitutable expressions in `gmm` work much like those used in `nl` and `nlSUR`, though with one important difference. For the latter two commands, you type the name of the dependent variable, an equal sign, and then the regression function. For example, in `nl`, if you want to fit the function  $y = f(\mathbf{x}; \beta) + u$ , you would type

```
nl (y = <expression for f(x; beta)>), ...
```

On the other hand, `gmm` requires you to write a substitutable expression for  $u$ ; in this example,  $u = y - f(\mathbf{x}; \beta)$ , so you would type

```
gmm (y - <expression for f(x; beta)>), ...
```

The advantage of writing the substitutable expression directly in terms of  $u$  is that you are not restricted to fitting models with additive error terms as you are with `nl` and `nlSUR`.

You specify substitutable expressions just like any other mathematical expression involving scalars and variables, such as those you would use with Stata's `generate` command, except that the parameters to be estimated are bound in braces. See [U] 13.2 Operators and [U] 13.3 Functions for more information on expressions. Parameter names must follow the same conventions as variable names. See [U] 11.3 Naming conventions.

For example, say that the  $t$ th observation on a sample moment is

$$u_t = 1 - \beta \left\{ (1 + r_{t+1})(c_{t+1}/c_t)^{-\gamma} \right\}$$

where  $t$  denotes time period,  $\beta$  and  $\gamma$  are the parameters to be estimated, and  $r$  and  $c$  are variables in your dataset. Then you would type

```
gmm (1 - {beta}*((1 + F.r)*(F.c/c)^(-1*{gamma}))), ...
```

Because  $\beta$  and  $\gamma$  are parameters, we enclose them in braces. Also notice our use of the forward operator to refer to the values of  $r$  and  $c$  one period ahead; time-series operators are allowed in substitutable expressions as long as you have previously `tsset` your data. See [U] 13.9 Time-series operators for more information on time-series operators.

To specify initial values for some parameters, you can include an equal sign and the initial value after a parameter:

```
gmm (1 - {beta}*((1 + F.r)*(F.c/c)^(-1*{gamma=1}))), ...
```

would initialize  $\gamma$  to be one. If you do not specify an initial value for a parameter, it is initialized to zero.

Frequently, even nonlinear functions contain linear combinations of variables. As an example, suppose you have this moment equation:

$$u = \{y - \exp(\beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3)\} / \exp(\beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3)$$

Instead of typing

```
gmm ((y - exp({beta1}*x1 + {beta2}*x2 + {beta3}*x3)) / ///
 exp({beta1}*x1 + {beta2}*x2 + {beta3}*x3)) ...
```

you can type

```
gmm ((y - exp({xb:x1 x2 x3})) / exp({xb:}))
```

The notation `{xb:x1 x2 x3}` tells `gmm` that you want a linear combination of the variables `x1`, `x2`, and `x3`. We named this linear combination `xb`, so `gmm` will name the three parameters corresponding to the three variables `xb_x1`, `xb_x2`, and `xb_x3`. You can name the linear combination anything you wish (subject to Stata's naming conventions for variable names); `gmm` then names the parameter corresponding to variable `x` `lc_x`, where `lc` is the name of your linear combination. You cannot use the same name for both an individual parameter and a linear combination. You can, however, refer to one parameter in a linear combination after it has been declared as you would any other parameter by using the notation `{lc_x}`. Linear combinations do not include a constant term.

Once we have declared the variables in the linear combination `xb`, we can subsequently refer to the linear combination in our substitutable expression by using the notation `xb:`. The colon is not optional; it tells `gmm` that you are referring to a previously declared linear combination, not an individual parameter. This shorthand notation is also handy when specifying derivatives, as we will show later.

In general, there are three rules to follow when defining substitutable expressions:

1. Parameters of the model are bound in braces: `{b0}`, `{param}`, etc.
2. Initial values for parameters are given by including an equal sign and the initial value inside the braces: `{b0=1}`, `{param=3.571}`, etc.
3. Linear combinations of variables can be included using the notation `{eqname:varlist}`: `{xb:mpg price weight}`, `{score: w x z}`, etc. Parameters of linear combinations are initialized to zero.

If you specify initial values by using the `from()` option, they override whatever initial values are given within the substitutable expression. Substitutable expressions are so named because, once values are assigned to the parameters, the resulting expressions can be handled by `generate` and `replace`.

## ► Example 1: OLS regression

In [Introduction](#), we stated that OLS is an MM estimator. Say that we want to fit the model

$$\text{mpg} = \beta_0 + \beta_1 \text{weight} + \beta_2 \text{length} + u$$

where  $u$  is an i.i.d. error term. We type

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. gmm (mpg - {b1}*weight - {b2}*length - {b0}), instruments(weight length)

Step 1
Iteration 0: GMM criterion Q(b) = 475.4138
Iteration 1: GMM criterion Q(b) = 2.696e-20
Iteration 2: GMM criterion Q(b) = 3.329e-27

Step 2
Iteration 0: GMM criterion Q(b) = 5.109e-28
Iteration 1: GMM criterion Q(b) = 7.237e-32
```

GMM estimation

Number of parameters = 3

Number of moments = 3

Initial weight matrix: Unadjusted

Number of obs = 74

GMM weight matrix: Robust

|     | Coef.     | Robust<br>Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-----|-----------|---------------------|-------|-------|----------------------|-----------|
| /b1 | -.0038515 | .0019472            | -1.98 | 0.048 | -.0076678            | -.0000351 |
| /b2 | -.0795935 | .0677528            | -1.17 | 0.240 | -.2123866            | .0531996  |
| /b0 | 47.88487  | 7.50599             | 6.38  | 0.000 | 33.1734              | 62.59634  |

Instruments for equation 1: weight length \_cons

Recall that the moment condition for OLS regression is  $E(xu) = \mathbf{0}$ , where  $\mathbf{x}$ , the list of instruments, is the same as the list of regressors in the model. In our command, we defined the residual term,  $u$ , inside parentheses by using a substitutable expression; because linear combinations declared in substitutable expressions do not include a constant term, we included our own (b0). Inside the `instruments()` option, we listed our instruments; by default, `gmm` includes a constant term among the instrument list.

Because the number of moments equals the number of parameters we are estimating, the model is said to be “just identified” or “exactly identified.” Therefore, the choice of weight matrix has no impact on the solution to (2), and the criterion function  $Q(\beta)$  achieves its minimum value at zero.

The OLS estimator is a one-step GMM estimator, but we did not bother to specify the `onestep` option because the model is just identified. Doing a second step of GMM estimation affects neither the point estimates nor the standard errors, so to keep the syntax as simple as possible, we did not include the `onestep` option. The first step of estimation resulted in  $Q(\beta) = 0$  as expected, and the second step of estimation did not change the minimized value of  $Q(\beta)$ . ( $3 \times 10^{-27}$  and  $7 \times 10^{-32}$  are both zero for all practical purposes.)

When you do not specify either the `wmatrix()` or the `vce()` option, `gmm` reports heteroskedasticity-robust standard errors. The parameter estimates reported here match those that we would obtain from the command

```
. regress mpg weight length, vce(robust)
```

The standard errors reported by that `regress` command would be larger than those reported by `gmm` by a factor of  $\sqrt{74/71}$  because `regress` makes a small-sample adjustment to the estimated variance matrix while `gmm` does not. Likewise, had we specified the `vce(unadjusted)` option with our `gmm` command, then our standard errors would differ by a factor of  $\sqrt{74/71}$  from those reported by `regress` without the `vce(robust)` option.

Using the notation for linear combinations of parameters, we could have typed

```
. gmm (mpg - {xb: weight length} - {b0}), instruments(weight length)
```

and obtained identical results. Instead of having parameters `b1` and `b2`, with this syntax we would have parameters `xb_weight` and `xb_length`.

◀

## ► Example 2: Instrumental-variables regression

In [Introduction](#), we mentioned that 2SLS can be viewed as a GMM estimator. In [example 1](#) of [\[R\] ivregress](#), we fit by 2SLS a model of rental rates (`rent`) as a function of the value of owner-occupied housing (`hshngval`) and the percentage of the population living in urban areas (`pcturban`):

$$\text{rent} = \beta_0 + \beta_1 \text{hsngval} + \beta_2 \text{pcturban} + u$$

by 2SLS. We argued that random shocks that affect rental rates likely also affect housing values, so we treated `hsngval` as an endogenous variable. As additional instruments, we used family income, `faminc`, and three regional dummies (`reg2-reg4`).

To replicate the results of `ivregress 2sls` by using `gmm`, we type

```
. use http://www.stata-press.com/data/r11/hsng2
(1980 Census housing data)

. gmm (rent - {xb:hsngval pcturban} - {b0}),
> instruments(pcturban faminc reg2-reg4) vce(unadjusted) onestep

Step 1
Iteration 0: GMM criterion Q(b) = 56115.03
Iteration 1: GMM criterion Q(b) = 110.91583
Iteration 2: GMM criterion Q(b) = 110.91583

GMM estimation
Number of parameters = 3
Number of moments = 6
Initial weight matrix: Unadjusted Number of obs = 50
```

|              | Coef.    | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|--------------|----------|-----------|------|-------|----------------------|----------|
| /xb_hsngval  | .0022398 | .0003284  | 6.82 | 0.000 | .0015961             | .0028836 |
| /xb_pcturban | .081516  | .2987652  | 0.27 | 0.785 | -.5040531            | .667085  |
| /b0          | 120.7065 | 15.22839  | 7.93 | 0.000 | 90.85942             | 150.5536 |

```
Instruments for equation 1: pcturban faminc reg2 reg3 reg4 _cons
```

We specified `vce(unadjusted)` so that we would obtain an unadjusted VCE matrix and our standard errors would match those reported in [R] `ivregress`.

Pay attention to how we specified the `instruments()` option. In [Introduction](#), we mentioned that the moment conditions for the 2SLS estimator are  $E(zu) = \mathbf{0}$ , and we mentioned that if some elements of  $\mathbf{x}$  (the regressors) are not endogenous, then they should also appear in  $\mathbf{z}$ . In this model, we assume the regressor `pcturban` is exogenous, so we included it in the list of instrumental variables. Commands like `ivregress`, `ivprobit`, and `ivtobit` accept standard *varlists*, so they can deduce the exogenous regressors in the model. Because `gmm` accepts arbitrary functions in the form of substitutable expressions, it has no way of discerning the exogenous variables of the model on its own.

Also notice that we specified the `onestep` option. The 2SLS estimator is a one-step GMM estimator that is based on a weight matrix that assumes the error terms are i.i.d. Unlike the previous example, here we had more instruments than parameters, so the minimized value of  $Q(\beta)$  is nonzero. We discuss the weight matrix and its relationship to two-step estimation next.



## The weight matrix and two-step estimation

Recall our definition of the GMM estimator given in (2). The estimator,  $\widehat{\beta}$ , depends on the choice of the weight matrix,  $\mathbf{W}$ . Under relatively mild assumptions, our estimator,  $\widehat{\beta}$ , is consistent regardless of the choice of  $\mathbf{W}$ , so how are we to decide what  $\mathbf{W}$  to use? The most common solution is to use the two-step estimator, which we now describe.

A key result in Hansen's (1982) seminal paper is that if we denote by  $\mathbf{S}$  the covariance matrix of the moment conditions, then the optimal (in a way we make precise later) GMM estimator is the one that uses a weight matrix equal to the inverse of the moment covariance matrix. That is, if we let  $\mathbf{S} = \text{Cov}(\mathbf{z}u)$ , then we want to use  $\mathbf{W} = \mathbf{S}^{-1}$ . But how do we obtain  $\mathbf{S}$  in the first place?

If we assume that the errors are i.i.d., then

$$\text{Cov}(\mathbf{z}u) = E(u^2 \mathbf{z}\mathbf{z}') = \sigma^2 E(\mathbf{z}\mathbf{z}')$$

where  $\sigma^2$  is the variance of  $u$ . Because  $\sigma^2$  is a positive scalar, we can ignore it when solving (2). Thus we compute

$$\widehat{\mathbf{W}}_1 = \left( \frac{1}{N} \sum_i \mathbf{z}_i \mathbf{z}_i' \right)^{-1} \quad (4)$$

which does not depend on any unknown model parameters. (Notice that  $\widehat{\mathbf{W}}_1$  is the same weight matrix used in 2SLS.) Given  $\widehat{\mathbf{W}}_1$ , we can solve (2) to obtain an initial estimate, say,  $\hat{\beta}_1$ .

Our estimate,  $\hat{\beta}_1$ , is consistent, so by Slutsky's theorem, the sample residuals  $\hat{u}$  computed at this value of  $\beta$  will also be consistent. Using virtually the same arguments used to justify the Huber/Eicker/White heteroskedasticity-robust VCE, if we assume that the residuals are independent though not identically distributed, we can estimate  $\mathbf{S}$  as

$$\hat{\mathbf{S}} = \frac{1}{N} \sum_i \hat{u}_i^2 \mathbf{z}_i \mathbf{z}_i'$$

Then, in the second step, we re-solve (2), using  $\widehat{\mathbf{W}}_2 = \hat{\mathbf{S}}^{-1}$ , yielding the two-step GMM estimate  $\hat{\beta}_2$ . If the residuals exhibit clustering, you can specify `wmatrix(cluster varname)` so that `gmm` computes a weight matrix that does not assume the  $u_i$ 's are independent within clusters identified by `varname`. You can specify `wmatrix(hac ...)` to obtain weight matrices that are suitable for when the  $u_i$ 's exhibit autocorrelation as well as heteroskedasticity.

We could take the point estimates from the second round of estimation and use them to compute yet another weight matrix,  $\widehat{\mathbf{W}}_3$ , say, re-solve (2) yet again, and so on, stopping when the parameters or weight matrix do not change much from one iteration to the next. This procedure is known as the iterative GMM estimator and is obtained with the `igmm` option. Asymptotically, the two-step and iterative GMM estimators have the same distribution. However, Hall (2005, 90) suggests that the iterative estimator may have better finite-sample properties.

Instead of computing  $\widehat{\mathbf{W}}_1$  as in (4), we could simply choose  $\widehat{\mathbf{W}}_1 = \mathbf{I}$ , the identity matrix. The initial estimate,  $\hat{\beta}_1$ , would still be consistent. You can request this behavior by specifying the `winitial(identity)` option. However, if you specify all your moment equations of the form  $E(\mathbf{z}u) = \mathbf{0}$ , we recommend using the default `winitial(unadjusted)` instead; the rescaling of the moment conditions implied by using a homoskedastic initial weight matrix makes the numerical routines used to solve (2) more stable.

If you fit a model with more than one of the moment equations of the form  $E\{h(\mathbf{z}; \beta)\} = \mathbf{0}$ , then you must use `winitial(identity)` or `winitial(unadjusted, independent)`. With moment equations of that form, you do not specify a list of instruments, and `gmm` cannot evaluate (4)—the matrix expression in parentheses would necessarily be singular, so it cannot be inverted.

➤ Example 3: Two-step linear GMM estimator

From the previous discussion and the comments in [Introduction](#), we see that the linear 2SLS estimator is a one-step GMM estimator where we use the weight matrix defined in (4) that assumes the errors are i.i.d. If we use the 2SLS estimate of  $\beta$  to obtain the sample residuals, compute a new weight matrix based on those residuals, and then do a second step of GMM estimation, we obtain the linear two-step GMM estimator as implemented by `ivregress gmm`.

In [example 3](#) of [\[R\] ivregress](#), we fit the model of rental rates as discussed in [example 2](#) above. We now allow the residuals to be heteroskedastic, though we will maintain our assumption that they are independent. We type

```
. gmm (rent - {xb:hsngval pcturban} - {b0}), inst(pcturban faminc reg2-reg4)

Step 1
Iteration 0: GMM criterion Q(b) = 56115.03
Iteration 1: GMM criterion Q(b) = 110.91583
Iteration 2: GMM criterion Q(b) = 110.91583

Step 2
Iteration 0: GMM criterion Q(b) = .2406087
Iteration 1: GMM criterion Q(b) = .13672801
Iteration 2: GMM criterion Q(b) = .13672801

GMM estimation
Number of parameters = 3
Number of moments = 6
Initial weight matrix: Unadjusted
GMM weight matrix: Robust

 Number of obs = 50

-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

|              | Coef.    | Robust<br>Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|--------------|----------|---------------------|-------|-------|----------------------|----------|
| /xb_hsgval   | .0014643 | .0004473            | 3.27  | 0.001 | .0005877             | .002341  |
| /xb_pcturban | .7615482 | .2895105            | 2.63  | 0.009 | .1941181             | 1.328978 |
| /b0          | 112.1227 | 10.80234            | 10.38 | 0.000 | 90.95052             | 133.2949 |

```
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
Instruments for equation 1: pcturban faminc reg2 reg3 reg4 _cons
```

By default, `gmm` computes a heteroskedasticity-robust weight matrix before the second step of estimation, though we could have specified `wmatrix(robust)` if we wanted to be explicit. Because we did not specify the `vce()` option, `gmm` used a heteroskedasticity-robust one. Our results match those in [example 3](#) of [\[R\] ivregress](#). Moreover, the only difference between this example and the previous example of 2SLS is that here we did not use the `onestep` option.

Obtaining standard errors

This section is a bit more theoretical and can be skipped on first reading. However, the information is sufficiently important that you should return to this section at some point.

So far in our discussion, we have focused on point estimation without much mention of how we obtain the standard errors of the estimates. We also mentioned that if we choose  $\mathbf{W}$  to be the inverse of the covariance matrix of the moment conditions, then we obtain the “optimal” GMM estimator. We elaborate those points now.

Using mostly standard statistical arguments, we can show that for the GMM estimator defined in (2), the variance of  $\hat{\beta}$  is given by



$$\text{Var}(\hat{\beta}) = \frac{1}{N} \left\{ \overline{\mathbf{G}}(\hat{\beta})' \mathbf{W} \overline{\mathbf{G}}(\hat{\beta}) \right\}^{-1} \overline{\mathbf{G}}(\hat{\beta})' \mathbf{W} \mathbf{S} \mathbf{W} \overline{\mathbf{G}}(\hat{\beta}) \left\{ \overline{\mathbf{G}}(\hat{\beta})' \mathbf{W} \overline{\mathbf{G}}(\hat{\beta}) \right\}^{-1} \quad (5)$$

where

$$\overline{\mathbf{G}}(\hat{\beta}) = \frac{1}{N} \sum_i \mathbf{z}_i \left. \frac{\partial u_i}{\partial \beta} \right|_{\beta=\hat{\beta}} \quad \text{or} \quad \overline{\mathbf{G}}(\hat{\beta}) = \frac{1}{N} \sum_i \left. \frac{\partial \mathbf{h}_i}{\partial \beta} \right|_{\beta=\hat{\beta}}$$

as the case may be and  $\mathbf{S} = E(\mathbf{z}u u' \mathbf{z}')$ .

Assuming the `vce(unadjusted)` option is not specified, `gmm` reports standard errors based on the robust variance matrix defined in (5). For the two-step estimator,  $\mathbf{W}$  is the weight matrix requested using the `wmatrix()` option, and it is calculated based on the residuals obtained after the first estimation step. The second-step point estimates and residuals are obtained, and  $\mathbf{S}$  is calculated based on the specification of the `vce()` option. For the iterated estimator,  $\mathbf{W}$  is calculated based on the second-to-last round of estimation, while  $\mathbf{S}$  is based on the residuals obtained after the last round of estimation. Computation of the covariance matrix for the one-step estimator is, perhaps surprisingly, more involved; we discuss the covariance matrix with the one-step estimator in the technical note at the end of this section.

If we choose the weight matrix to be the inverse of the covariance matrix of the moment conditions so that  $\mathbf{W} = \mathbf{S}^{-1}$ , then (5) simplifies substantially:

$$\text{Var}(\hat{\beta}) = \frac{1}{N} \left\{ \overline{\mathbf{G}}(\hat{\beta})' \mathbf{W} \overline{\mathbf{G}}(\hat{\beta}) \right\}^{-1} \quad (6)$$

The GMM estimator constructed using this choice of weight matrix along with the covariance matrix in (6) is known as the “optimal” GMM estimator. One can show that if in fact  $\mathbf{W} = \mathbf{S}^{-1}$ , then the variance in (6) is smaller than the variance in (5) of any other GMM estimator based on the same moment conditions but with a different choice of weight matrix. Thus the optimal GMM estimator is also known as the efficient GMM estimator, because it has the smallest variance of any estimator based on the given moment conditions.

To obtain standard errors from `gmm` based on the optimal GMM estimator, you specify the `vce(unadjusted)` option. We call that VCE unadjusted because we do not recompute the residuals after estimation to obtain the matrix  $\mathbf{S}$  required in (5) or allow for the fact that those residuals may not be i.i.d. Some statistical packages by default report standard errors based on (6) and offer standard errors based on (5) only as an option or not at all. While the optimal GMM estimator is theoretically appealing, [Cameron and Trivedi \(2005, 177\)](#) suggest that in finite samples it need not perform better than the GMM estimator that uses (5) to obtain standard errors.

## □ Technical note

Computing the covariance matrix of the parameters after using the one-step estimator is actually a bit more complex than after using the two-step or iterative estimators. We can illustrate most of the intricacies by using linear regression with moment conditions of the form  $E\{\mathbf{x}(y - \mathbf{x}'\beta)\} = \mathbf{0}$ .

If you specify `winitial(unadjusted)` and `vce(unadjusted)`, then the initial weight matrix will be computed as

$$\widehat{\mathbf{W}}_1 = \left( \frac{1}{N} \sum_i \mathbf{x}_i \mathbf{x}_i' \right)^{-1} \quad (7)$$

Moreover, for linear regression, we can show that

$$\overline{\mathbf{G}}(\hat{\beta}) = \frac{1}{N} \sum_i \mathbf{x}_i \mathbf{x}_i'$$

so that (6) becomes

$$\begin{aligned}\text{Var}(\hat{\beta}) &= \frac{1}{N} \left\{ \left( \frac{1}{N} \sum_i \mathbf{x}_i \mathbf{x}_i' \right) \left( \frac{1}{N} \sum_i \mathbf{x}_i \mathbf{x}_i' \right)^{-1} \left( \frac{1}{N} \sum_i \mathbf{x}_i \mathbf{x}_i' \right) \right\}^{-1} \\ &= \left( \sum_i \mathbf{x}_i \mathbf{x}_i' \right)^{-1} \\ &= (\mathbf{X}'\mathbf{X})^{-1}\end{aligned}\tag{8}$$

However, we know that the nonrobust covariance matrix for the OLS estimator is actually  $\hat{\sigma}^2(\mathbf{X}'\mathbf{X})^{-1}$ . What is missing from (8) is the scalar  $\hat{\sigma}^2$ , the estimated variance of the residuals. When you use the one-step estimator and specify `winitial(unadjusted)`, the weight matrix (7) does not include the  $\hat{\sigma}^2$  term because `gmm` does not have a consistent estimate of  $\beta$  from which it can then estimate  $\sigma^2$ . The point estimates are still correct, because multiplying the weight matrix by a scalar factor does not affect the solution to the minimization problem.

To circumvent this issue, if you specify `winitial(unadjusted)` and `vce(unadjusted)`, `gmm` uses the estimated  $\hat{\beta}$  (which is consistent) to obtain a new unadjusted weight matrix that does include the term  $\hat{\sigma}^2$  so that evaluating (6) will yield correct standard errors.

If you use the two-step or iterated GMM estimators, this extra effort is not needed to obtain standard errors because the first-step (and subsequent steps') estimate of  $\beta$  is consistent and can be used to estimate  $\sigma^2$  or some other weight matrix based on the `wmatrix()` option. Straightforward algebra shows that this extra effort is also not needed if you request any type of adjusted (robust) covariance matrix with the one-step estimator.

A similar issue arises when you specify `winitial(identity)` and `vce(unadjusted)` with the one-step estimator. Again the solution is to compute an unadjusted weight matrix after obtaining  $\hat{\beta}$  so that (6) provides the correct standard errors.

We have illustrated the problem and solution using a single-equation linear model. However, the problem arises whenever you use the one-step estimator with an unadjusted VCE, regardless of the number of equations; and `gmm` handles all the details automatically. Computation of Hansen's  $J$  statistic presents an identical issue, and `gmm` takes care of that as well.

If you supply your own initial weight matrix by using `winitial(matname)`, then the standard errors (as well as the  $J$  statistic reported by `estat overid`) are based on that weight matrix. You should verify that the weight matrix you provide will yield appropriate statistics.

□

## Exponential (Poisson) regression models

Exponential regression models are frequently encountered in applied work. For example, they can be used as alternatives to linear regression models on log-transformed dependent variables, obviating the need for post-hoc transformations to obtain predicted values in the original metric of the dependent variable. When the dependent variable represents a discrete count variable, they are also known as Poisson regression models; see [Cameron and Trivedi \(1998\)](#).

For now, we consider models of the form

$$y = \exp(\mathbf{x}'\beta) + u\tag{9}$$

where  $u$  is a zero-mean additive error term so that  $E(y) = \exp(\mathbf{x}'\beta)$ . Because the error term is additive, if  $\mathbf{x}$  represents strictly exogenous regressors, then we have the population moment condition

$$E[\mathbf{x}\{y - \exp(\mathbf{x}'\beta)\}] = \mathbf{0} \quad (10)$$

Moreover, because the number of parameters in the model is equal to the number of instruments, there is no point to using the two-step GMM estimator.

## ► Example 4: Exponential regression

Cameron and Trivedi (2009, 317) fit a model of the number of doctor visits based on whether the patient has private insurance, whether the patient has a chronic disease, gender, and income. Here we fit that model by using `gmm`. To allow for potential excess dispersion, we will obtain a robust VCE matrix, which is the default for `gmm` anyway. We type

```
. use http://www.stata-press.com/data/r11/docvisits
. gmm (docvis - exp({xb:private chronic female income}+{b0})),
> instruments(private chronic female income) onestep
Step 1
Iteration 0: GMM criterion Q(b) = 16.853973
Iteration 1: GMM criterion Q(b) = 2.2706472
Iteration 2: GMM criterion Q(b) = .19088097
Iteration 3: GMM criterion Q(b) = .00041101
Iteration 4: GMM criterion Q(b) = 3.939e-09
Iteration 5: GMM criterion Q(b) = 6.572e-19
GMM estimation
Number of parameters = 5
Number of moments = 5
Initial weight matrix: Unadjusted Number of obs = 4412
```

|             | Coef.     | Robust<br>Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-------------|-----------|---------------------|-------|-------|----------------------|-----------|
| /xb_private | .7986654  | .1089891            | 7.33  | 0.000 | .5850507             | 1.01228   |
| /xb_chronic | 1.091865  | .0559888            | 19.50 | 0.000 | .9821291             | 1.201601  |
| /xb_female  | .4925481  | .0585298            | 8.42  | 0.000 | .3778317             | .6072644  |
| /xb_income  | .003557   | .0010824            | 3.29  | 0.001 | .0014356             | .0056784  |
| /b0         | -.2297263 | .1108607            | -2.07 | 0.038 | -.4470093            | -.0124434 |

Instruments for equation 1: private chronic female income \_cons

Our point estimates agree with those reported by Cameron and Trivedi to at least six significant digits; the small discrepancies are attributable to different optimization techniques and convergence criteria being used by `gmm` and `poisson`. The standard errors differ by a factor of  $\sqrt{4412/4411}$  because `gmm` uses  $N$  in the denominator of the formula for the robust covariance matrix, while the robust covariance matrix estimator used by `poisson` uses  $N - 1$ .

◄

## □ Technical note

That the GMM and maximum likelihood estimators of the exponential regression model coincide is not a general property of these two classes of estimators. The maximum likelihood estimator solves the score equations

$$\frac{1}{N} \sum_{i=1}^N \frac{\partial \ln \ell_i}{\partial \beta} = \mathbf{0}$$

where  $\ell_i$  is the likelihood for the  $i$ th observation. These score equations can be viewed as the sample analogues of the population moment conditions

$$E \left\{ \frac{\partial \ln \ell_i}{\partial \beta} \right\} = \mathbf{0}$$

establishing that maximum likelihood estimators represent a subset of the class of GMM estimators.

For the Poisson model,

$$\ln \ell_i = -\exp(\mathbf{x}'_i \beta) + y_i \mathbf{x}'_i \beta - \ln y_i!$$

so the score equations are

$$\frac{1}{N} \sum_{i=1}^N \mathbf{x}_i \{y_i - \exp(\mathbf{x}'_i \beta)\} = \mathbf{0}$$

which are just the sample moment conditions implied by (10) that we used in the previous example. That is why our results using `gmm` match Cameron and Trivedi's results using `poisson`.

On the other hand, an intuitive set of moment conditions to consider for GMM estimation of a probit model is

$$E\{\mathbf{x}(y - \Phi(\mathbf{x}'\beta))\} = \mathbf{0}$$

where  $\Phi()$  is the standard normal cumulative distribution function. Differentiating the likelihood function for the maximum likelihood probit estimator, we can show that the corresponding score equations are

$$\frac{1}{N} \sum_{i=1}^N \left[ \mathbf{x}_i \left\{ y_i \frac{\phi(\mathbf{x}'_i \beta)}{\Phi(\mathbf{x}'_i \beta)} - (1 - y_i) \frac{\phi(\mathbf{x}'_i \beta)}{1 - \Phi(\mathbf{x}'_i \beta)} \right\} \right] = \mathbf{0}$$

where  $\phi()$  is the standard normal density function. These two moment conditions are not equivalent, so the maximum likelihood and GMM probit estimators are distinct. □

## ► Example 5: Comparison of GMM and maximum likelihood

Using the automobile dataset, here we fit a probit model of `foreign` on `gear_ratio`, `length`, and `headroom` using first the score equations and then the intuitive set of GMM equations. We type

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. global xb "{b1}*gear_ratio + {b2}*length + {b3}*headroom + {b0}"
. global phi "normalden($xb)"
. global Phi "normal($xb)"
. gmm (foreign*$phi/$Phi - (1-foreign)*$phi/(1-$Phi)),
> instruments(gear_ratio length headroom) onestep
(output omitted)
. estimates store ml

. gmm (foreign - $Phi), instruments(gear_ratio length headroom) onestep
(output omitted)
. estimates store gmm
```

```
. estimates table ml gmm, b se
```

| Variable | ml                            | gmm                     |
|----------|-------------------------------|-------------------------|
| b1       |                               |                         |
|          | _cons 2.9586277<br>.64042341  | 2.8489213<br>.63570246  |
| b2       |                               |                         |
|          | _cons -.02148933<br>.01382043 | -.02056033<br>.01396954 |
| b3       |                               |                         |
|          | _cons .01136927<br>.27278528  | .02240761<br>.2849891   |
| b0       |                               |                         |
|          | _cons -6.0222289<br>3.5594588 | -5.8595615<br>3.5188028 |

legend: b/se

The coefficients on `gear_ratio` and `length` are close for the two estimators. The GMM estimate of the coefficient on `headroom` is twice that of the maximum likelihood estimate, though the relatively large standard errors imply that this difference is not significant. You can verify that the coefficients in the column marked “ml” match those you would obtain using `probit`. We have not discussed the differences among standard errors based on the various GMM and maximum-likelihood covariance matrix estimators to avoid tedious algebra, though you can verify that the robust covariance matrix after one-step GMM estimation differs by only a finite-sample adjustment factor of  $(N/N - 1)$  from the robust covariance matrix reported by `probit`. Both the maximum likelihood and GMM `probit` estimators require the normality assumption, and the maximum likelihood estimator is efficient if that normality assumption is correct; therefore, in this particular example, there is no reason to prefer the GMM estimator.

◀

We can modify (10) easily to allow for endogenous regressors. Suppose that  $x_j$  is endogenous in the sense that  $E(u|x_j) \neq 0$ . Then (10) is no longer a valid moment condition. However, suppose we have some variables other than  $\mathbf{x}$  such that  $E(u|\mathbf{z}) = 0$ . We can instead use the moment conditions

$$E(\mathbf{z}u) = E[\mathbf{z}\{y - \exp(\mathbf{x}'\beta)\}] = \mathbf{0} \quad (11)$$

As usual, if some elements of  $\mathbf{x}$  are exogenous, then they should appear in  $\mathbf{z}$  as well.

## ► Example 6: Exponential regression with endogenous regressors

Returning to the model discussed in [example 4](#), here we treat `income` as endogenous; unobservable factors that determine a person’s income may also affect the number of times a person visits a doctor. We use a person’s age and race as instruments. These are valid instruments if we believe that age and race influence a person’s income but do not have a direct impact on the number of doctor visits. (Whether this belief is justified is another matter; we test that belief in [\[R\] gmm postestimation.](#)) Because we have more instruments (seven) than parameters (five), we have an overidentified model. Therefore, the choice of weight matrix does matter. We will utilize the default two-step GMM estimator. In the first step, we will use a weight matrix that assumes the errors are i.i.d. In the second step, we will use a weight matrix that assumes heteroskedasticity. When you specify `twostep`, these are the defaults for the first- and second-step weight matrices, so we do not have to use the `winitial()` or `wmatrix()` options. We will again obtain a robust VCE, which is also the default. We type

```
. use http://www.stata-press.com/data/r11/docvisits
. gmm (docvis - exp({xb:private chronic female income}+{b0})),
> instruments(private chronic female age black hispanic)

Step 1
Iteration 0: GMM criterion Q(b) = 16.910173
Iteration 1: GMM criterion Q(b) = .82276104
Iteration 2: GMM criterion Q(b) = .21832032
Iteration 3: GMM criterion Q(b) = .12685935
Iteration 4: GMM criterion Q(b) = .12672369
Iteration 5: GMM criterion Q(b) = .12672365

Step 2
Iteration 0: GMM criterion Q(b) = .00234641
Iteration 1: GMM criterion Q(b) = .00215957
Iteration 2: GMM criterion Q(b) = .00215911
Iteration 3: GMM criterion Q(b) = .00215911

GMM estimation
Number of parameters = 5
Number of moments = 7
Initial weight matrix: Unadjusted
GMM weight matrix: Robust

Number of obs = 4412
```

|             | Robust    |           |       |       | [95% Conf. Interval] |          |
|-------------|-----------|-----------|-------|-------|----------------------|----------|
|             | Coef.     | Std. Err. | z     | P> z  |                      |          |
| /xb_private | .535335   | .1599039  | 3.35  | 0.001 | .2219291             | .8487409 |
| /xb_chronic | 1.090126  | .0617659  | 17.65 | 0.000 | .9690668             | 1.211185 |
| /xb_female  | .6636579  | .0959884  | 6.91  | 0.000 | .4755241             | .8517918 |
| /xb_income  | .0142855  | .0027162  | 5.26  | 0.000 | .0089618             | .0196092 |
| /b0         | -.5983477 | .138433   | -4.32 | 0.000 | -.8696713            | -.327024 |

Instruments for equation 1: private chronic female age black hispanic\_cons

Once we control for the endogeneity of income, we find that its coefficient has quadrupled in size. Additionally, access to private insurance has less of an impact on the number of doctor visits and gender has more of an impact.

□ Technical note

Although perhaps at first tempting, unlike the Poisson model, you cannot simply replace  $\mathbf{x}$  in the moment conditions for the probit (or logit) model with a vector of instruments,  $\mathbf{z}$ , if you have endogenous regressors. See [Wilde \(2008\)](#).

[Mullahy \(1997\)](#) considers a slightly more complicated version of the exponential regression model that incorporates nonadditive unobserved heterogeneity. His model can be written as

$$y_i = \exp(\mathbf{x}_i'\beta)\eta_i + \epsilon_i$$

where  $\eta_i > 0$  is an unobserved heterogeneity term that may be correlated with  $\mathbf{x}_i$ . One result from his paper is that instead of using the additive moment condition (9), we can use the multiplicative moment condition

$$E\left\{\mathbf{z}\frac{y - \exp(\mathbf{x}'\beta)}{\exp(\mathbf{x}'\beta)}\right\} = E[\mathbf{z}\{y\exp(-\mathbf{x}'\beta) - 1\}] = \mathbf{0} \tag{12}$$

Windmeijer and Santos Silva (1997) discuss the use of additive versus multiplicative moment conditions with endogenous regressors and note that a set of instruments that satisfies the additive moment conditions will not also satisfy the multiplicative moment conditions. They remark that which to use is an empirical issue that can at least partially be settled by using the test of overidentifying restrictions that is implemented by `estat overid` after `gmm` to ascertain whether the instruments for a given model are valid. See [R] [gmm postestimation](#) for information on the test of overidentifying restrictions.

## Specifying derivatives

By default, `gmm` calculates derivatives numerically, and the method used produces accurate results for the vast majority of applications. However, if you refit the same model repeatedly or else have the derivatives available, then `gmm` will run more quickly if you supply it with analytic derivatives.

When you use the interactive version of `gmm`, you specify derivatives using substitutable expressions in much the same way you specify the moment equations. There are three rules you must follow:

1. As with the substitutable expressions that define residual equations, you bind parameters of the model in braces: `{b0}`, `{param}`, etc.
2. You must specify a derivative for each parameter that appears in each moment equation. If a parameter does not appear in a moment equation, then you do not specify a derivative for that parameter in that moment equation.
3. If you declare a linear combination in an equation, then you specify a derivative with respect to that linear combination. `gmm` applies the chain rule to obtain the derivatives with respect to the individual parameters encompassed by that linear combination.

We illustrate with several examples.

### ► Example 7: Derivatives for a single-equation model

Consider a simple exponential regression model with one exogenous regressor and a constant term. We have

$$u = y - \exp(\beta_0 + \beta_1 x)$$

Now

$$\frac{\partial u}{\partial \beta_0} = -\exp(\beta_0 + \beta_1 x) \quad \text{and} \quad \frac{\partial u}{\partial \beta_1} = -x \exp(\beta_0 + \beta_1 x)$$

In Stata, we type

```
. gmm (docvis - exp({b0} + {b1}*income)), instruments(income)
> deriv(/b0 = -1*exp({b0} + {b1}*income))
> deriv(/b1 = -1*income*exp({b0}+{b1}*income)) onestep
```

Step 1

```
Iteration 0: GMM criterion Q(b) = 9.1548611
Iteration 1: GMM criterion Q(b) = 3.5146131
Iteration 2: GMM criterion Q(b) = .01344695
Iteration 3: GMM criterion Q(b) = 3.690e-06
Iteration 4: GMM criterion Q(b) = 4.606e-13
Iteration 5: GMM criterion Q(b) = 1.502e-26
```

```
GMM estimation
Number of parameters = 2
Number of moments = 2
Initial weight matrix: Unadjusted
Number of obs = 4412
```

|     | Coef.    | Robust Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-----|----------|------------------|-------|-------|----------------------|----------|
| /b0 | 1.204888 | .0462355         | 26.06 | 0.000 | 1.114268             | 1.295507 |
| /b1 | .0046702 | .0009715         | 4.81  | 0.000 | .0027662             | .0065743 |

```
Instruments for equation 1: income_cons
```

Notice how we specified the `derivative()` option for each parameter. We simply specified a slash, the name of the parameter, an equal sign, then a substitutable expression that represents the derivative. Because our model has only one residual equation, we do not need to specify equation numbers in the `derivative()` options.



When you specify a linear combination of variables, your derivative should be with respect to the entire linear combination. For example, say we have the residual equation

$$u = y - \exp(\mathbf{x}'\beta + \beta_0)$$

for which we would type

```
. gmm (y - exp({xb: x1 x2 x3} + {b0}) ...
```

Then in addition to the derivative  $\partial u/\partial \beta_0$ , we are to compute and specify

$$\frac{\partial u}{\partial (\mathbf{x}'\beta)} = -\exp(\mathbf{x}'\beta + \beta_0)$$

Using the chain rule,  $\partial u/\partial \beta_j = \partial u/\partial (\mathbf{x}'\beta) \times \partial (\mathbf{x}'\beta)/\partial \beta_j = -x_j \exp(\mathbf{x}'\beta + \beta_0)$ . Stata does this last calculation automatically. It knows the variables in the linear combination, so all it needs is the derivative of the residual function with respect to the linear combination. This allows you to change the variables in your linear combination without having to change the derivatives.

➤ Example 8: Derivatives with a linear combination

We refit the model described in the example illustrating exponential regression with endogenous regressors, now providing analytic derivatives. We type

```
. gmm (docvis - exp({xb:private chronic female income}+{b0})),
> instruments(private chronic female age black hispanic)
> derivative(/xb = -1*exp({xb:} + {b0}))
> derivative(/b0 = -1*exp({xb:} + {b0}))
```

```
Step 1
Iteration 0: GMM criterion Q(b) = 16.910173
Iteration 1: GMM criterion Q(b) = .82270871
Iteration 2: GMM criterion Q(b) = .21831995
Iteration 3: GMM criterion Q(b) = .12685934
Iteration 4: GMM criterion Q(b) = .12672369
Iteration 5: GMM criterion Q(b) = .12672365
```

```
Step 2
Iteration 0: GMM criterion Q(b) = .00234641
Iteration 1: GMM criterion Q(b) = .00215957
Iteration 2: GMM criterion Q(b) = .00215911
Iteration 3: GMM criterion Q(b) = .00215911
```



GMM estimation

Number of parameters = 5

Number of moments = 7

Initial weight matrix: Unadjusted

Number of obs = 4412

GMM weight matrix: Robust

|             | Coef.     | Robust<br>Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-------------|-----------|---------------------|-------|-------|----------------------|----------|
| /xb_private | .535335   | .159904             | 3.35  | 0.001 | .221929              | .848741  |
| /xb_chronic | 1.090126  | .0617659            | 17.65 | 0.000 | .9690668             | 1.211185 |
| /xb_female  | .6636579  | .0959885            | 6.91  | 0.000 | .475524              | .8517918 |
| /xb_income  | .0142855  | .0027162            | 5.26  | 0.000 | .0089618             | .0196092 |
| /b0         | -.5983477 | .138433             | -4.32 | 0.000 | -.8696714            | -.327024 |

Instruments for equation 1: private chronic female age black hispanic\_cons

In the first `derivative()` option, we specified the name of the linear combination, `xb`, instead of an individual parameter's name. We already declared the variables of our linear combination in the substitutable expression for the residual equation, so in our substitutable expressions for the derivatives, we can use the shorthand notation `{xb:}` to refer to it.

Our point estimates are identical to those we obtained earlier. The standard errors and confidence intervals differ by only trivial amounts.

◀

## Exponential regression models with panel data

In addition to supporting cross-sectional and time-series data, `gmm` also works with panel-data models. Here we illustrate `gmm`'s panel-data capabilities by expanding our discussion of exponential regression models to allow for panel data. This also provides us the opportunity to demonstrate the moment-evaluator program version of `gmm`. Our discussion is based on [Blundell, Griffith, and Windmeijer \(2002\)](#).

First, we expand (9) for panel data. With individual heterogeneity term  $\eta_i$ , we have

$$E(y_{it}|\mathbf{x}_{it}, \eta_i) = \exp(\mathbf{x}'_{it}\beta + \eta_i) = \mu_{it}\nu_i$$

where  $\mu_{it} = \exp(\mathbf{x}'_{it}\beta)$  and  $\nu_i = \exp(\eta_i)$ . Note that there is no constant term in this model because its effect cannot be disentangled from  $\nu_i$ . With an additive idiosyncratic error term, we have the regression model

$$y_{it} = \mu_{it}\nu_i + \epsilon_{it}$$

We do not impose the assumption  $E(\mathbf{x}_{it}\eta_i) = \mathbf{0}$ , so  $\eta_i$  can be considered a fixed effect in the sense that it may be correlated with the regressors.

As discussed by [Blundell, Griffith, and Windmeijer \(2002\)](#), if  $\mathbf{x}_{it}$  is strictly exogenous, meaning  $E(\mathbf{x}_{it}\epsilon_{is}) = \mathbf{0}$  for all  $t$  and  $s$ , then we can estimate the parameters of the model by using the sample moment conditions

$$\sum_i \sum_t \mathbf{x}_{it} \left( y_{it} - \mu_{it} \frac{\bar{y}_i}{\bar{\mu}_i} \right) = \mathbf{0} \quad (13)$$

where  $\bar{y}_i$  and  $\bar{\mu}_i$  are the means of  $y_{it}$  and  $\mu_{it}$  for panel  $i$ , respectively. Because  $\bar{\mu}_i$  depends on the parameters of the model, it must be recomputed each time `gmm` needs to evaluate the residual equation. Therefore, we cannot use the substitutable expression version of `gmm`. Instead, we must use the moment-evaluator program version.

The moment-evaluator program version of `gmm` functions much like the function-evaluator program versions of `nl` and `nlsur`. The program you write is passed one or more variables to be filled in with the residuals evaluated at the parameter values specified in an option passed to your program. For the fixed-effects Poisson model with strictly exogenous regressors, our first crack at a function-evaluator program is

```

program gmm_poi
 version 11
 syntax varlist if, at(name)
 quietly {
 tempvar mu mubar ybar
 gen double 'mu' = exp(x1*'at'[1,1] + x2*'at'[1,2] + x3*'at'[1,3]) 'if' ///
 egen double 'mubar' = mean('mu') 'if', by(id)
 egen double 'ybar' = mean(y) 'if', by(id)
 replace 'varlist' = y - 'mu'*'ybar'/'mubar' 'if'
 }
end

```

You can save your program in an ado-file named *name.ado*, where *name* is the name you use for your program; here we would save the program in the ado-file `gmm_poi.ado`. Alternatively, if you are working from within a do-file, you can simply define the program before calling `gmm`. The `syntax` statement declares that we are expecting to receive a *varlist*, containing the names of variables whose values we are to replace with the values of the residual equations, and an *if* expression that will mark the estimation sample; because our model has one residual equation, *varlist* will consist of one variable. `at()` is a required option to our program, and it will contain the name of a matrix containing the parameter values at which we are to evaluate the residual equation. All moment-evaluator programs must accept the *varlist*, *if* condition, and `at()` option.

The first part of our program computes  $\mu_{it}$ . In the model we will fit shortly, we have three regressors, named `x1`, `x2`, and `x3`. The `'at'` vector will have three elements, one for each of those variables. Notice that we included `'if'` at the end of each statement that affects variables to restrict the computations to the relevant estimation sample. The two `egen` statements compute  $\bar{\mu}_i$  and  $\bar{y}_i$ ; in the example dataset we will use shortly, the panel variable is named `id`, and for simplicity we hardcoded that variable into our program as well. Finally, we compute the residual equation, which is the portion of (13) bound in parentheses.

## ► Example 9: Panel poisson with strictly exogenous regressors

To fit our model, we type

```

. use http://www.stata-press.com/data/r11/poisson1
. gmm gmm_poi, nequations(1) parameters(b1 b2 b3)
> instruments(x1 x2 x3, noconstant) vce(cluster id) onestep

Step 1
Iteration 0: GMM criterion Q(b) = 51.99142
Iteration 1: GMM criterion Q(b) = .04345191
Iteration 2: GMM criterion Q(b) = 8.720e-06
Iteration 3: GMM criterion Q(b) = 7.115e-13
Iteration 4: GMM criterion Q(b) = 5.130e-27

```

## GMM estimation

Number of parameters = 3

Number of moments = 3

Initial weight matrix: Unadjusted

Number of obs = 409

(Std. Err. adjusted for 45 clusters in id)

|     | Coef.     | Robust<br>Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|-----|-----------|---------------------|--------|-------|----------------------|-----------|
| /b1 | 1.94866   | .1000265            | 19.48  | 0.000 | 1.752612             | 2.144709  |
| /b2 | -2.966119 | .0923592            | -32.12 | 0.000 | -3.14714             | -2.785099 |
| /b3 | 1.008634  | .1156561            | 8.72   | 0.000 | .781952              | 1.235315  |

Instruments for equation 1: x1 x2 x3

All three of our regressors are strictly exogenous, so they can serve as their own regressors. There is no constant term in the model (it would be unidentified), so we exclude a constant term from our list of instruments. We have one residual equation as indicated by `nequations(1)`, and we have three parameters, named `b1`, `b2`, and `b3`. The order in which you declare parameters in the `parameters()` option determines the order in which they appear in the `'at'` vector in the moment-evaluator program. We specified `vce(cluster id)` to obtain standard errors that allow for correlation among observations within each panel.

◀

## □ Technical note

The program we just wrote is sufficient to fit the model to the `poisson1` dataset, but if we want to fit that model to other datasets, we would need to change the variable names and perhaps account for having a different number of parameters as well. With a bit more programming, advanced users can write a more general program to work with arbitrary datasets. Any options not understood by `gmm` are passed along to the moment-evaluator program, and we can take advantage of that feature to pass the name of the dependent variable, list of regressors, and panel identifier variable. A better version of `gmm_poi` would be

```

program gmm_poi2
 version 11
 syntax varlist if, at(name) myrhs(varlist) ///
 mylhs(varlist) myidvar(varlist)
 quietly {
 tempvar mu mubar ybar
 gen double 'mu' = 0 'if'
 local j = 1
 foreach var of varlist 'myrhs' {
 replace 'mu' = 'mu' + 'var'*'at'[1,'j'] 'if'
 local j = 'j' + 1
 }
 replace 'mu' = exp('mu')
 egen double 'mubar' = mean('mu') 'if', by('myidvar')
 egen double 'ybar' = mean('mylhs') 'if', by('myidvar')
 replace 'varlist' = 'mylhs' - 'mu'*'ybar'/'mubar' 'if'
 }
end

```

Our program now accepts three more options. Our call to `gmm` is

```
. gmm gmm_poi2, mylhs(y) myrhs(x1 x2 x3) myidvar(id) nequations(1)
> parameters(b1 b2 b3) instruments(x1 x2 x3, noconstant) vce(cluster id) onestep
```

With `mylhs()`, `myrhs()`, and `myidvar()`, we are now able to fit our model to any panel dataset we wish without having to modify the `gmm_poi2` program. In the section [Details of moment-evaluator programs](#) below, we show how to incorporate weights and derivatives in moment-evaluator programs. □

When past values of the idiosyncratic error term affect the value of a regressor, we say that regressor is *predetermined*. When one or more regressors are predetermined, sample moment condition (9) is no longer valid. However, [Chamberlain \(1992\)](#) shows that a simple alternative is to consider moment conditions of the form

$$\sum_i \sum_{t=2}^T \mathbf{x}_{i,t-1} \left( y_{i,t-1} - \mu_{i,t-1} \frac{y_{it}}{\mu_{it}} \right) = \mathbf{0} \quad (14)$$

Also see [Wooldridge \(1997\)](#) and [Windmeijer \(2000\)](#) for other moment conditions that can be used with predetermined regressors.

## ► Example 10: Panel Poisson with predetermined regressors

Here we refit the previous model, treating all the regressors as predetermined and using the moment conditions in (10). Our moment-evaluator program is

```
program gmm_poiPRE
 version 11
 syntax varlist if, at(name) myrhs(varlist) ///
 mylhs(varlist) myidvar(varlist)

 quietly {
 tempvar mu mubar ybar
 gen double 'mu' = 0 'if'
 local j = 1
 foreach var of varlist 'myrhs' {
 replace 'mu' = 'mu' + 'var'*'at'[1,'j'] 'if'
 local j = 'j' + 1
 }
 replace 'mu' = exp('mu')
 replace 'varlist' = L.'mylhs' - L.'mu'*'mylhs'/'mu' 'if'
 }
end
```

As before, the first part of our program computes  $\mu_{it}$ ; the only difference is in how we compute the residual equation. We used lag-operator notation so that Stata properly handles gaps in our dataset. Equation (14) shows that we are to use the first lags of the regressors as instruments, so we type

```
. gmm gmm_poipre, mylhs(y) myrhs(x1 x2 x3) nequations(1) parameters(b1 b2 b3)
> instruments(L.(x1 x2 x3), noconstant) vce(cluster id) myidvar(id) onestep
(obs = 364)
```

Step 1

```
Iteration 0: GMM criterion Q(b) = 52.997808
Iteration 1: GMM criterion Q(b) = 2.1678071
Iteration 2: GMM criterion Q(b) = .08716503
Iteration 3: GMM criterion Q(b) = .00007136
Iteration 4: GMM criterion Q(b) = 4.699e-11
Iteration 5: GMM criterion Q(b) = 1.932e-23
```

GMM estimation

Number of parameters = 3

Number of moments = 3

Initial weight matrix: Unadjusted

Number of obs = 364

(Std. Err. adjusted for 45 clusters in id)

|     | Coef.     | Robust<br>Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|-----|-----------|---------------------|--------|-------|----------------------|-----------|
| /b1 | 2.035125  | .2662377            | 7.64   | 0.000 | 1.513308             | 2.556941  |
| /b2 | -2.929362 | .2290397            | -12.79 | 0.000 | -3.378272            | -2.480453 |
| /b3 | 1.235219  | .1673295            | 7.38   | 0.000 | .9072596             | 1.563179  |

Instruments for equation 1: L.x1 L.x2 L.x3

Here, like earlier with strictly exogenous regressors, the number of instruments equals the number of parameters, so there is no gain to using the two-step or iterated estimators. However, if you do have more instruments than parameters, you will most likely want to use one of those other estimators instead.

◀

In the [previous example](#), we used  $\mathbf{x}_{i,t-1}$  as instruments. A more efficient GMM estimator would also use  $\mathbf{x}_{i,t-2}, \mathbf{x}_{i,t-3}, \dots, \mathbf{x}_{i,1}$  as instruments in period  $t$  as well. `gmm`'s `xtinstruments()` option allows you to specify instrument lists that grow as  $t$  increases. Later we discuss the `xtinstruments()` option in detail in the context of linear dynamic panel-data models.

When a regressor is contemporaneously correlated with the idiosyncratic error term, we say that regressor is endogenous. [Windmeijer \(2000\)](#) shows that here we can use the moment condition

$$\sum_i \sum_{t=3}^T \mathbf{x}_{i,t-2} \left( \frac{y_{it}}{\mu_{it}} - \frac{y_{i,t-1}}{\mu_{i,t-1}} \right)$$

Here we use the second lag of the endogenous regressor as an instrument. If a variable is strictly exogenous, it can of course serve as its own instrument.

## ► Example 11: Panel Poisson with endogenous regressors

Here we refit the model, treating  $\mathbf{x}_3$  as endogenous and  $\mathbf{x}_1$  and  $\mathbf{x}_2$  as strictly exogenous. Our moment-evaluator program is

```
program gmm_poiend
 version 11
 syntax varlist if, at(name) myrhs(varlist) ///
 mylhs(varlist) myidvar(varlist)

 quietly {
 tempvar mu mubar ybar
 gen double 'mu' = 0 'if'
 local j = 1
 foreach var of varlist 'myrhs' {
 replace 'mu' = 'mu' + 'var'*'at'[1,'j'] 'if'
 local j = 'j' + 1
 }
 replace 'mu' = exp('mu')
 replace 'varlist' = 'mylhs'/'mu' - L.'mylhs'/L.'mu' 'if'
 }
end
```

Now we call gmm using x1, x2, and L2.x3 as instruments:

```
. use http://www.stata-press.com/data/r11/poisson2
. gmm gmm_poiend, mylhs(y) myrhs(x1 x2 x3) nequations(1) parameters(b1 b2 b3)
> instruments(x1 x2 L2.x3, noconstant) vce(cluster id) myidvar(id) onestep

Step 1
Iteration 0: GMM criterion Q(b) = 47.376537
Iteration 1: GMM criterion Q(b) = .08115406
Iteration 2: GMM criterion Q(b) = .03477036
Iteration 3: GMM criterion Q(b) = .00041056
Iteration 4: GMM criterion Q(b) = 1.189e-07
Iteration 5: GMM criterion Q(b) = 1.298e-14
Iteration 6: GMM criterion Q(b) = 1.574e-28

GMM estimation
Number of parameters = 3
Number of moments = 3
Initial weight matrix: Unadjusted Number of obs = 3766
 (Std. Err. adjusted for 500 clusters in id)
```

|     | Coef.     | Robust<br>Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|-----|-----------|---------------------|--------|-------|----------------------|-----------|
| /b1 | 1.844082  | .1515252            | 12.17  | 0.000 | 1.547098             | 2.141066  |
| /b2 | -2.904011 | .108117             | -26.86 | 0.000 | -3.115916            | -2.692105 |
| /b3 | 3.277512  | 2.459066            | 1.33   | 0.183 | -1.542169            | 8.097193  |

Instruments for equation 1: x1 x2 L2.x3

As with the predetermined case previously, instead of using just  $\mathbf{x}_{i,t-2}$  as an instrument, we could use all further lags of  $\mathbf{x}_{it}$  as instruments as well.



Rational-expectations models

Macroeconomic models typically assume that agents’ expectations about the future are formed rationally. By rational expectations, we mean that agents use all information available when forming their forecasts, so the forecast error is uncorrelated with the information available when the forecast was made. Say that at time  $t$ , people make a forecast,  $\hat{y}_{t+1}$ , of variable  $y$  in the next period. If  $\Omega_t$  denotes all available information at time  $t$ , then rational expectations implies that  $E\{(\hat{y}_{t+1} - y_{t+1})|\Omega_t\} = 0$ . If  $\Omega_t$  denotes observable variables such as interest rates or prices, then this conditional expectation can serve as the basis of a moment condition for GMM estimation.

## ► Example 12: Fitting a Euler equation

In a well-known article, [Hansen and Singleton \(1982\)](#) consider a model of portfolio decision making and discuss parameter estimation using GMM. We will consider a simple example with one asset in which the agent can invest. A consumer wants to maximize the present value of his lifetime utility derived from consuming the good. On the one hand, the consumer is impatient, so he would rather consume today than wait until tomorrow. On the other hand, if he consumes less today, he can invest more of his money, earning more interest that he can then use to consume more of the good tomorrow. Thus there is a tradeoff between having his cake today or sacrificing a bit today to have more cake tomorrow.

If we assume a specific form for the agent's utility function, known as the constant relative-risk aversion utility function, we can show that the Euler equation is

$$E \left[ z_t \left\{ 1 - \beta(1 + r_{t+1})(c_{t+1}/c_t)^{-\gamma} \right\} \right] = 0$$

where  $\beta$  and  $\gamma$  are the parameters to estimate,  $r_t$  is the return to the financial asset, and  $c_t$  is consumption in period  $t$ .  $\beta$  measures the agent's discount factor. If  $\beta$  is near one, the agent is patient and is more willing to forgo consumption this period. If  $\beta$  is close to zero, the agent is less patient and prefers to consume more now. The parameter  $\gamma$  characterizes the agent's utility function. If  $\gamma$  equals one, the utility function is linear. As  $\gamma$  tends toward zero, the utility function tends toward  $u = \log(c)$ .

We have data on 3-month Treasury bills ( $r_t$ ) and consumption expenditures ( $c_t$ ). As instruments, we will use lagged rates of return and past growth rates of consumption. We will use the two-step estimator and a weight matrix that allows for heteroskedasticity and autocorrelation up to four lags with the Bartlett kernel. In Stata, we type

```
. use http://www.stata-press.com/data/r11/cr
. generate cgrowth = c / L.c
(1 missing value generated)
. gmm (1 - {b=1}*(1+F.r)*(F.c/c)^(-1*{gamma=1})), inst(L.r L2.r cgrowth L.cgrowth)
> wmat(hac nw 4) twostep
warning: 1 missing value returned for equation 1 at initial values

Step 1
Iteration 0: GMM criterion Q(b) = .00226482
Iteration 1: GMM criterion Q(b) = .00054369
Iteration 2: GMM criterion Q(b) = .00053904
Iteration 3: GMM criterion Q(b) = .00053904

Step 2
Iteration 0: GMM criterion Q(b) = .0600729
Iteration 1: GMM criterion Q(b) = .0596369
Iteration 2: GMM criterion Q(b) = .0596369

GMM estimation
Number of parameters = 2
Number of moments = 5
Initial weight matrix: Unadjusted
GMM weight matrix: HAC Bartlett 4
 Number of obs = 239
```

|        | HAC       |           |       |       |                      |           |
|--------|-----------|-----------|-------|-------|----------------------|-----------|
|        | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
| /b     | .9204617  | .0134646  | 68.36 | 0.000 | .8940716             | .9468518  |
| /gamma | -4.222361 | 1.473895  | -2.86 | 0.004 | -7.111143            | -1.333579 |

HAC standard errors based on Bartlett kernel with 4 lags

Instruments for equation 1: L.r L2.r cgrowth L.cgrowth \_cons

The warning message at the top of the output appears because the forward operator in our substitutable expression says that residuals can be computed only for 239 observations; our dataset contains 240 observations. Our estimate of  $\beta$  is near one, in line with expectations and published results. ◀

## System estimators

In many economic models, two or more variables are determined jointly through a system of simultaneous equations. Indeed, some of the earliest work in econometrics, including that of the Cowles Commission, was centered around estimation of the parameters of simultaneous equations. The 2SLS and IV estimators we have already discussed are used in some circumstances to estimate such parameters. Here we focus on the joint estimation of all the parameters of systems of equations, and we begin with the well-known three-stage least-squares (3SLS) estimator.

Recall that the 2SLS estimator is based on the moment conditions  $E(\mathbf{z}u) = \mathbf{0}$ . The 2SLS estimator can be used to estimate the parameters of one equation of a system of structural equations. Moreover, with the 2SLS estimator, we do not even need to specify the structural relationship among all the endogenous variables; we need to specify only the equation on which interest focuses and simply assume reduced-form relationships among the endogenous regressors of the equation of interest and the exogenous variables of the model. If we are willing to specify the complete system of structural equations, then assuming our model is correctly specified, by estimating all the equations jointly, we can obtain estimates that are more efficient than equation-by-equation 2SLS.

In [R] [reg3](#), we fit a simple two-equation macroeconomic model:

$$\text{consump} = \beta_0 + \beta_1 \text{wagepriv} + \beta_2 \text{wagegovt} + \epsilon_1 \quad (15)$$

$$\text{wagepriv} = \beta_3 + \beta_4 \text{consump} + \beta_5 \text{govt} + \beta_6 \text{capital1} + \epsilon_2 \quad (16)$$

where `consump` represents aggregate consumption; `wagepriv` and `wagegovt` are total wages paid by the private and government sectors, respectively; `govt` is government spending; and `capital1` is the previous period's capital stock. We are not willing to assume that  $\epsilon_1$  and  $\epsilon_2$  are independent, so we must treat both `consump` and `wagepriv` as endogenous. Suppose that a random shock makes  $\epsilon_2$  positive. Then by (16), `wagepriv` will be higher than it otherwise would. Moreover,  $\epsilon_1$  will either be higher or lower, depending on the correlation between it and  $\epsilon_2$ . The shock to  $\epsilon_2$  has made both `wagepriv` and  $\epsilon_1$  move, implying that in (15) `wagepriv` is an endogenous regressor. A similar argument shows that `consump` is an endogenous regressor in the second equation. In our model, `wagegovt`, `govt`, and `capital1` are all exogenous variables.

Let  $\mathbf{z}_1$  and  $\mathbf{z}_2$  denote the instruments for the first and second equations, respectively; we will discuss what comprises them shortly. We have two sets of moment conditions:

$$E \left\{ \begin{array}{c} \mathbf{z}_1(\text{consump} - \beta_0 - \beta_1 \text{wagepriv} - \beta_2 \text{wagegovt}) \\ \mathbf{z}_2(\text{wagepriv} - \beta_3 - \beta_4 \text{consump} - \beta_5 \text{govt} - \beta_6 \text{capital1}) \end{array} \right\} = \mathbf{0} \quad (17)$$

One of the defining characteristics of 3SLS is that the errors are homoskedastic conditional on the instrumental variables. Using this assumption, we have

$$E \left[ \begin{array}{c} \mathbf{z}_1 \epsilon_1 \\ \mathbf{z}_2 \epsilon_2 \end{array} \right] \{ \mathbf{z}_1' \epsilon_1 \quad \mathbf{z}_2' \epsilon_2 \} = \begin{Bmatrix} \sigma_{11} E(\mathbf{z}_1 \mathbf{z}_1') & \sigma_{12} E(\mathbf{z}_1 \mathbf{z}_2') \\ \sigma_{21} E(\mathbf{z}_2 \mathbf{z}_1') & \sigma_{22} E(\mathbf{z}_2 \mathbf{z}_2') \end{Bmatrix} \quad (18)$$

where  $\sigma_{ij} = \text{cov}(\epsilon_i, \epsilon_j)$ . Let  $\Sigma$  denote the  $2 \times 2$  matrix with typical element  $\sigma_{ij}$ .



The second defining characteristic of the 3SLS estimator is that it uses all the exogenous variables as instruments for all equations; here  $\mathbf{z}_1 = \mathbf{z}_2 = (\text{wagegovt}, \text{govt}, \text{capital1}, 1)$ , where the 1 indicates a constant term. From our discussion on the weight matrix and two-step estimation, we want to use the sample analogue of the matrix inverse of the right-hand side of (18) as our weight matrix.

To implement the 3SLS estimator, we apparently need to know  $\Sigma$  or at least have a consistent estimator of it. The solution is to fit (15) and (16) by 2SLS, use the sample residuals  $\hat{\epsilon}_1$  and  $\hat{\epsilon}_2$  to estimate  $\Sigma$ , then estimate the parameters of (17) via GMM by using the weight matrix just discussed.

### ► Example 13: 3SLS estimation

3SLS is easier to do using `gmm` than it sounds. The 3SLS estimator is a two-step GMM estimator. In the first step, we do the equivalent of 2SLS on each equation, and then we compute a weight matrix based on (18). Finally, we perform a second step of GMM with this weight matrix.

In Stata, we type

```
. use http://www.stata-press.com/data/r11/klein, clear
. gmm (eq1:consump - {b0} - {xb: wagepriv wagegovt})
> (eq2:wagepriv - {c0} - {xc: consump govt capital1}),
> instruments(eq1 eq2: wagegovt govt capital1) winitial(unadjusted, independent)
> wmatrix(unadjusted) twostep
```

Step 1

```
Iteration 0: GMM criterion Q(b) = 4195.4487
Iteration 1: GMM criterion Q(b) = .22175631
Iteration 2: GMM criterion Q(b) = .22175631
```

Step 2

```
Iteration 0: GMM criterion Q(b) = .09716589
Iteration 1: GMM criterion Q(b) = .07028208
Iteration 2: GMM criterion Q(b) = .07028208
```

GMM estimation

```
Number of parameters = 7
Number of moments = 8
Initial weight matrix: Unadjusted
GMM weight matrix: Unadjusted
Number of obs = 22
```

|              | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|--------------|-----------|-----------|-------|-------|----------------------|----------|
| /b0          | 19.3559   | 3.583772  | 5.40  | 0.000 | 12.33184             | 26.37996 |
| /xb_wagepriv | .8012754  | .1279329  | 6.26  | 0.000 | .5505314             | 1.052019 |
| /xb_wagegovt | 1.029531  | .3048424  | 3.38  | 0.001 | .432051              | 1.627011 |
| /c0          | 14.63026  | 10.26693  | 1.42  | 0.154 | -5.492552            | 34.75306 |
| /xc_consump  | .4026076  | .2567312  | 1.57  | 0.117 | -.1005764            | .9057916 |
| /xc_govt     | 1.177792  | .5421253  | 2.17  | 0.030 | .1152461             | 2.240338 |
| /xc_capital1 | -.0281145 | .0572111  | -0.49 | 0.623 | -.1402462            | .0840173 |

```
Instruments for equation 1: wagegovt govt capital1 _cons
Instruments for equation 2: wagegovt govt capital1 _cons
```

The `independent` suboption of the `winitial()` option tells `gmm` to assume that the residuals are independent across equations; this suboption sets  $\sigma_{21} = \sigma_{12} = 0$  in (18). Assuming both homoskedasticity and cross-equation independence is equivalent to fitting the two equations of our model independently by 2SLS. The `wmatrix()` option controls how the weight matrix is computed based on the first-step parameter estimates before the second step of estimation; here we request a weight matrix that assumes conditional homoskedasticity but that does not impose the cross-equation independence like the initial weight matrix we used. In this example, we also illustrated how to

name equations and how equation names can be used in the `instruments()` option. Our results are identical to those in [R] [reg3](#).

We could have specified our instruments with the syntax

```
instruments(wagegovt govt capital1)
```

because `gmm` uses the variables listed in the `instruments()` option for all equations unless you use the `equations()` suboption to restrict those variables to certain equations. However, we wanted to emphasize that the same instruments are being used for both equations; in a moment, we will discuss an estimator that does not use the same instruments in all equations.

◀

In the [previous example](#), if we omit the `twostep` option, the resulting coefficients will be equivalent to equation-by-equation 2SLS, which [Wooldridge \(2002, 191\)](#) calls the “system 2SLS estimator”. Eliminating the `twostep` option makes the `wmatrix()` option irrelevant, so that option can be eliminated as well.

So far, we have developed the traditional 3SLS estimator. [Wooldridge \(2002, chap. 8\)](#) discusses the “GMM 3SLS” estimator that extends the traditional 3SLS estimator by allowing for heteroskedasticity and different instruments for different equations.

Generalizing (18) to an arbitrary number of equations, we have

$$E(\mathbf{Z}'\epsilon\epsilon'\mathbf{Z}) = E(\mathbf{Z}'\Sigma\mathbf{Z}) \tag{19}$$

where

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{z}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{z}_m \end{bmatrix}$$

and  $\Sigma$  is now  $m \times m$ . Equation (19) is the multivariate analogue of a homoskedasticity assumption; for each equation, the error variance is constant for all observations, as is the covariance between any two equations’ errors.

We can relax this homoskedasticity assumption by considering different weight matrices. For example, if we continue to assume that observations are independent but not necessarily identically distributed, then by specifying `wmatrix(robust)`, we would obtain a weight matrix that allows for heteroskedasticity:

$$\widehat{W} = \frac{1}{N} \sum_i \mathbf{z}_i' \widehat{\epsilon}_i \widehat{\epsilon}_i' \mathbf{z}_i$$

This is the weight matrix in [Wooldridge’s \(2002, 193\) Procedure 8.1](#), “GMM with Optimal Weighting Matrix”. By default, `gmm` would report standard errors based on his covariance matrix (8.27); specifying `vce(unadjusted)` would provide the optimal GMM standard errors. If you have multiple observations for each individual or firm in your dataset, you could specify `wmatrix(cluster id)`, where *id* identifies individuals or firms. This would allow arbitrary within-individual correlation, though it does not account for an individual-specific fixed or random effect. In both cases, we would continue to use `initial(unadjusted, independent)` so that the first-step estimates are the system 2SLS estimates.

[Wooldridge \(2002, sec. 9.6\)](#) discusses instances where it is necessary to use different instruments in different equations. The GMM 3SLS estimator with different instruments in different equations but with conditional homoskedasticity is what [Hayashi \(2000, 275\)](#) calls the “full-information instrumental

variables efficient” (FIVE) estimator. Implementing the FIVE estimator is easy with `gmm`. For example, say we have a two-equation system, where `kids`, `age`, `income`, and `education` are all valid instruments for the first equation; but `education` is not a valid instrument for the second equation. Then our syntax would take the form

```
gmm (<mexp_1>) (<mexp_2>), instruments(1:kids age income education)
 instruments(2:kids age income)
```

The following syntax is equivalent:

```
gmm (<mexp_1>) (<mexp_2>), instruments(kids age income)
 instruments(1:education)
```

Because we did not specify a list of equations in the second example’s first `instruments()` option, those variables are used as instruments in both equations. You can use whichever syntax you prefer. The first requires a bit more typing but is arguably more transparent.

If all the regressors in the model are exogenous, then the traditional 3SLS estimator is the seemingly unrelated regression (SUR) estimator. Here you would specify all the regressors as instruments.

## Dynamic panel-data models

Commands in Stata that work with panel data expect the data to be in the “long” format, meaning that each row of the dataset consists of one subobservation that is a member of a logical observation (represented by the panel identifier variable). See [D] [reshape](#) for a discussion of the long versus “wide” data forms. `gmm` is no exception in this respect when used with panel data. From a theoretical perspective, however, it is sometimes easier to view GMM estimators for panel data as system estimators in which we have  $N$  observations on a system of  $T$  equations, where  $N$  and  $T$  are the number of observations and panels, respectively, rather than a single-equation estimator with  $NT$  observations. Usually, each of the  $T$  equations will in fact be the same, though we will want to specify different instruments for each of these equations.

In a dynamic panel-data model, lagged values of the dependent variable are included as regressors. Here we consider a simple model with one lag of the dependent variable  $y$  as a regressor and a vector of strictly exogenous regressors,  $\mathbf{x}_{it}$ :

$$y_{it} = \rho y_{i,t-1} + \mathbf{x}'_{it}\beta + u_i + \epsilon_{it} \quad (20)$$

$u_i$  can be either a fixed- or a random-effect term, in the sense that we do not require  $\mathbf{x}_{it}$  to be independent of it. Even with the assumption that  $\epsilon_{it}$  is i.i.d., the presence of both  $y_{i,t-1}$  and  $u_i$  in (20) renders both the standard fixed- and random-effects estimators to be inconsistent because of the well-known [Nickell \(1981\)](#) bias. OLS regression of  $y_{it}$  on  $y_{i,t-1}$  and  $\mathbf{x}_{it}$  also produces inconsistent estimates, because  $y_{i,t-1}$  will be correlated with the error term.

### □ Technical note

Stata has the `xtabond`, `xtdpd`, and `xtdpdsys` commands (see [XT] [xtabond](#), [XT] [xtdpd](#), and [XT] [xtdpdsys](#)) to fit equations like (20), and for everyday use those commands are preferred because they offer features such as [Windmeijer \(2005\)](#) bias-corrected standard errors to account for the bias of traditional two-step GMM standard errors seen in dynamic panel-data models and, being linear estimators, only require you to specify variable names instead of complete equations. However, using `gmm` has several pedagogical advantages, including the ability to tie those model-specific commands into a more general framework, a clear illustration of how certain types of instrument matrices for panel-data models are formed, and demonstrations of several advanced features of `gmm`.

□

First-differencing (20) removes the panel-specific  $u_i$  term:

$$y_{it} - y_{i,t-1} = \rho(y_{i,t-1} - y_{i,t-2}) + (\mathbf{x}_{it} - \mathbf{x}_{i,t-1})'\beta + (\epsilon_{it} - \epsilon_{i,t-1}) \quad (21)$$

However, now  $(y_{i,t-1} - y_{i,t-2})$  is correlated with  $(\epsilon_{it} - \epsilon_{i,t-1})$ . Thus we need an instrument that is correlated with the former but not the latter. The lagged variables in (21) mean that equation is not estimable for  $t < 3$ , so consider when  $t = 3$ . We have

$$y_{i3} - y_{i2} = \rho(y_{i2} - y_{i1}) + (\mathbf{x}_{i3} - \mathbf{x}_{i2})'\beta + (\epsilon_{i3} - \epsilon_{i2}) \quad (22)$$

In the Arellano–Bond (1991) estimator, lagged levels of the dependent variable are used as instruments. With our assumption that the  $\epsilon_{it}$  are i.i.d., (20) intimates that  $y_{i1}$  can serve as an instrumental variable when we fit (22).

Next consider (21) when  $t = 4$ . We have

$$y_{i4} - y_{i3} = \rho(y_{i3} - y_{i2}) + (\mathbf{x}_{i4} - \mathbf{x}_{i3})'\beta + (\epsilon_{i4} - \epsilon_{i3})$$

Now (20) shows that both  $y_{i1}$  and  $y_{i2}$  are uncorrelated with the error term  $(\epsilon_{i4} - \epsilon_{i3})$ , so we have two instruments available. For  $t = 5$ , you can show that  $y_{i1}$ ,  $y_{i2}$ , and  $y_{i3}$  can serve as instruments. As may now be apparent, one of the key features of these dynamic panel-data models is that the available instruments depend on the time period,  $t$ , as was the case for some of the panel Poisson models we considered earlier. Because the  $\mathbf{x}_{it}$  are strictly exogenous by assumption, they can serve as their own instruments.

The initial weight matrix that is appropriate for the GMM dynamic panel-data estimator is slightly more involved than the unadjusted matrix we have used in most of our previous examples that assumes the errors are i.i.d. First, rewrite (21) for panel  $i$  as

$$\mathbf{y}_i - \mathbf{y}_i^L = \rho(\mathbf{y}_i^L - \mathbf{y}_i^{LL}) + (\mathbf{X}_i - \mathbf{X}_i^L)\beta + (\epsilon_i - \epsilon_i^L)$$

where  $\mathbf{y}_i = (y_{i3}, \dots, y_{iT})$  and  $\mathbf{y}_i^L = (y_{i2}, \dots, y_{i,T-1})$ ,  $\mathbf{y}_i^{LL} = (y_{i1}, \dots, y_{i,T-2})$ , and  $\mathbf{X}_i$ ,  $\mathbf{X}_i^L$ ,  $\epsilon_i$ , and  $\epsilon_i^L$  are defined analogously. Let  $\mathbf{Z}$  denote the full matrix of instruments for panel  $i$ , including the variables specified in both the `instruments()` and `xtinstruments()` options; the exact structure is detailed in [Methods and formulas](#).

By assumption,  $\epsilon_{it}$  is i.i.d., so the first-difference  $(\epsilon_{it} - \epsilon_{i,t-1})$  is necessarily autocorrelated with correlation  $-0.5$ . Therefore, we should not use a weight matrix that assumes the errors are independent. For dynamic panel-data models, we can show that the appropriate initial weight matrix is

$$\widehat{\mathbf{W}} = \left( \frac{1}{N} \sum_i \mathbf{Z}_i' \mathbf{H}_D \mathbf{Z}_i \right)^{-1}$$

where

$$\mathbf{H}_D = \begin{bmatrix} 1 & -0.5 & 0 & \dots & 0 & 0 \\ -0.5 & 1 & -0.5 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & -0.5 \\ 0 & 0 & 0 & \dots & -0.5 & 1 \end{bmatrix}$$

We can obtain this initial weight matrix by specifying `winitial(xt D)`. The letter D indicates that the equation we are estimating is specified in first-differences.

## ► Example 14: Arellano–Bond estimator

Say we want to fit the model

$$\mathbf{n}_{it} = \rho \mathbf{n}_{i,t-1} + \beta_1 \mathbf{w}_{it} + \beta_2 \mathbf{w}_{i,t-1} + \beta_3 \mathbf{k}_{it} + \beta_4 \mathbf{k}_{i,t-1} + u_i + \epsilon_{it} \quad (23)$$

where we assume that  $\mathbf{w}_{it}$  and  $\mathbf{k}_{it}$  are strictly exogenous. First-differencing, our residual equation is

$$\begin{aligned} \epsilon_{it}^* = (\epsilon_{it} - \epsilon_{i,t-1}) = & \mathbf{n}_{it} - \mathbf{n}_{i,t-1} - \rho (\mathbf{n}_{i,t-1} - \mathbf{n}_{i,t-2}) - \beta_1 (\mathbf{w}_{it} - \mathbf{w}_{i,t-1}) \\ & - \beta_2 (\mathbf{w}_{i,t-1} - \mathbf{w}_{i,t-2}) - \beta_3 (\mathbf{k}_{it} - \mathbf{k}_{i,t-1}) - \beta_4 (\mathbf{k}_{i,t-1} - \mathbf{k}_{i,t-2}) \end{aligned} \quad (24)$$

In Stata, we type

```
. use http://www.stata-press.com/data/r11/abdata
. gmm (D.n - {rho}*LD.n - {xb:D.w LD.w D.k LD.k}), xtinstruments(n, lags(2/.))
> instruments(D.w LD.w D.k LD.k, noconstant) deriv(/rho = -1*LD.n)
> deriv(/xb = -1) winitial(xt D) onestep
```

Step 1

Iteration 0: GMM criterion Q(b) = .0011455

Iteration 1: GMM criterion Q(b) = .00009103

Iteration 2: GMM criterion Q(b) = .00009103

GMM estimation

Number of parameters = 5

Number of moments = 32

Initial weight matrix: XT D

Number of obs = 751

|          | Coef.     | Robust<br>Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|---------------------|-------|-------|----------------------|-----------|
| /rho     | .8041712  | .1199819            | 6.70  | 0.000 | .5690111             | 1.039331  |
| /xb_D_w  | -.5600476 | .1619472            | -3.46 | 0.001 | -.8774583            | -.242637  |
| /xb_LD_w | .3946699  | .1092229            | 3.61  | 0.000 | .1805969             | .6087429  |
| /xb_D_k  | .3520286  | .0536546            | 6.56  | 0.000 | .2468676             | .4571897  |
| /xb_LD_k | -.2160435 | .0679689            | -3.18 | 0.001 | -.3492601            | -.0828269 |

Instruments for equation 1:

XT-style: L(2/.)n

Standard: D.w LD.w D.k LD.k

Because  $\mathbf{w}$  and  $\mathbf{k}$  are strictly exogenous, we specified the variants of them that appear in (24) in the `instruments()` option; because there is no constant term in the model, we specified `noconstant` to omit the constant from the instrument list.

We specified `xtinstruments(n, lags(2/.))` to tell `gmm` what instruments to use for the lagged dependent variable included as a regressor in (23). Based on our previous discussion, lags two and higher of  $\mathbf{n}_{it}$  can serve as instruments. The `lags(2/.)` suboption tells `gmm` that the first available instrument for  $\mathbf{n}_{it}$  is the lag-two value  $\mathbf{n}_{i,t-2}$ . The “.” tells `gmm` to use all further lags of  $\mathbf{n}_{it}$  as instruments as well. The instrument matrices in dynamic panel-data models can become large if the dataset has many time periods per panel. In those cases, you could specify, for example, `lags(2/4)` to use just lags two through four instead of using all available lags.

Our results are identical to those we would obtain using `xtabond` with the syntax

```
xtabond n L(0/1).w L(0/1).k, lags(1) noconstant vce(robust)
```

Had we left off the `vce(robust)` option in our call to `xtabond`, we would have had to specify `vce(unadjusted)` in our call to `gmm` to obtain the same standard errors.

□ Technical note

`gmm` automatically excludes observations for which there are no valid observations for the panel-style instruments. However, it keeps in the estimation sample those observations for which fewer than the maximum number of instruments you requested are available. For example, if you specify the `lags(2/4)` suboption, you have requested three instruments, but `gmm` will keep observations even if only one or two instruments are available.

► Example 15: Two-step Arellano–Bond estimator

Here we refit the model in the previous example, using the two-step GMM estimator.

```
. gmm (D.n - {rho}*LD.n - {xb:D.w LD.w D.k LD.k}),
> xtinstruments(n, lags(2/.)) instruments(D.w LD.w D.k LD.k, noconstant)
> deriv(/rho = -1*LD.n) deriv(/xb = -1) winitial(xt D) wmatrix(robust)
> vce(unadjusted)

Step 1
Iteration 0: GMM criterion Q(b) = .0011455
Iteration 1: GMM criterion Q(b) = .00009103
Iteration 2: GMM criterion Q(b) = .00009103

Step 2
Iteration 0: GMM criterion Q(b) = .44107941
Iteration 1: GMM criterion Q(b) = .4236729
Iteration 2: GMM criterion Q(b) = .4236729

GMM estimation
Number of parameters = 5
Number of moments = 32
Initial weight matrix: XT D
GMM weight matrix: Robust
Number of obs = 751
```

|          | Coef.     | Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|----------|-----------|-----------|--------|-------|----------------------|-----------|
| /rho     | .8044783  | .0534763  | 15.04  | 0.000 | .6996667             | .90929    |
| /xb_D_w  | -.5154978 | .0335506  | -15.36 | 0.000 | -.5812557            | -.4497399 |
| /xb_LD_w | .4059309  | .0637294  | 6.37   | 0.000 | .2810235             | .5308384  |
| /xb_D_k  | .3556204  | .0390892  | 9.10   | 0.000 | .2790071             | .4322337  |
| /xb_LD_k | -.2204521 | .046439   | -4.75  | 0.000 | -.3114709            | -.1294332 |

```
Instruments for equation 1:
 XT-style: L(2/.).n
 Standard: D.w LD.w D.k LD.k
```

Our results match those you would obtain with the command

```
xtabond n L(0/1).(w k), lags(1) noconstant twostep
```

□ Technical note

Had we specified `vce(robust)` in our call to `gmm`, we would have obtained the traditional sandwich-based robust covariance matrix, but our standard errors would not match those we would obtain by specifying `vce(robust)` with the `xtabond` command. The `xtabond`, `xtdpd`, and `xtdpdsys` commands implement a bias-corrected robust VCE for the two-step GMM dynamic panel-data estimator. Traditional VCEs computed after the two-step dynamic panel-data estimator have been shown to exhibit often-severe bias; see [Windmeijer \(2005\)](#).

Neither of the two dynamic panel-data examples we have fit so far include a constant term. When a constant term is included, the dynamic panel-data estimator is in fact a two-equation system estimator. For notational simplicity, consider a simple model containing just a constant term and one lag of the dependent variable:

$$y_{it} = \alpha + \rho y_{i,t-1} + u_i + \epsilon_{it}$$

First-differencing to remove the  $u_i$  term, we have

$$y_{it} - y_{i,t-1} = \rho(y_{i,t-1} - y_{i,t-2}) + (\epsilon_{it} - \epsilon_{i,t-1}) \quad (25)$$

This has also eliminated the constant term. If we assume  $E(u_i) = 0$ , which is reasonable if a constant term is included in the model, then we can recover  $\alpha$  by including the moment condition

$$y_{it} = \alpha + \rho y_{i,t-1} + \epsilon'_{it} \quad (26)$$

where  $\epsilon'_{it} = u_i + \epsilon_{it}$ . The parameter  $\rho$  continues to be identified by (25), so the only instrument we use with (26) is a constant term. As before, the error term  $(\epsilon_{i,t} - \epsilon_{i,t-1})$  is necessarily autocorrelated with correlation coefficient  $-0.5$ , though the error term  $\epsilon'_{it}$  is white noise. Therefore, our initial weight matrix should be

$$\widehat{\mathbf{W}} = \left( \frac{1}{N} \sum_i \mathbf{z}'_i \mathbf{H} \mathbf{z}_i \right)^{-1}$$

where

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_D & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

and  $\mathbf{I}$  is a conformable identity matrix.

One complication arises concerning the relevant estimation sample. Looking at (25), we apparently lose the first two observations from each panel because of the presence of  $y_{i,t-2}$ , but in (26) we need only to sacrifice one observation, for  $y_{i,t-1}$ . For most multiple-equation models, we need to use the same estimation sample for all equations. However, in dynamic panel-data models, we can use more observations to fit the equation in level form [(26) here] than the equation in first-differences [equation (25)]. To request this behavior, we specify the `nocommonesample` option to `gmm`. That option tells `gmm` to use as many observations as possible for each equation, ignoring the loss of observations due to lagging or differencing.

## ► Example 16: Arellano–Bond estimator with constant term

Here we fit the model

$$\mathbf{n}_{it} = \alpha + \rho \mathbf{n}_{i,t-1} + u_i + \epsilon_{it}$$

Without specifying derivatives, our command would be

```
. gmm (D.n - {rho}*LD.n) (n - {alpha} - {rho}*L.n),
> xtstruments(1: n, lags(2/.)) instruments(1:, noconstant) onestep
> winitial(xt DL) vce(unadj) nocommonesample
```

We would specify `winitial(xt DL)` to obtain the required initial weight matrix. The notation `DL` indicates that our first moment equation is in first-differences and the second moment equation is in levels (not first-differenced). We exclude a constant in the instrument list for the first equation, because first-differencing removed the constant term. Because we do not specify the `instruments()` option for the second moment equation, a constant is used by default.

This example also provides us the opportunity to illustrate how to specify derivatives for multiple-equation GMM models. Within the `derivative()` option, instead of specifying just the parameter name, now you must specify the equation name or number, a slash, and the parameter name to which the derivative applies. In Stata, we type

```
. gmm (D.n - {rho}*LD.n) (n - {alpha} - {rho}*L.n),
> xtinstruments(1: n, lags(2/.)) instruments(1:, noconstant)
> derivative(1/rho = -1*LD.n) derivative(2/alpha = -1)
> derivative(2/rho = -1*L.n) winitial(xt DL) vce(unadj) nocommonesample onestep
```

```
Step 1
Iteration 0: GMM criterion Q(b) = .09894466
Iteration 1: GMM criterion Q(b) = .00023508
Iteration 2: GMM criterion Q(b) = .00023508
```

```
GMM estimation
Number of parameters = 2
Number of moments = 29
Initial weight matrix: XT DL
Number of obs = *
```

|        | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|--------|-----------|-----------|-------|-------|----------------------|----------|
| /rho   | 1.023349  | .0608293  | 16.82 | 0.000 | .9041259             | 1.142572 |
| /alpha | -.0690864 | .0660343  | -1.05 | 0.295 | -.1985112            | .0603384 |

```
* Number of observations for equation 1: 751
Number of observations for equation 2: 891
```

```
Instruments for equation 1:
XT-style: L(2/.)n
Instruments for equation 2:
Standard: _cons
```

These results are identical to those we would obtain by typing

```
xtabond n, lags(1)
```

Because we specified `nocommonesample`, `gmm` did not report the number of observations used in the header of the output. In this dataset, there are in fact 1,031 observations on 140 panels. In the second equation, the presence of the lagged value of `n` reduces the sample size for that equation to  $1031 - 140 = 891$ . In the first equation, we lose the first two observations per panel due to lagging and differencing, leading to 751 usable observations. These tallies are listed after the coefficient table in the output.



□ Technical note

Specifying

```
xtinstruments(x1 x2 x3, lags(1/3))
```

differs from

```
instruments(L(1/3).(x1 x2 x3))
```

in how observations are excluded from the estimation sample. When you use the latter syntax, `gmm` must exclude the first three observations from each panel when computing the moment equation: you requested three lags of each regressor be used as instruments, so the first residual that could be interacted with those instruments is the one for  $t = 4$ . On the other hand, when you use `xtinstruments()`, you are telling `gmm` that you would like to use up to the first three lags of `x1`, `x2`, and `x3` as instruments



but that using just one lag is acceptable. Because most panel datasets have a relatively modest number of observations per panel, dynamic instrument lists are typically used so that the number of usable observations is maximized. Dynamic instrument lists also accommodate the fact that there are more valid instruments for later time periods than earlier time periods.

Specifying panel-style instruments using the `xtinstruments()` option also affects how the standard instruments specified in the `instruments()` option are treated. To illustrate, suppose we have a balanced panel dataset with  $T = 5$  observations per panel and we specify

```
. gmm ..., xtinstruments(w, lags(1/2)) instruments(x)
```

We will lose the first observation because we need at least one lag of  $w$  to serve as an instrument. Our instrument matrix for panel  $i$  will therefore be

$$\mathbf{Z}_i = \begin{bmatrix} w_{i1} & 0 & 0 & 0 \\ 0 & w_{i1} & 0 & 0 \\ 0 & w_{i2} & 0 & 0 \\ 0 & 0 & w_{i2} & 0 \\ 0 & 0 & w_{i3} & 0 \\ 0 & 0 & 0 & w_{i3} \\ 0 & 0 & 0 & w_{i4} \\ x_{i2} & x_{i3} & x_{i4} & x_{i5} \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad (27)$$

The vector of ones in the final row represents the constant term implied by the `instruments()` option. Because we lost the first observation, the residual vector  $\mathbf{u}_i$  will be  $4 \times 1$ . Thus our moment conditions for the  $i$ th panel can be written in matrix notation as

$$E\{\mathbf{Z}_i \mathbf{u}_i(\beta)\} = E\left\{\mathbf{Z}_i \begin{bmatrix} u_{i2}(\beta) \\ u_{i3}(\beta) \\ u_{i4}(\beta) \\ u_{i5}(\beta) \end{bmatrix}\right\} = \mathbf{0}$$

The moment conditions corresponding to the final two rows of (27) say that

$$E\left\{\sum_{t=2}^{T=4} x_{it} u_{it}(\beta)\right\} = 0 \quad \text{and} \quad E\left\{\sum_{t=2}^{T=4} u_{it}(\beta)\right\} = 0$$

Because we specified panel-style instruments with the `xtinstruments()` option, `gmm` no longer uses moment conditions for strictly exogenous variables of the form  $E\{x_{it} u_{it}(\beta)\} = 0$  for each  $t$ . Instead, the moment conditions now stipulate that the average (over  $t$ ) of  $x_{it} u_{it}(\beta)$  has expectation zero. This corresponds to the approach proposed by [Arellano and Bond \(1991, 280\)](#) and others.

When you request panel-style instruments with the `xtinstruments()` option, the number of instruments in the  $\mathbf{Z}_i$  matrix increases quadratically in the number of periods. The dynamic panel-data estimators we have discussed in this section are designed for datasets that contain a large number of panels and a modest number of time periods. When the number of time periods is large, estimators that use standard (non-panel-style) instruments are more appropriate.

□

We have focused on the Arellano–Bond dynamic panel-data estimator because of its relative simplicity. `gmm` can additionally fit any models that can be formulated using the `xtdpd` and `xtdpdsys` commands; see [XT] [xtdpd](#) and [XT] [xtdpdsys](#). The key is to determine the appropriate instruments to use for the level and difference equations. You may find it useful to fit a version of your model with those commands to determine what instruments and XT-style instruments to use. We conclude this section with an example using the Arellano–Bover/Blundell–Bond estimator.

➤ Example 17: Arellano–Bover/Blundell–Bond estimator

We fit a small model that includes one lag of the dependent variable `n` as a regressor as well as the contemporaneous and first lag of `w`, which we assume are strictly exogenous. We could fit our model using `xtdpdsys` using the syntax

```
xtdpdsys n L(0/1).w, lags(1) twostep
```

Applying virtually all the syntax issues we have discussed so far, the equivalent `gmm` command is

```
. gmm (n - {rho}*L.n - {w}*w - {lagw}*L.w - {c})
> (D.n - {rho}*LD.n - {w}*D.w - {lagw}*LD.w),
> xtinst(1: D.n, lags(1/1)) xtinst(2: n, lags(2/.))
> inst(2: D.w LD.w, noconstant)
> deriv(1/rho = -1*L.n)
> deriv(1/w = -1*w)
> deriv(1/lagw = -1*L.w)
> deriv(1/c = -1)
> deriv(2/rho = -1*LD.n)
> deriv(2/w = -1*D.w)
> deriv(2/lagw = -1*LD.w)
> winit(xt LD) wmatrix(robust) vce(unadjusted) nocommonesample
```

```
Step 1
Iteration 0: GMM criterion Q(b) = .10170339
Iteration 1: GMM criterion Q(b) = .00022772
Iteration 2: GMM criterion Q(b) = .00022772
```

```
Step 2
Iteration 0: GMM criterion Q(b) = .59965014
Iteration 1: GMM criterion Q(b) = .56578186
Iteration 2: GMM criterion Q(b) = .56578186
```

```
GMM estimation
Number of parameters = 4
Number of moments = 39
Initial weight matrix: XT LD
GMM weight matrix: Robust

Number of obs = *
```

|       | Coef.     | Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|-------|-----------|-----------|--------|-------|----------------------|-----------|
| /rho  | 1.122738  | .0206512  | 54.37  | 0.000 | 1.082263             | 1.163214  |
| /w    | -.6719909 | .0246148  | -27.30 | 0.000 | -.7202351            | -.6237468 |
| /lagw | .571274   | .0403243  | 14.17  | 0.000 | .4922398             | .6503083  |
| /c    | .154309   | .17241    | 0.90   | 0.371 | -.1836084            | .4922263  |

```
* Number of observations for equation 1: 891
Number of observations for equation 2: 751
```

```
Instruments for equation 1:
 XT-style: LD.n
 Standard: _cons
Instruments for equation 2:
 XT-style: L(2/.)n
 Standard: D.w LD.w
```

## Details of moment-evaluator programs

In examples 9, 10, and 11, we used moment-evaluator programs to evaluate moment conditions that could not be specified using the interactive version of `gmm`. In the [technical note](#) after example 9, we showed how to make our program accept optional arguments so that we could pass the name of our dependent and independent variables and panel identifier variable. Here we discuss how to make moment-evaluator programs accept weights and provide derivatives.

The complete specification for a moment-evaluator program's `syntax` statement is

```
syntax varlist if [weight], at(name) options [derivatives(varlist)]
```

The macro `'varlist'` contains the list of variables that we are to fill in with the values of our residual equations. The macro `'if'` represents an if condition that restricts the estimation sample. The macro `'at'` represents a vector containing the parameter values at which we are to evaluate our residual equations. `options` represent other options that you specify in your call to `gmm` and want to have passed to your moment-evaluator programs. In our previous examples, we included the `mylhs()`, `myrhs()`, and `myidvar()` options.

Two new elements of the `syntax` statement allow for weights and derivatives. `weight` specifies the types of weights your program allows. The interactive version of `gmm` allows for `fweights`, `awweights`, and `pweights`. However, unless you explicitly allow your moment evaluator program to accept weights, you cannot specify weights in your call to `gmm` with the moment-evaluator program version.

The `derivatives()` option is used to pass to your program a set of variables that you are to fill in with the derivatives of your residual equations with respect to the parameters. Say you specify  $k$  parameters in the `nparameters()` or `parameters()` option and  $q$  equations in the `nequations()` or `equations()` option. Then `'derivatives'` will contain  $k \times q$  variables. The first  $k$  variables are for the derivatives of the first residual equation with respect to the  $k$  parameters, the second  $k$  variables are for the derivatives of the second residual equation, and so on.

### ► Example 18: Panel Poisson with strictly exogenous regressors and derivatives

To focus on how to specify derivatives, we return to the simple moment-evaluator program we used in [example 9](#), in which we had three regressors, and extend it to supply derivatives. The error equation corresponding to moment condition (13) is

$$u_{it}(\beta) = y_{it} - \mu_{it} \frac{\bar{y}_i}{\bar{\mu}_i}$$

where  $\mu_{it}$ ,  $\bar{\mu}_i$ , and  $\bar{y}_i$  were defined previously. Now

$$\frac{\partial}{\partial \beta_j} u_{it}(\beta) = -\mu_{it} \frac{\bar{y}_i}{\bar{\mu}_i^2} \left( x_{it}^{(j)} \bar{\mu}_i - \frac{1}{T} \sum_{l=1}^{l=T} x_{il}^{(j)} \mu_{il} \right) \quad (28)$$

where  $x_{it}^{(j)}$  represents the  $j$ th element of  $\mathbf{x}_{it}$ .

Our moment-evaluator program is

```

program gmm_poideriv
 version 11
 syntax varlist if, at(name) [derivatives(varlist)]
 quietly {
 // Calculate residuals as before
 tempvar mu mubar ybar
 gen double 'mu' = exp(x1*'at'[1,1] + x2*'at'[1,2] + x3*'at'[1,3]) 'if' ///
 egen double 'mubar' = mean('mu') 'if', by(id)
 egen double 'ybar' = mean(y) 'if', by(id)
 replace 'varlist' = y - 'mu'*'ybar'/'mubar' 'if'
 // Did -gmm- request derivatives?
 if "derivatives" == "" {
 exit // no, so we are done
 }
 // Calculate derivatives
 // We need the panel means of x1*mu, x2*mu, and x3*mu
 tempvar work x1mubar x2mubar x3mubar
 generate double 'work' = x1*'mu' 'if'
 egen double 'x1mubar' = mean('work') 'if', by(id)
 replace 'work' = x2*'mu' 'if'
 egen double 'x2mubar' = mean('work') 'if', by(id)
 replace 'work' = x3*'mu' 'if'
 egen double 'x3mubar' = mean('work') 'if', by(id)
 local d1: word 1 of 'derivatives'
 local d2: word 2 of 'derivatives'
 local d3: word 3 of 'derivatives'
 replace 'd1' = -1*'mu'*'ybar'/'mubar'^2*(x1*'mubar' - 'x1mubar')
 replace 'd2' = -1*'mu'*'ybar'/'mubar'^2*(x2*'mubar' - 'x2mubar')
 replace 'd3' = -1*'mu'*'ybar'/'mubar'^2*(x3*'mubar' - 'x3mubar')
 }
end

```

The `derivatives()` option is made optional in the `syntax` statement by placing it in square brackets. If `gmm` needs to evaluate your moment equations but does not need derivatives at that time, then the `derivatives()` option will be empty. In our program, we check to see if that is the case, and, if so, `exit` without calculating derivatives. As is often the case with [R] `ml` as well, the portion of our program devoted to derivatives is longer than the code to compute the objective function.

The first part of our derivative code computes the term

$$\frac{1}{T} \sum_{l=1}^{l=T} x_{il}^{(j)} \mu_{il} \quad (29)$$

for  $x_{it}^{(j)} = x_1, x_2$ , and,  $x_3$ . The `'derivatives'` macro contains three variable names, corresponding to the three parameters of the `'at'` matrix. We extract those names into local macros `'d1'`, `'d2'`, and `'d3'`, and then fill in the variables those macros represent with the derivatives shown in (28).

With our program written, we fit our model by typing

```
. use http://www.stata-press.com/data/r11/poisson1
. gmm gmm_poideriv, nequations(1) parameters(b1 b2 b3)
> instruments(x1 x2 x3, noconstant) vce(cluster id) onestep hasderivatives
```

Step 1

```
Iteration 0: GMM criterion Q(b) = 51.99142
Iteration 1: GMM criterion Q(b) = .04345191
Iteration 2: GMM criterion Q(b) = 8.720e-06
Iteration 3: GMM criterion Q(b) = 7.115e-13
Iteration 4: GMM criterion Q(b) = 5.130e-27
```

GMM estimation

```
Number of parameters = 3
Number of moments = 3
Initial weight matrix: Unadjusted
```

Number of obs = 409

(Std. Err. adjusted for 45 clusters in id)

|     | Coef.     | Robust<br>Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|-----|-----------|---------------------|--------|-------|----------------------|-----------|
| /b1 | 1.94866   | .1000265            | 19.48  | 0.000 | 1.752612             | 2.144709  |
| /b2 | -2.966119 | .0923592            | -32.12 | 0.000 | -3.14714             | -2.785099 |
| /b3 | 1.008634  | .1156561            | 8.72   | 0.000 | .781952              | 1.235315  |

Instruments for equation 1: x1 x2 x3

Our results are identical to those in [example 9](#). Another way to verify that our program calculates derivatives correctly would be to type

```
. gmm gmm_poideriv, nequations(1) parameters(b1 b2 b3)
> instruments(x1 x2 x3, noconstant) vce(cluster id) onestep
```

Without the `hasderivatives` option, `gmm` will not request derivatives from your program, even if it contains code to compute them. If you have trouble obtaining convergence with the `hasderivatives` option but do not have trouble without that option, then you need to recheck your derivatives.

◀

In the [technical note](#) after example 9, we modified the `gmm_poi` program so that we could specify the name of our dependent variable and a list of regressors. Here is the analogous version that computes analytic derivatives:

*(Continued on next page)*

```

program gmm_poideriv2
 version 11
 syntax varlist if, at(name) myrhs(varlist) ///
 mylhs(varlist) myidvar(varlist) ///
 [derivatives(varlist)]

 quietly {
 tempvar mu mubar ybar
 gen double 'mu' = 0 'if'
 local j = 1
 foreach var of varlist 'myrhs' {
 replace 'mu' = 'mu' + 'var'*'at'[1,'j'] 'if'
 local j = 'j' + 1
 }
 replace 'mu' = exp('mu')
 egen double 'mubar' = mean('mu') 'if', by('myidvar')
 egen double 'ybar' = mean('mylhs') 'if', by('myidvar')
 replace 'varlist' = 'mylhs' - 'mu'*'ybar'/'mubar' 'if'
 if "'derivatives'" == "" {
 exit
 }
 tempvar work xmubar
 local j = 1
 foreach var of varlist 'myrhs' {
 generate double 'work' = 'var'*'mu' 'if'
 egen double 'xmubar' = mean('work') 'if' 'wt','exp', ///
 by('myidvar')
 local deriv : word 'j' of 'derivatives'
 replace 'deriv' = -1*'mu'*'ybar'/'mubar'^2* ///
 ('var'*'mubar' - 'xmubar')
 local '++j'
 drop 'work' 'xmubar'
 }
 }
end

```

To use this program, we type

```

gmm gmm_poideriv2, mylhs(y) myrhs(x1 x2 x3)
 nequations(1) parameters(b1 b2 b3)
 instruments(x1 x2 x3, noconstant)
 vce(cluster id) myidvar(id) onestep
 hasderivatives

```

We obtain results identical to those shown in [example 18](#).

Depending on your model, allowing your moment-evaluator program to accept weights may be as easy as modifying the `syntax` command to allow them, or it may require significantly more work. If your program uses only commands like `generate` and `replace`, then just modifying the `syntax` command is all you need to do; `gmm` takes care of applying the weights to the observation-level residuals when computing the sample moments, derivatives, and weight matrices. On the other hand, if your moment-evaluator program computes residuals using statistics that depend on multiple observations, then you must apply the weights passed to your program when computing those statistics.

In our examples of panel Poisson with strictly exogenous regressors, we used the statistics  $\bar{\mu}_i$  and  $\bar{y}_i$  when computing the residuals. If we are to allow weights with our moment-evaluator program, then we must incorporate those weights when computing  $\bar{\mu}_i$  and  $\bar{y}_i$ . Moreover, looking at the derivative in (28), the term highlighted in (29) is in fact a sample mean, so we must incorporate weights when computing it.

## ► Example 19: Panel Poisson with derivatives and weights

Here we modify the program in [example 18](#). to accept frequency weights. One complication immediately arises: we had been using `egen` to compute  $\bar{\mu}_i$  and  $\bar{y}_i$ . `egen` does not accept weights, so we must compute  $\bar{\mu}_i$  and  $\bar{y}_i$  ourselves, incorporating any weights the user may specify. Our program is

```

program gmm_poiderivfw
 version 11
 syntax varlist if [fweight/], at(name) [derivatives(varlist)]
 quietly {
 if "'exp'" == "" { // no weights
 local exp 1 // weight each observation equally
 }

 // Calculate residuals as before
 tempvar mu mubar ybar sumwt
 gen double 'mu' = exp(x1*'at'[1,1] + x2*'at'[1,2] + x3*'at'[1,3]) 'if'

 bysort id: gen double 'sumwt' = sum('exp')
 by id: gen double 'mubar' = sum('mu'*'exp')
 by id: gen double 'ybar' = sum(y*'exp')
 by id: replace 'mubar' = 'mubar'[_N] / 'sumwt'[_N]
 by id: replace 'ybar' = 'ybar'[_N] / 'sumwt'[_N]
 replace 'varlist' = y - 'mu'*'ybar'/'mubar' 'if'

 // Did -gmm- request derivatives?
 if "'derivatives'" == "" {
 exit // no, so we are done
 }

 // Calculate derivatives
 // We need the panel means of x1*mu, x2*mu, and x3*mu
 tempvar work x1mubar x2mubar x3mubar
 generate double 'work' = x1*'mu' 'if'
 by id: generate double 'x1mubar' = sum('work'*'exp')
 by id: replace 'x1mubar' = 'x1mubar'[_N] / 'sumwt'[_N]
 replace 'work' = x2*'mu' 'if'
 by id: generate double 'x2mubar' = sum('work'*'exp')
 by id: replace 'x2mubar' = 'x2mubar'[_N] / 'sumwt'[_N]
 replace 'work' = x3*'mu' 'if'
 by id: generate double 'x3mubar' = sum('work'*'exp')
 by id: replace 'x3mubar' = 'x3mubar'[_N] / 'sumwt'[_N]

 local d1: word 1 of 'derivatives'
 local d2: word 2 of 'derivatives'
 local d3: word 3 of 'derivatives'
 replace 'd1' = -1*'mu'*'ybar'/'mubar'^2*(x1*'mubar' - 'x1mubar')
 replace 'd2' = -1*'mu'*'ybar'/'mubar'^2*(x2*'mubar' - 'x2mubar')
 replace 'd3' = -1*'mu'*'ybar'/'mubar'^2*(x3*'mubar' - 'x3mubar')
 }
end

```

Our `syntax` command now indicates that `fweights` are allowed. The first part of our code looks at the macro `'exp'`. If it is empty, then the user did not specify weights in their call to `gmm`; and we set the macro equal to 1, so that we weight each observation equally. After we compute  $\mu_{it}$ , we calculate  $\bar{\mu}_i$  and  $\bar{y}_i$ , taking into account weights. To compute frequency-weighted means for each panel, we just multiply each observation by its respective weight, sum over all observations in the panel, then divide by the sum of the weights for the panel. (See [\[U\] 20.18 Weighted estimation](#) for information on how to handle `aweights` and `pweights`.) We use the same procedure to compute the frequency-weighted variant of expression (29) in the derivative calculations.

To use our program, we type

```
. use http://www.stata-press.com/data/r11/poissonwts
. gmm gmm_poiderivfw [fw=fwt], nequations(1) parameters(b1 b2 b3)
> instruments(x1 x2 x3, noconstant) vce(cluster id) onestep hasderivatives
(sum of wgt is 819)

Step 1
Iteration 0: GMM criterion Q(b) = 49.8292
Iteration 1: GMM criterion Q(b) = .11136736
Iteration 2: GMM criterion Q(b) = .00008519
Iteration 3: GMM criterion Q(b) = 7.110e-11
Iteration 4: GMM criterion Q(b) = 5.596e-23

GMM estimation
Number of parameters = 3
Number of moments = 3
Initial weight matrix: Unadjusted Number of obs = 819
 (Std. Err. adjusted for 45 clusters in id)
```

|     | Coef.     | Robust<br>Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|-----|-----------|---------------------|--------|-------|----------------------|-----------|
| /b1 | 1.967766  | .111795             | 17.60  | 0.000 | 1.748652             | 2.186881  |
| /b2 | -3.060838 | .0935561            | -32.72 | 0.000 | -3.244205            | -2.877472 |
| /b3 | 1.037594  | .1184227            | 8.76   | 0.000 | .80549               | 1.269698  |

```
Instruments for equation 1: x1 x2 x3
```

Testing whether our program works correctly with frequency weights is easy. A frequency-weighted dataset is just a compact form of a larger dataset in which identical observations are omitted and a frequency-weight variable is included to tell us how many times each observation in the smaller dataset appears in the larger dataset. Therefore, we can expand our smaller dataset by the frequency-weight variable and then refit our model without specifying frequency weights. If we obtain the same results, our program works correctly. When we type

```
. expand fw
(410 observations created)

. gmm gmm_poiderivfw, nequations(1) parameters(b1 b2 b3)
> instruments(x1 x2 x3, noconstant) vce(cluster id) onestep
```

we obtain the same results as before.





## Saved results

`gmm` saves the following in `e()`:

### Scalars

|                            |                                                                 |
|----------------------------|-----------------------------------------------------------------|
| <code>e(N)</code>          | number of observations                                          |
| <code>e(k)</code>          | number of parameters                                            |
| <code>e(k_eq)</code>       | number of moment equations                                      |
| <code>e(k_eq_model)</code> | number of equations in model Wald test                          |
| <code>e(k_aux)</code>      | number of auxiliary parameters                                  |
| <code>e(n_moments)</code>  | number of moments                                               |
| <code>e(n_eq)</code>       | number of equations in moment-evaluator program                 |
| <code>e(Q)</code>          | criterion function                                              |
| <code>e(J)</code>          | Hansen $J$ $\chi^2$ statistic                                   |
| <code>e(J_df)</code>       | $J$ statistic degrees of freedom                                |
| <code>e(k_i)</code>        | number of parameters in equation $i$                            |
| <code>e(has_xtinst)</code> | 1 if panel-style instruments specified, 0 otherwise             |
| <code>e(N_clust)</code>    | number of clusters                                              |
| <code>e(type)</code>       | 1 if interactive version, 2 if moment-evaluator program version |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                                       |
| <code>e(ic)</code>         | number of iterations used by iterative GMM estimator            |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                                     |

### Macros

|                                 |                                                                    |
|---------------------------------|--------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>gmm</code>                                                   |
| <code>e(cmdline)</code>         | command as typed                                                   |
| <code>e(title)</code>           | title specified in <code>title()</code>                            |
| <code>e(title_2)</code>         | title specified in <code>title2()</code>                           |
| <code>e(clustvar)</code>        | name of cluster variable                                           |
| <code>e(inst_i)</code>          | equation $i$ instruments                                           |
| <code>e(eqnames)</code>         | equation names                                                     |
| <code>e(winit)</code>           | initial weight matrix used                                         |
| <code>e(winitname)</code>       | name of user-supplied initial weight matrix                        |
| <code>e(estimator)</code>       | <code>onestep</code> , <code>twostep</code> , or <code>igmm</code> |
| <code>e(rhs)</code>             | variables specified in <code>variables()</code>                    |
| <code>e(params_i)</code>        | equation $i$ parameters                                            |
| <code>e(wmatrix)</code>         | <code>wmtype</code> specified in <code>wmatrix()</code>            |
| <code>e(vce)</code>             | <code>vcetype</code> specified in <code>vce()</code>               |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                      |
| <code>e(params)</code>          | parameter names                                                    |
| <code>e(sexp_i)</code>          | substitutable expression for equation $i$                          |
| <code>e(evalprog)</code>        | moment-evaluator program                                           |
| <code>e(evalopts)</code>        | options passed to moment-evaluator program                         |
| <code>e(nocommonesample)</code> | <code>nocommonesample</code> , if specified                        |
| <code>e(technique)</code>       | optimization technique                                             |
| <code>e(properties)</code>      | <code>b</code> <code>V</code>                                      |
| <code>e(estat_cmd)</code>       | program used to implement <code>estat</code>                       |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                     |
| <code>e(marginsnotok)</code>    | predictions disallowed by <code>margins</code>                     |

|                 |                                                                                |
|-----------------|--------------------------------------------------------------------------------|
| Matrices        |                                                                                |
| e(b)            | coefficient vector                                                             |
| e(init)         | initial values of the estimators                                               |
| e(Wuser)        | user-supplied initial weight matrix                                            |
| e(W)            | weight matrix used for final round of estimation                               |
| e(S)            | moment covariance matrix used in robust VCE computations                       |
| e(N_byequation) | number of observations per equation, if <code>nocommonesample</code> specified |
| e(V)            | variance–covariance matrix                                                     |
| e(V_modelbased) | model-based variance                                                           |
| Functions       |                                                                                |
| e(sample)       | marks estimation sample                                                        |

## Methods and formulas

`gmm` is implemented as an ado-file.

Let  $q$  denote the number of moment equations. For observation  $i$ ,  $i = 1, \dots, N$ , write the  $j$ th moment equation as  $\mathbf{z}_{ij}u_{ij}(\beta_j)$  for  $j = 1, \dots, q$ .  $\mathbf{z}_{ij}$  is a  $1 \times m_j$  vector, where  $m_j$  is the number of instruments specified for equation  $j$ . Let  $m = m_1 + \dots + m_q$ .

Our notation can incorporate moment conditions of the form  $h_{ij}(\mathbf{w}_{ij}; \beta_j)$  with instruments  $\mathbf{w}_{ij}$  by defining  $\mathbf{z}_{ij} = 1$  and  $u_{ij}(\beta_j) = h_{ij}(\mathbf{w}_{ij}; \beta_j)$ , so except when necessary we do not distinguish between the two types of moment conditions. We could instead use notation so that all our moment conditions are of the form  $h_{ij}(\mathbf{w}_{ij}; \beta_j)$ , or we could adopt notation that explicitly combines both forms of moment equations. However, because moment conditions of the form  $\mathbf{z}'_{ij}u_{ij}(\beta_j)$  are arguably more common, we use that notation.

Let  $\beta$  denote a  $k \times 1$  vector of parameters, consisting of all the unique parameters of  $\beta_1, \dots, \beta_q$ . Then we can stack the moment conditions and write them more compactly as  $\mathbf{Z}'_i\mathbf{u}_i(\beta)$ , where

$$\mathbf{Z}_i = \begin{bmatrix} \mathbf{z}_{i1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{z}_{i2} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{z}_{iq} \end{bmatrix} \quad \text{and} \quad \mathbf{u}_i(\beta) = \begin{bmatrix} u_{i1}(\beta_1) \\ u_{i2}(\beta_2) \\ \vdots \\ u_{iq}(\beta_q) \end{bmatrix}$$

The GMM estimator  $\hat{\beta}$  is the value of  $\beta$  that minimizes

$$Q(\beta) = \left\{ N^{-1} \sum_{i=1}^N \mathbf{z}'_i \mathbf{u}_i(\beta) \right\}' \mathbf{W} \left\{ N^{-1} \sum_{i=1}^N \mathbf{z}'_i \mathbf{u}_i(\beta) \right\} \tag{A1}$$

for  $q \times q$  weight matrix  $\mathbf{W}$ .

By default, `gmm` minimizes (A1) using the Gauss–Newton method. See Hayashi (2000, 498) for a derivation. This technique is typically faster than quasi-Newton methods and does not require second-order derivatives.

Methods and formulas are presented under the following headings:

- Initial weight matrix
- Weight matrix
- Variance–covariance matrix
- Hansen’s J statistic
- Panel-style instruments

## Initial weight matrix

If you specify `winitial(identity)`, then we set  $\mathbf{W} = \mathbf{I}_q$ .

If you specify `winitial(unadjusted)`, then we create matrix  $\mathbf{\Lambda}$  with typical submatrix

$$\mathbf{\Lambda}_{rs} = N^{-1} \sum_{i=1}^N \mathbf{z}'_{ir} \mathbf{z}_{is}$$

for  $r = 1, \dots, q$  and  $s = 1, \dots, q$ . If you include the `independent` suboption, then we set  $\mathbf{\Lambda}_{rs} = \mathbf{0}$  for  $r \neq s$ . The weight matrix  $\mathbf{W}$  equals  $\mathbf{\Lambda}^{-1}$ .

If you specify `winitial(matname)`, then we set  $\mathbf{W}$  equal to Stata matrix *matname*.

If you specify `winitial(xt xtspec)`, then you must specify one or two items in *xtspec*, one for each equation. `gmm` allows you to specify at most two moment equations when you specify `winitial(xt xtspec)`, one in first-differences and one in levels. We create the block-diagonal matrix  $\mathbf{H}$  with typical block  $\mathbf{H}_j$ . If the  $j$ th element of *xtspec* is “L”, then  $\mathbf{H}_j$  is the identity matrix of suitable dimension. If the  $j$ th element of *xtspec* is “D”, then

$$\mathbf{H}_j = \begin{bmatrix} 1 & -0.5 & 0 & \dots & 0 & 0 \\ -0.5 & 1 & -0.5 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & -0.5 \\ 0 & 0 & 0 & \dots & -0.5 & 1 \end{bmatrix}$$

Then

$$\mathbf{\Lambda}_H = N_g^{-1} \sum_{g=1}^{g=N_G} \mathbf{Z}'_g \mathbf{H} \mathbf{Z}_g$$

where  $g$  indexes panels in the dataset,  $N_G$  is the number of panels,  $\mathbf{Z}_g$  is the full instrument matrix for panel  $g$ , and  $\mathbf{W} = \mathbf{\Lambda}_H^{-1}$ . See [Panel-style instruments](#) below for a discussion of how  $\mathbf{Z}_g$  is formed.

## Weight matrix

Specification of the weight matrix applies only to the two-step and iterative estimators. When you use the `onestep` option, the `wmatrix()` option is ignored.

We first evaluate (A1) using the initial weight matrix described above and then compute  $\mathbf{u}_i(\hat{\beta})$ . In all cases,  $\mathbf{W} = \mathbf{\Lambda}^{-1}$ . If you specify `wmatrix(unadjusted)`, then we create  $\mathbf{\Lambda}$  to have typical submatrix

$$\mathbf{\Lambda}_{rs} = \sigma_{rs} N^{-1} \sum_{i=1}^N \mathbf{z}'_{ir} \mathbf{z}_{is}$$

where

$$\sigma_{rs} = N^{-1} \sum_{i=1}^N u_{ir}(\hat{\beta}) u_{is}(\hat{\beta})$$

and  $r$  and  $s$  index moment equations. For all types of weight matrices, if the `independent` suboption is specified, then  $\mathbf{\Lambda}_{rs} = \mathbf{0}$  for  $r \neq s$ , where  $\mathbf{\Lambda}_{rs}$  measures the covariance between moment conditions for equations  $r$  and  $s$ .

If you specify `wmatrix(robust)`, then

$$\mathbf{\Lambda} = N^{-1} \sum_{i=1}^N \mathbf{Z}_i \mathbf{u}_i(\hat{\beta}) \mathbf{u}_i'(\hat{\beta}) \mathbf{Z}_i'$$

If you specify `wmatrix(cluster clustvar)`, then

$$\mathbf{\Lambda} = N^{-1} \sum_{c=1}^{c=N_C} \mathbf{q}_c \mathbf{q}_c'$$

where  $c$  indexes clusters,  $N_C$  is the number of clusters, and

$$\mathbf{q}_c = \sum_{i \in c_j} \mathbf{Z}_i \mathbf{u}_i(\hat{\beta})$$

If you specify `wmatrix(hac kernel [#])`, then

$$\begin{aligned} \mathbf{\Lambda} = & N^{-1} \sum_{i=1}^N \mathbf{Z}_i \mathbf{u}_i(\hat{\beta}) \mathbf{u}_i(\hat{\beta})' \mathbf{Z}_i' + \\ & N^{-1} \sum_{l=1}^{l=n-1} \sum_{i=l+1}^N K(l, m) \left\{ \mathbf{Z}_i \mathbf{u}_i(\hat{\beta}) \mathbf{u}_{i-l}'(\hat{\beta}) \mathbf{Z}_{i-l}' + \mathbf{Z}_{i-l} \mathbf{u}_{i-l}(\hat{\beta}) \mathbf{u}_i'(\hat{\beta}) \mathbf{Z}_i' \right\} \end{aligned}$$

where  $m = \#$  if  $\#$  is specified and  $m = N - 2$  otherwise. Define  $z = l/(m + 1)$ . If *kernel* is `bartlett` or `nwest`, then

$$K(l, m) = \begin{cases} 1 - z & 0 \leq z \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

If *kernel* is `parzen` or `gallant`, then

$$K(l, m) = \begin{cases} 1 - 6z^2 + 6z^3 & 0 \leq z \leq 0.5 \\ 2(1 - z)^3 & 0.5 < z \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

If *kernel* is `quadraticspectral` or `andrews`, then

$$K(l, m) = \begin{cases} 1 & z = 0 \\ 3\{\sin(\theta)/\theta - \cos(\theta)\}/\theta^2 & \text{otherwise} \end{cases}$$

where  $\theta = 6\pi z/5$ .

If `wmatrix(hac kernel opt)` is specified, then `gmm` uses Newey and West's (1994) automatic lag-selection algorithm, which proceeds as follows. Define  $\mathbf{h}$  to be an  $m \times 1$  vector of ones. Note that this definition of  $\mathbf{h}$  is slightly different than the one used by `ivregress`. There, the element of  $\mathbf{h}$  corresponding to the constant term equals zero, effectively ignoring the effect of the constant in determining the optimal lag length. Here we include the effect of the constant term. Now define

$$\begin{aligned}
f_i &= \{\mathbf{Z}'_i \mathbf{u}_i(\beta)\}' \mathbf{h} \\
\hat{\sigma}_j &= N^{-1} \sum_{i=j+1}^N f_i f_{i-j} \quad j = 0, \dots, m^* \\
\hat{s}^{(q)} &= 2 \sum_{j=1}^{j=m^*} \hat{\sigma}_j j^q \\
\hat{s}^{(0)} &= \hat{\sigma}_0 + 2 \sum_{j=1}^{j=m^*} \hat{\sigma}_j \\
\hat{\gamma} &= c_\gamma \left\{ \left( \frac{\hat{s}^{(q)}}{\hat{s}^{(0)}} \right)^2 \right\}^{1/(2q+1)} \\
m &= \hat{\gamma} N^{1/(2q+1)}
\end{aligned}$$

where  $q$ ,  $m^*$ , and  $c_\gamma$  depend on the kernel specified:

| Kernel                    | $q$ | $m^*$                             | $c_\gamma$ |
|---------------------------|-----|-----------------------------------|------------|
| Bartlett/Newey–West       | 1   | $\text{int} \{20(T/100)^{2/9}\}$  | 1.1447     |
| Parzen/Gallant            | 2   | $\text{int} \{20(T/100)^{4/25}\}$ | 2.6614     |
| Quadratic spectral/Andres | 2   | $\text{int} \{20(T/100)^{2/25}\}$ | 1.3221     |

where  $\text{int}(x)$  denotes the integer obtained by truncating  $x$  toward zero. For the Bartlett and Parzen kernels, the optimal lag is  $\min\{\text{int}(m), m^*\}$ . For the quadratic spectral kernel, the optimal lag is  $\min\{m, m^*\}$ .

## Variance–covariance matrix

If you specify `vce(unadjusted)`, then the VCE matrix is computed as

$$\text{Var}(\hat{\beta}) = N^{-1} \left\{ \overline{\mathbf{G}}(\hat{\beta})' \mathbf{W} \overline{\mathbf{G}}(\hat{\beta}) \right\}^{-1} \quad (\text{A2})$$

where

$$\overline{\mathbf{G}}(\hat{\beta}) = N^{-1} \sum_{i=1}^N \mathbf{Z}'_i \left. \frac{\partial \mathbf{u}_i(\beta)}{\partial \beta'} \right|_{\beta=\hat{\beta}}$$

For the two-step and iterated estimators, we use the weight matrix  $\mathbf{W}$  that was used to compute the final-round estimate  $\hat{\beta}$ .

For the one-step estimator, how the unadjusted VCE is computed depends on the type of initial weight matrix requested and the form of the moment equations. If you specify two or more moment equations of the form  $h_{ij}(\mathbf{w}_{ij}; \beta_j)$ , then `gmm` issues a warning message and computes a heteroskedasticity-robust VCE because here the matrix  $\mathbf{Z}'\mathbf{Z}$  is necessarily singular; moreover, here you must use the identity matrix as the initial weight matrix. Otherwise, if you specify `winitial(identity)` or `winitial(unadjusted)`, then `gmm` first computes an unadjusted weight matrix based on  $\hat{\beta}$  before evaluating (A2). If you specify `winitial(matname)`, then (A2) is evaluated based on *matname*; the user is responsible for verifying that the VCE and other statistics so produced are appropriate.

All types of robust VCEs computed by `gmm` take the form

$$\text{Var}(\hat{\beta}) = N^{-1} \left\{ \overline{\mathbf{G}}(\hat{\beta})' \mathbf{W} \overline{\mathbf{G}}(\hat{\beta}) \right\}^{-1} \overline{\mathbf{G}}(\hat{\beta})' \mathbf{W} \mathbf{S} \mathbf{W} \overline{\mathbf{G}}(\hat{\beta}) \left\{ \overline{\mathbf{G}}(\hat{\beta})' \mathbf{W} \overline{\mathbf{G}}(\hat{\beta}) \right\}^{-1}$$

For the one-step estimator,  $\mathbf{W}$  represents the initial weight matrix requested using the `winitial()` option, and  $\mathbf{S}$  is computed based on the specification of the `vce()` option. The formulas for the  $\mathbf{S}$  matrix are identical to the ones that define the  $\mathbf{\Lambda}$  matrix in [Weight matrix](#) above, except that  $\mathbf{S}$  is computed after the moment equations are reevaluated using the final estimate of  $\hat{\beta}$ . For the two-step and iterated GMM estimators, computation of  $\mathbf{W}$  is controlled by the `wmatrix()` option based on the penultimate estimate of  $\hat{\beta}$ .

For details on computation of the VCE matrix with dynamic panel-data models, see [Panel-style instruments](#) below.

## Hansen's J statistic

Hansen's (1982)  $J$  test of overidentifying restrictions is  $J = N \times Q(\hat{\beta})$ .  $J \sim \chi^2(m - k)$ . If  $m < k$ , `gmm` issues an error message without estimating the parameters. If  $m = k$ , the model is just-identified and  $J$  is saved as missing (“.”). For the two-step and iterated GMM estimators, the  $J$  statistic is based on the last-computed weight matrix as determined by the `wmatrix()` option. For the one-step estimator, `gmm` recomputes a weight matrix as described in the second paragraph of [Variance–covariance matrix](#) above. To obtain Hansen's  $J$  statistic, you use `estat overid`; see [\[R\] gmm postestimation](#).

## Panel-style instruments

Here we discuss several issues that arise only when you specify panel-style instruments by using the `xtinstruments()` option. When you specify the `xtinstruments()` option, we can no longer consider the instruments for one observation in isolation; instead, we must consider the instrument matrix for an entire panel at once. In the following discussion, we let  $T$  denote the number of time periods in a panel. To accommodate unbalanced datasets, conceptually we simply use zeros as instruments and residuals for time periods that are missing in a panel.

We consider the case where you specify both an equation in levels and an equation in differences, yielding two residual equations. Let  $u_{pt}^L(\beta)$  denote the residual for the level equation for panel  $p$  in period  $t$ , and let  $u_{pt}^D(\beta)$  denote the residual for the corresponding difference equation. Now define the  $2T \times 1$  vector  $\mathbf{u}_p(\beta)$  as

$$\mathbf{u}_p(\beta) = [u_{p1}^L(\beta), u_{p2}^L(\beta), \dots, u_{pT}^L(\beta), u_{p2}^D(\beta), u_{p3}^D(\beta), \dots, u_{pT}^D(\beta)]$$

The  $T + 1$  element of  $\mathbf{u}_p$  is  $u_{p2}^D(\beta)$  since we lose the first observation of the difference equation because of differencing.

We write the moment conditions for the  $p$ th panel as  $\mathbf{Z}_p \mathbf{u}_p(\beta)$ . To see how  $\mathbf{Z}_p$  is defined, let  $\mathbf{w}_{pt}^L$  and  $\mathbf{w}_{pt}^D$  denote the vectors of panel-style instruments for the level and difference equations, respectively, and let time be denoted by  $t$ ; we discuss their dimensions momentarily. Also let  $\mathbf{x}_{pt}^L$  and  $\mathbf{x}_{pt}^D$  denote the vectors of instruments specified in `instruments()` for the level and difference equations at time  $t$ . Without loss of generality, for our discussion we assume that you specify the level equation first. Then  $\mathbf{Z}_p$  has the form

$$\mathbf{Z}_p = \begin{bmatrix} \mathbf{w}_1^L & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{w}_2^L & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{w}_T^L & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{x}_1^L & \mathbf{x}_2^L & \cdots & \mathbf{x}_T^L & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{w}_1^D & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{w}_2^D & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{w}_T^D \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{x}_1^D & \mathbf{x}_2^D & \cdots & \mathbf{x}_T^D \end{bmatrix} \quad (\text{A3})$$

To see how the  $\mathbf{w}$  vectors are formed, suppose you specify

```
xtinstruments(eq(1): d, lags(a/ b))
```

Then  $\mathbf{w}_t^L$  will be a  $(b - a + 1) \times 1$  vector consisting of  $d_{t-a}, \dots, d_{t-b}$ . If  $(t - a) \leq 0$ , then instead we set  $\mathbf{w}_t^L = \mathbf{0}$ . If  $(t - a) > 0$  but  $(t - b) \leq 0$ , then we create  $\mathbf{w}_t^L$  to consist of  $d_{t-a}, \dots, d_1$ . With this definition,  $(b - a + 1)$  defines the maximum number of lags of  $d$  used, but `gmm` will proceed with fewer lags if all  $(b - a + 1)$  lags are not available. If you specify two panel-style instruments, `d` and `e`, say, then  $\mathbf{w}_t^L$  will consist of  $d_{t-a}, \dots, d_{t-b}, e_{t-a}, \dots, e_{t-b}$ .  $\mathbf{w}_t^D$  is handled analogously.

The  $\mathbf{x}_t^L$  vectors are simply  $j \times 1$  vectors, where  $j$  is the number of regular instruments specified with the `instruments()` option; these vectors include a “1” unless you specify the `noconstant` suboption.

Looking carefully at (A3), you will notice that for dynamic panel-data models, moment conditions corresponding to the instruments  $\mathbf{x}_{pt}^L$  take the form

$$E \left[ \sum_{t=1}^{t=T} \mathbf{x}_{pt}^L u_{pt}^L(\beta) \right] = \mathbf{0}$$

and likewise for  $\mathbf{x}_{pt}^D$ . Instead of having separate moment conditions for each time period, there is one moment condition equal to the average of individual periods’ moments. See [Arellano and Bond \(1991, 280\)](#). To include separate moment conditions for each time period, instead of specifying, say,

```
instruments(1: x)
```

you could instead first generate a variable called `one` equal to unity for all observations and specify

```
xtinstruments(1: x one)
```

(Creating the variable `one` is necessary because a constant is not automatically included in variable lists specified in `xtinstruments()`.)

Unbalanced panels are essentially handled by including zeros rows and columns of  $\mathbf{Z}_p$  and  $\mathbf{u}_p(\beta)$  corresponding to missing time periods. However, the numbers of instruments and moment conditions reported by `gmm` do not reflect this trickery and instead reflect the numbers of instruments and moment conditions that are not manipulated in this way. Moreover, `gmm` includes code to work through these situations efficiently without actually having to fill in zeros.

When you specify `winitial(xt ...)`, the one-step unadjusted VCE is computed as

$$\text{Var}(\hat{\beta}) = \hat{\sigma}_1^2 \mathbf{\Lambda}_H$$

where  $\Lambda_H$  was defined previously,

$$\hat{\sigma}_1^2 = (N - k)^{-1} \sum_{p=1}^{p=P} \mathbf{u}_p^D(\hat{\beta})$$

and  $\mathbf{u}_p^D(\hat{\beta}) = [u_{p2}^D(\hat{\beta}), \dots, u_{pT}^D(\hat{\beta})]$ . Here we use  $(N - k)^{-1}$  instead of  $N^{-1}$  to match `xtddpd`.

## References

- Arellano, M., and S. Bond. 1991. Some tests of specification for panel data: Monte Carlo evidence and an application to employment equations. *Review of Economic Studies* 58: 277–297.
- Baum, C. F. 2006. *An Introduction to Modern Econometrics Using Stata*. College Station, TX: Stata Press.
- Blundell, R., R. Griffith, and F. Windmeijer. 2002. Individual effects and dynamics in count data models. *Journal of Econometrics* 108: 113–131.
- Cameron, A. C., and P. K. Trivedi. 1998. *Regression Analysis of Count Data*. Cambridge: Cambridge University Press.
- . 2005. *Microeconometrics: Methods and Applications*. New York: Cambridge University Press.
- . 2009. *Microeconometrics Using Stata*. College Station, TX: Stata Press.
- Chamberlain, G. 1992. Comment: Sequential moment restrictions in panel data. *Journal of Business and Economic Statistics* 10: 20–26.
- Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.
- . 2004. *Econometric Theory and Methods*. New York: Oxford University Press.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- Hall, A. R. 2005. *Generalized Method of Moments*. Oxford: Oxford University Press.
- Hamilton, J. D. 1994. *Time Series Analysis*. Princeton: Princeton University Press.
- Hansen, L. P. 1982. Large sample properties of generalized method of moments estimators. *Econometrica* 50: 1029–1054.
- Hansen, L. P., and K. J. Singleton. 1982. Generalized instrumental variables estimation of nonlinear rational expectations models. *Econometrica* 50: 1269–1286.
- Hayashi, F. 2000. *Econometrics*. Princeton, NJ: Princeton University Press.
- Manski, C. F. 1988. *Analog Estimation Methods in Econometrics*. New York: Chapman & Hall/CRC.
- Mátyás, L. 1999. *Generalized Method of Moments Estimation*. Cambridge: Cambridge University Press.
- Mullahy, J. 1997. Instrumental-variable estimation of count data models: Applications to models of cigarette smoking behavior. *Review of Economics and Statistics* 79: 586–593.
- Newey, W. K., and K. D. West. 1994. Automatic lag selection in covariance matrix estimation. *Review of Economic Studies* 61: 631–653.
- Nickell, S. 1981. Biases in dynamic models with fixed effects. *Econometrica* 49: 1417–1426.
- Ruud, P. A. 2000. *An Introduction to Classical Econometric Theory*. New York: Oxford University Press.
- Wilde, J. 2008. A note on GMM estimation of probit models with endogenous regressors. *Statistical Papers* 49: 471–484.
- Windmeijer, F. 2000. Moment conditions for fixed effects count data models with endogenous regressors. *Economics Letters* 68: 21–24.
- . 2005. A finite sample correction for the variance of linear efficient two-step GMM estimators. *Journal of Econometrics* 126: 25–51.
- Windmeijer, F., and J. M. C. Santos Silva. 1997. Endogeneity in count data models: An application to demand for health care. *Journal of Applied Econometrics* 12: 281–294.



Wooldridge, J. M. 1997. Multiplicative panel data models without the strict exogeneity assumption. *Econometric Theory* 13: 667–678.

———. 2002. *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA: MIT Press.

## Also see

[R] [gmm postestimation](#) — Postestimation tools for gmm

[R] [ivregress](#) — Single-equation instrumental-variables regression

[R] [ivprobit](#) — Probit model with continuous endogenous regressors

[R] [ivtobit](#) — Tobit model with continuous endogenous regressors

[R] [nl](#) — Nonlinear least-squares estimation

[R] [nlsur](#) — Estimation of nonlinear systems of equations

[R] [regress](#) — Linear regression

[XT] [xtdpd](#) — Linear dynamic panel-data estimation

## Description

The following postestimation command is of special interest after `gmm`:

| command                   | description                                  |
|---------------------------|----------------------------------------------|
| <code>estat overid</code> | perform test of overidentifying restrictions |

The following standard postestimation commands are also available:

| command                | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>     | VCE                                                                                                 |
| <code>estimates</code> | cataloging estimation results                                                                       |
| <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>   | residuals                                                                                           |
| <code>predictnl</code> | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details.

## Special-interest postestimation command

`estat overid` reports Hansen’s  $J$  statistic, which is used to determine the validity of the overidentifying restrictions in a GMM model. If the model is correctly specified in the sense that  $E\{\mathbf{z}_i u_i(\boldsymbol{\beta})\} = \mathbf{0}$ , then the sample analog to that condition should hold at the estimated value of  $\boldsymbol{\beta}$ . Hansen’s  $J$  statistic is valid only if the weight matrix is optimal, meaning that it equals the inverse of the covariance matrix of the moment conditions. Therefore, `estat overid` only reports Hansen’s  $J$  statistic after two-step or iterated estimation, or if you specified `winitial(matname)` when calling `gmm`. In the latter case, it is your responsibility to determine the validity of the  $J$  statistic.

## Syntax for predict

```

predict [type] newvar [if] [in] [, equation(#eqno|eqname)]

predict [type] { stub*|newvar1 ... newvarq } [if] [in]

```

Residuals are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

You specify one new variable and (optionally) `equation()`, or you specify `stub*` or  $q$  new variables, where  $q$  is the number of moment equations.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Option for predict

`equation(#eqno | eqname)` specifies the equation for which residuals are desired. Specifying `equation(#1)` indicates that the calculation is to be made for the first moment equation. Specifying `equation(demand)` would indicate that the calculation is to be made for the moment equation named `demand`, assuming there is an equation named `demand` in the model.

If you specify one new variable name and omit `equation()`, results are the same as if you had specified `equation(#1)`.

For more information on using `predict` after multiple-equation estimation commands, see [\[R\] predict](#).

## Syntax for estat overid

```
estat overid
```

## Menu

Statistics > Postestimation > Reports and statistics

## Remarks

As we noted in [Introduction](#) of [\[R\] gmm](#), underlying generalized method of moments (GMM) estimators is a set of  $l$  moment conditions,  $E\{\mathbf{z}_i u_i(\beta)\} = \mathbf{0}$ . When  $l$  is greater than the number of parameters,  $k$ , any size- $k$  subset of the moment conditions would yield a consistent parameter estimate. We remarked that the parameter estimates we would obtain would in general depend on which  $k$  moment conditions we used. However, if all our moment conditions are indeed valid, then the parameter estimates should not differ too much regardless of which  $k$  moment conditions we used to estimate the parameters. The test of overidentifying restrictions is a model specification test based on this observation. The test of overidentifying restrictions requires that the number of moment conditions be greater than the number of parameters in the model.

Recall that the GMM criterion function is

$$Q = \left\{ \frac{1}{N} \sum_i \mathbf{z}_i u_i(\beta) \right\}' \mathbf{W} \left\{ \frac{1}{N} \sum_i \mathbf{z}_i u_i(\beta) \right\}$$

The test of overidentifying restrictions is remarkably simple. If  $\mathbf{W}$  is an optimal weight matrix, under the null hypothesis  $H_0: E\{\mathbf{z}_i u_i(\beta)\} = \mathbf{0}$ , the test statistic  $J = N \times Q \sim \chi^2(l - k)$ . A large test statistic casts doubt on the null hypothesis.

For the test to be valid,  $\mathbf{W}$  must be optimal, meaning that  $\mathbf{W}$  must be the inverse of the covariance matrix of the moment conditions:

$$\mathbf{W}^{-1} = E\{\mathbf{z}_i u_i(\beta) u_i'(\beta) \mathbf{z}_i'\}$$

Therefore, `estat overid` works only after the two-step and iterated estimators, or if you supplied your own initial weight matrix by using the `winitial(matname)` option to `gmm` and used the one-step estimator.

Often the overidentifying restrictions test is interpreted as a test of the validity of the instruments  $\mathbf{z}$ . However, other forms of model misspecification can sometimes lead to a significant test statistic. See Hall (2005, sec. 5.1) for a discussion of the overidentifying restrictions test and its behavior in correctly and misspecified models.

## ► Example 1

In [example 6](#) of [\[R\] gmm](#), we fit an exponential regression model of the number of doctor visits based on the person's gender, income, possession of private health insurance, and presence of a chronic disease. We argued that the variable `income` may be endogenous; we used the person's age and race as additional instrumental variables. Here we refit the model and test the specification of the model. We type

```
. use http://www.stata-press.com/data/r11/docvisits
. gmm (docvis - exp({xb:private chronic female income} + {b0})),
> instruments(private chronic female age black hispanic)
(output omitted)
. estat overid
Test of overidentifying restriction:
Hansen's J chi2(2) = 9.52598 (p = 0.0085)
```

The  $J$  statistic is significant even at the 1% significance level, so we conclude that our model is misspecified. One possibility is that age and race directly affect the number of doctor visits, so we are not justified in excluding them from the model.

A simple technique to explore whether any of the instruments is invalid is to examine the statistics

$$r_j = \mathbf{W}_{jj}^{1/2} \left\{ \frac{1}{N} \sum_{i=1}^N z_{ij} u_i(\hat{\beta}) \right\}$$

for  $j = 1, \dots, k$ , where  $\mathbf{W}_{jj}$  denotes the  $j$ th diagonal element of  $\mathbf{W}$ ,  $u_i(\hat{\beta})$  denotes the sample residuals, and  $k$  is the number of instruments. If all the instruments are valid, then the scaled sample moments should at least be on the same order of magnitude. If one (or more) instrument's  $r_j$  is large in absolute value relative to the others, then that could be an indication that instrument is not valid.

In Stata, we type

```
. predict double r if e(sample) // obtain residual from the model
. matrix W = e(W) // retrieve weight matrix
. local i 1
. // loop over each instrument and compute r_j
. foreach var of varlist private chronic female age black hispanic {
2. generate double r`var' = r*`var'*sqrt(W[`i', `i'])
3. local ++i
4. }
```

. summarize r\*

| Variable  | Obs  | Mean      | Std. Dev. | Min       | Max      |
|-----------|------|-----------|-----------|-----------|----------|
| r         | 4412 | .0344373  | 8.26176   | -151.1847 | 113.059  |
| rprivate  | 4412 | .007988   | 3.824118  | -72.66254 | 54.33852 |
| rchronic  | 4412 | .0026947  | 2.0707    | -43.7311  | 32.703   |
| rfemale   | 4412 | .0028168  | 1.566397  | -12.7388  | 24.43621 |
| rage      | 4412 | .0360978  | 4.752986  | -89.74112 | 55.58143 |
| rblack    | 4412 | -.0379317 | 1.062027  | -24.39747 | 27.34512 |
| rhispanic | 4412 | -.017435  | 1.08567   | -5.509386 | 31.53512 |

We notice that the  $r_j$  statistics for `age`, `black`, and `hispanic` are larger than those for the other instruments in our model, supporting our suspicion that age and race may have a direct impact on the number of doctor visits.



Saved results

`estat overid` saves the following in `r()`:

|                      |                                  |
|----------------------|----------------------------------|
| Scalars              |                                  |
| <code>r(J)</code>    | Hansen's $J$ statistic           |
| <code>r(J_df)</code> | $J$ statistic degrees of freedom |
| <code>r(J_p)</code>  | $J$ statistic $p$ -value         |

Methods and formulas

All postestimation commands listed above are implemented as ado-files.

Reference

Hall, A. R. 2005. *Generalized Method of Moments*. Oxford: Oxford University Press.

Also see

- [R] [gmm](#) — Generalized method of moments estimation
- [U] [20 Estimation and postestimation commands](#)

Syntax

```
grmeanby varlist [if] [in] [weight], summarize(varname) [options]
```

| options                                                                                                | description                                                             |
|--------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------|
| Main                                                                                                   |                                                                         |
| * <u>s</u> ummarize( <i>varname</i> )                                                                  | graph mean (or median) of <i>varname</i>                                |
| <u>m</u> edian                                                                                         | graph medians; default is to graph means                                |
| Plot                                                                                                   |                                                                         |
| <i>c</i> line_options                                                                                  | change the look of the lines                                            |
| <i>m</i> arker_options                                                                                 | change look of markers (color, size, etc.)                              |
| <i>m</i> arker_label_options                                                                           | add marker labels; change look or position                              |
| Y axis, X axis, Titles, Legend, Overall                                                                |                                                                         |
| <i>t</i> woway_options                                                                                 | any options other than by() documented in<br>[G] <i>t</i> woway_options |
| *summarize( <i>varname</i> ) is required.<br>aweights and fweights are allowed; see [U] 11.1.6 weight. |                                                                         |

Menu

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Graph means/medians by groups

Description

grmeanby graphs the (optionally weighted) means or medians of *varname* according to the values of the variables in *varlist*. The variables in *varlist* may be string or numeric and, if numeric, may be labeled.

Options

|      |                                                                                                                                                                                                                   |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Main | <p>summarize(<i>varname</i>) is required; it specifies the name of the variable whose mean or median is to be graphed.</p> <p>median is optional; it specifies that the graph is to be of medians, not means.</p> |
| Plot | <p><i>cline_options</i> affect the rendition of the lines through the markers, including their color, pattern, and width; see [G] <i>cline_options</i>.</p>                                                       |

*marker\_options* affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [G] [marker\\_options](#).

*marker\_label\_options* specify if and how the markers are to be labeled; see [G] [marker\\_label\\_options](#).

Y axis, X axis, Titles, Legend, Overall

*twoway\_options* are any of the options documented in [G] [twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

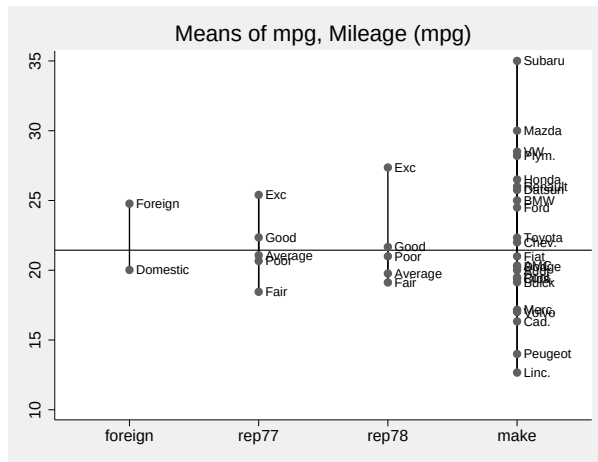
## Remarks

The idea of graphing means of categorical variables was shown in [Chambers and Hastie \(1992, 3\)](#). Because this was shown in the context of an S function for making such graphs, it doubtless has roots going back further than that. `grmeanby` is, in any case, another implementation of what we will assume is their idea.

### ► Example 1

Using a variation of our auto dataset, we graph the mean of `mpg` by `foreign`, `rep77`, `rep78`, and `make`:

```
. use http://www.stata-press.com/data/r11/auto1
(Automobile Models)
. grmeanby foreign rep77 rep78 make, sum(mpg)
```



If we had wanted a graph of medians rather than means, we could have typed

```
. grmeanby foreign rep77 rep78 make, sum(mpg) median
```

## Methods and formulas

`grmeanby` is implemented as an ado-file.

## References

- Chambers, J. M., and T. J. Hastie, ed. 1992. *Statistical Models in S*. Pacific Grove, CA: Wadsworth and Brooks/Cole.
- Gould, W. W. 1993. [gr12: Graphs of means and medians by categorical variables](#). *Stata Technical Bulletin* 12: 13.  
Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 44–45. College Station, TX: Stata Press.



Syntax

```
hausman name-consistent [name-efficient] [, options]
```

| options                               | description                                                                                   |
|---------------------------------------|-----------------------------------------------------------------------------------------------|
| Main                                  |                                                                                               |
| <u>constant</u>                       | include estimated intercepts in comparison; default is to exclude                             |
| <u>alleqs</u>                         | use all equations to perform test; default is first equation only                             |
| <u>skipeqs</u> ( <i>eqlist</i> )      | skip specified equations when performing test                                                 |
| <u>equations</u> ( <i>matchlist</i> ) | associate/compare the specified (by number) pairs of equations                                |
| <u>force</u>                          | force performance of test, even though assumptions are not met                                |
| <u>df</u> (#)                         | use # degrees of freedom                                                                      |
| <u>sigmamore</u>                      | base both (co)variance matrices on disturbance variance<br>estimate from efficient estimator  |
| <u>sigmaless</u>                      | base both (co)variance matrices on disturbance variance<br>estimate from consistent estimator |
| Advanced                              |                                                                                               |
| <u>tconsistent</u> ( <i>string</i> )  | consistent estimator column header                                                            |
| <u>tefficient</u> ( <i>string</i> )   | efficient estimator column header                                                             |

where *name-consistent* and *name-efficient* are names under which estimation results were saved via `estimates store`.

A period (.) may be used to refer to the last estimation results, even if these were not already stored. Not specifying *name-efficient* is equivalent to specifying the last estimation results as “.”.

Menu

Statistics > Postestimation > Tests > Hausman specification test

Description

hausman performs Hausman’s (1978) specification test.

Options

Main

`constant` specifies that the estimated intercept(s) be included in the model comparison; by default, they are excluded. The default behavior is appropriate for models in which the constant does not have a common interpretation across the two models.

`alleqs` specifies that all the equations in the models be used to perform the Hausman test; by default, only the first equation is used.

`skipeqs(eqlist)` specifies in *eqlist* the names of equations to be excluded from the test. Equation numbers are not allowed in this context, because the equation names, along with the variable names, are used to identify common coefficients.

`equations(matchlist)` specifies, by number, the pairs of equations that are to be compared.

The *matchlist* in `equations()` should follow the syntax

$$\#_c:\#_e \left[ , \#_c:\#_e \left[ , \dots \right] \right]$$

where  $\#_c$  ( $\#_e$ ) is an equation number of the always-consistent (efficient under  $H_0$ ) estimator. For instance, `equations(1:1)`, `equations(1:1, 2:2)`, or `equations(1:2)`.

If `equations()` is not specified, then equations are matched on equation names.

`equations()` handles the situation in which one estimator uses equation names and the other does not. For instance, `equations(1:2)` means that equation 1 of the always-consistent estimator is to be tested against equation 2 of the efficient estimator. `equations(1:1, 2:2)` means that equation 1 is to be tested against equation 1 and that equation 2 is to be tested against equation 2. If `equations()` is specified, the `alleqs` and `skipeqs` options are ignored.

`force` specifies that the Hausman test be performed, even though the assumptions of the Hausman test seem not to be met, for example, because the estimators were `pweighted` or the data were clustered.

`df(#)` specifies the degrees of freedom for the Hausman test. The default is the matrix rank of the variance of the difference between the coefficients of the two estimators.

`sigmamore` and `sigmaless` specify that the two covariance matrices used in the test be based on a common estimate of disturbance variance ( $\sigma^2$ ).

`sigmamore` specifies that the covariance matrices be based on the estimated disturbance variance from the efficient estimator. This option provides a proper estimate of the contrast variance for so-called tests of exogeneity and overidentification in instrumental-variables regression.

`sigmaless` specifies that the covariance matrices be based on the estimated disturbance variance from the consistent estimator.

These options can be specified only when both estimators save `e(sigma)` or `e(rmse)`, or with the `xtreg` command. `e(sigma_e)` is saved after the `xtreg` command with the option `fe` or `mle` option. `e(rmse)` is saved after the `xtreg` command with the `re` option.

`sigmamore` or `sigmaless` are recommended when comparing fixed-effects and random-effects linear regression because they are much less likely to produce a non-positive-definite-differenced covariance matrix (although the tests are asymptotically equivalent whether or not one of the options is specified).

#### Advanced

`tconsistent(string)` and `tefficient(string)` are formatting options. They allow you to specify the headers of the columns of coefficients that default to the names of the models. These options will be of interest primarily to programmers.

## Remarks

`hausman` is a general implementation of Hausman’s (1978) specification test, which compares an estimator  $\hat{\theta}_1$  that is known to be consistent with an estimator  $\hat{\theta}_2$  that is efficient under the assumption being tested. The null hypothesis is that the estimator  $\hat{\theta}_2$  is indeed an efficient (and consistent) estimator of the true parameters. If this is the case, there should be no systematic difference between the two estimators. If there exists a systematic difference in the estimates, you have reason to doubt the assumptions on which the efficient estimator is based.

The assumption of efficiency is violated if the estimator is `pweighted` or the data are clustered, so `hausman` cannot be used. The test can be forced by specifying the `force` option with `hausman`. For an alternative to using `hausman` in these cases, see [R] [suest](#).

To use `hausman`, you

```
. (compute the always-consistent estimator)
. estimates store name-consistent
. (compute the estimator that is efficient under H_0)
. hausman name-consistent .
```

Alternatively, you can turn this around:

```
. (compute the estimator that is efficient under H_0)
. estimates store name-efficient
. (fit the less-efficient model)
. (compute the always-consistent estimator)
. hausman . name-efficient
```

You can, of course, also compute and store both the always-consistent and efficient-under- $H_0$  estimators and perform the Hausman test with

```
. hausman name-consistent name-efficient
```

## ► Example 1

We are studying the factors that affect the wages of young women in the United States between 1970 and 1988, and we have a panel-data sample of individual women over that time span.

```
. use http://www.stata-press.com/data/r11/nlswork4
(National Longitudinal Survey. Young Women 14-26 years of age in 1968)
. describe
Contains data from http://www.stata-press.com/data/r11/nlswork4.dta
 obs: 28,534 National Longitudinal Survey.
 Young Women 14-26 years of age
 in 1968
 vars: 6 29 Jan 2009 16:35
 size: 485,078 (99.9% of memory free)
```

| variable name | storage<br>type | display<br>format | value<br>label | variable label               |
|---------------|-----------------|-------------------|----------------|------------------------------|
| idcode        | int             | %8.0g             |                | NLS ID                       |
| year          | byte            | %8.0g             |                | interview year               |
| age           | byte            | %8.0g             |                | age in current year          |
| msp           | byte            | %8.0g             |                | 1 if married, spouse present |
| ttl_exp       | float           | %9.0g             |                | total work experience        |
| ln_wage       | float           | %9.0g             |                | ln(wage/GNP deflator)        |

Sorted by: idcode year

We believe that a random-effects specification is appropriate for individual-level effects in our model. We fit a fixed-effects model that will capture all temporally constant individual-level effects.

```
. xtreg ln_wage age msp ttl_exp, fe
Fixed-effects (within) regression Number of obs = 28494
Group variable: idcode Number of groups = 4710
R-sq: within = 0.1373 Obs per group: min = 1
 between = 0.2571 avg = 6.0
 overall = 0.1800 max = 15
 F(3,23781) = 1262.01
corr(u_i, Xb) = 0.1476 Prob > F = 0.0000
```

| ln_wage | Coef.     | Std. Err.                         | t     | P> t  | [95% Conf. Interval] |           |
|---------|-----------|-----------------------------------|-------|-------|----------------------|-----------|
| age     | -.005485  | .000837                           | -6.55 | 0.000 | -.0071256            | -.0038443 |
| msp     | .0033427  | .0054868                          | 0.61  | 0.542 | -.0074118            | .0140971  |
| ttl_exp | .0383604  | .0012416                          | 30.90 | 0.000 | .0359268             | .0407941  |
| _cons   | 1.593953  | .0177538                          | 89.78 | 0.000 | 1.559154             | 1.628752  |
| sigma_u | .37674223 |                                   |       |       |                      |           |
| sigma_e | .29751014 |                                   |       |       |                      |           |
| rho     | .61591044 | (fraction of variance due to u_i) |       |       |                      |           |

```
F test that all u_i=0: F(4709, 23781) = 7.76 Prob > F = 0.0000
```

We assume that this model is consistent for the true parameters and save the results by using `estimates store` under a name, `fixed`:

```
. estimates store fixed
```

Now we fit a random-effects model as a fully efficient specification of the individual effects under the assumption that they are random and follow a normal distribution. We then compare these estimates with the previously saved results by using the `hausman` command.

```
. xtreg ln_wage age msp ttl_exp, re
Random-effects GLS regression Number of obs = 28494
Group variable: idcode Number of groups = 4710
R-sq: within = 0.1373 Obs per group: min = 1
 between = 0.2552 avg = 6.0
 overall = 0.1797 max = 15
Random effects u_i ~ Gaussian Wald chi2(3) = 5100.33
corr(u_i, X) = 0 (assumed) Prob > chi2 = 0.0000
```

| ln_wage | Coef.     | Std. Err.                         | z      | P> z  | [95% Conf. Interval] |           |
|---------|-----------|-----------------------------------|--------|-------|----------------------|-----------|
| age     | -.0069749 | .0006882                          | -10.13 | 0.000 | -.0083238            | -.0056259 |
| msp     | .0046594  | .0051012                          | 0.91   | 0.361 | -.0053387            | .0146575  |
| ttl_exp | .0429635  | .0010169                          | 42.25  | 0.000 | .0409704             | .0449567  |
| _cons   | 1.609916  | .0159176                          | 101.14 | 0.000 | 1.578718             | 1.641114  |
| sigma_u | .32648519 |                                   |        |       |                      |           |
| sigma_e | .29751014 |                                   |        |       |                      |           |
| rho     | .54633481 | (fraction of variance due to u_i) |        |       |                      |           |

```
. hausman fixed ., sigmamore
```

|         | Coefficients |           | (b-B)<br>Difference | sqrt(diag(V_b-V_B))<br>S.E. |
|---------|--------------|-----------|---------------------|-----------------------------|
|         | (b)<br>fixed | (B)<br>.  |                     |                             |
| age     | -.005485     | -.0069749 | .0014899            | .0004803                    |
| msp     | .0033427     | .0046594  | -.0013167           | .0020596                    |
| t11_exp | .0383604     | .0429635  | -.0046031           | .0007181                    |

b = consistent under Ho and Ha; obtained from xtreg

B = inconsistent under Ha, efficient under Ho; obtained from xtreg

Test: Ho: difference in coefficients not systematic

$\chi^2(3) = (b-B)'[(V_b-V_B)^{-1}](b-B)$

= 260.40

Prob>chi2 = 0.0000

Under the current specification, our initial hypothesis that the individual-level effects are adequately modeled by a random-effects model is resoundingly rejected. This result is based on the rest of our model specification, and random effects might be appropriate for some alternate model of wages.

◀

Jerry Allen Hausman was born in West Virginia in 1946. He studied economics at Brown and Oxford, has been at MIT since 1972, and has made many outstanding contributions to econometrics and applied microeconomics.

## ► Example 2

A stringent assumption of multinomial and conditional logit models is that outcome categories for the model have the property of independence of irrelevant alternatives (IIA). Stated simply, this assumption requires that the inclusion or exclusion of categories does not affect the relative risks associated with the regressors in the remaining categories.

One classic example of a situation in which this assumption would be violated involves the choice of transportation mode; see [McFadden \(1974\)](#). For simplicity, postulate a transportation model with the four possible outcomes: rides a train to work, takes a bus to work, drives the Ford to work, and drives the Chevrolet to work. Clearly, “drives the Ford” is a closer substitute to “drives the Chevrolet” than it is to “rides a train” (at least for most people). This means that excluding “drives the Ford” from the model could be expected to affect the relative risks of the remaining options and that the model would not obey the IIA assumption.

Using the data presented in [\[R\] mlogit](#), we will use a simplified model to test for IIA. The choice of insurance type among indemnity, prepaid, and uninsured is modeled as a function of age and gender. The indemnity category is allowed to be the base category, and the model including all three outcomes is fit. The results are then stored under the name `allcats`.

(Continued on next page)

```
. use http://www.stata-press.com/data/r11/sysdsn3
(Health insurance data)

. mlogit insure age male

Iteration 0: log likelihood = -555.85446
Iteration 1: log likelihood = -551.32973
Iteration 2: log likelihood = -551.32802
Iteration 3: log likelihood = -551.32802

Multinomial logistic regression Number of obs = 615
 LR chi2(4) = 9.05
 Prob > chi2 = 0.0598
Log likelihood = -551.32802 Pseudo R2 = 0.0081
```

| insure    | Coef.          | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-----------|----------------|-----------|-------|-------|----------------------|-----------|
| Indemnity | (base outcome) |           |       |       |                      |           |
| Prepaid   |                |           |       |       |                      |           |
| age       | -.0100251      | .0060181  | -1.67 | 0.096 | -.0218204            | .0017702  |
| male      | .5095747       | .1977893  | 2.58  | 0.010 | .1219147             | .8972346  |
| _cons     | .2633838       | .2787575  | 0.94  | 0.345 | -.2829708            | .8097383  |
| Uninsure  |                |           |       |       |                      |           |
| age       | -.0051925      | .0113821  | -0.46 | 0.648 | -.0275011            | .0171161  |
| male      | .4748547       | .3618462  | 1.31  | 0.189 | -.2343508            | 1.18406   |
| _cons     | -1.756843      | .5309602  | -3.31 | 0.001 | -2.797506            | -.7161803 |

```
. estimates store allcats
```

Under the IIA assumption, we would expect no systematic change in the coefficients if we excluded one of the outcomes from the model. (For an extensive discussion, see [Hausman and McFadden \[1984\]](#).) We reestimate the parameters, excluding the uninsured outcome, and perform a Hausman test against the fully efficient full model.

```
. mlogit insure age male if insure != "Uninsure":insure

Iteration 0: log likelihood = -394.8693
Iteration 1: log likelihood = -390.4871
Iteration 2: log likelihood = -390.48643
Iteration 3: log likelihood = -390.48643

Multinomial logistic regression Number of obs = 570
 LR chi2(2) = 8.77
 Prob > chi2 = 0.0125
Log likelihood = -390.48643 Pseudo R2 = 0.0111
```

| insure    | Coef.          | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-----------|----------------|-----------|-------|-------|----------------------|----------|
| Indemnity | (base outcome) |           |       |       |                      |          |
| Prepaid   |                |           |       |       |                      |          |
| age       | -.0101521      | .0060049  | -1.69 | 0.091 | -.0219214            | .0016173 |
| male      | .5144003       | .1981735  | 2.60  | 0.009 | .1259874             | .9028133 |
| _cons     | .2678043       | .2775563  | 0.96  | 0.335 | -.276196             | .8118046 |

```
. hausman . allcats, alleqs constant
```

|       | Coefficients |           | (b-B)<br>Difference | sqrt(diag(V_b-V_B))<br>S.E. |
|-------|--------------|-----------|---------------------|-----------------------------|
|       | (b)          | (B)       |                     |                             |
|       | .            | allcats   |                     |                             |
| age   | -.0101521    | -.0100251 | -.0001269           | .                           |
| male  | .5144003     | .5095747  | .0048256            | .0123338                    |
| _cons | .2678043     | .2633838  | .0044205            | .                           |

b = consistent under Ho and Ha; obtained from mlogit

B = inconsistent under Ha, efficient under Ho; obtained from mlogit

Test: Ho: difference in coefficients not systematic

$\chi^2(3) = (b-B)'[(V_b-V_B)^{-1}](b-B)$

= 0.08

Prob>chi2 = 0.9944

(V\_b-V\_B is not positive definite)

The syntax of the if condition on the mlogit command simply identified the "Uninsured" category with the insure value label; see [U] 12.6.3 Value labels. On examining the output from hausman, we see that there is no evidence that the IIA assumption has been violated.

Because the Hausman test is a standardized comparison of model coefficients, using it with mlogit requires that the base outcome be the same in both competing models. In particular, if the most-frequent category (the default base outcome) is being removed to test for IIA, you must use the baseoutcome() option in mlogit to manually set the base outcome to something else. Or you can use the equation() option of the hausman command to align the equations of the two models.

Having the missing values for the square root of the diagonal of the covariance matrix of the differences is not comforting, but it is also not surprising. This covariance matrix is guaranteed to be positive definite only asymptotically (it is a consequence of the assumption that one of the estimators is efficient), and assurances are not made about the diagonal elements. Negative values along the diagonal are possible, and the fourth column of the table is provided mainly for descriptive use.

We can also perform the Hausman IIA test against the remaining alternative in the model:

```
. mlogit insure age male if insure != "Prepaid":insure
```

```
Iteration 0: log likelihood = -132.59913
```

```
Iteration 1: log likelihood = -131.78009
```

```
Iteration 2: log likelihood = -131.76808
```

```
Iteration 3: log likelihood = -131.76807
```

```
Multinomial logistic regression
```

```
Number of obs = 338
```

```
LR chi2(2) = 1.66
```

```
Prob > chi2 = 0.4356
```

```
Pseudo R2 = 0.0063
```

```
Log likelihood = -131.76807
```

| insure    | Coef.          | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-----------|----------------|-----------|-------|-------|----------------------|-----------|
| Indemnity | (base outcome) |           |       |       |                      |           |
| Uninsure  |                |           |       |       |                      |           |
| age       | -.0041055      | .0115807  | -0.35 | 0.723 | -.0268033            | .0185923  |
| male      | .4591074       | .3595663  | 1.28  | 0.202 | -.2456296            | 1.163844  |
| _cons     | -1.801774      | .5474476  | -3.29 | 0.001 | -2.874752            | -.7287968 |

```
. hausman . allcats, alleqs constant
```

|       | Coefficients |           |            |                     |
|-------|--------------|-----------|------------|---------------------|
|       | (b)          | (B)       | (b-B)      | sqrt(diag(V_b-V_B)) |
|       | .            | allcats   | Difference | S.E.                |
| age   | -.0041055    | -.0051925 | .001087    | .0021355            |
| male  | .4591074     | .4748547  | -.0157473  | .                   |
| _cons | -1.801774    | -1.756843 | -.0449311  | .1333421            |

```

 b = consistent under Ho and Ha; obtained from mlogit
 B = inconsistent under Ha, efficient under Ho; obtained from mlogit
Test: Ho: difference in coefficients not systematic
 chi2(3) = (b-B)'[(V_b-V_B)^(-1)](b-B)
 = -0.18 chi2<0 ==> model fitted on these
 data fails to meet the asymptotic
 assumptions of the Hausman test;
 see suest for a generalized test
```

Here the  $\chi^2$  statistic is actually negative. We might interpret this result as strong evidence that we cannot reject the null hypothesis. Such a result is not an unusual outcome for the Hausman test, particularly when the sample is relatively small—there are only 45 uninsured individuals in this dataset.

Are we surprised by the results of the Hausman test in this example? Not really. Judging from the  $z$  statistics on the original multinomial logit model, we were struggling to identify any structure in the data with the current specification. Even when we were willing to assume IIA and computed the efficient estimator under this assumption, few of the effects could be identified as statistically different from those on the base category. Trying to base a Hausman test on a contrast (difference) between two poor estimates is just asking too much of the existing data.



In [example 2](#), we encountered a case in which the Hausman was not well defined. Unfortunately, in our experience this happens fairly often. Stata provides an alternative to the Hausman test that overcomes this problem through an alternative estimator of the variance of the difference between the two estimators. This other estimator is guaranteed to be positive semidefinite. This alternative estimator also allows a widening of the scope of problems to which Hausman-type tests can be applied by relaxing the assumption that one of the estimators is efficient. For instance, you can perform Hausman-type tests to clustered observations and survey estimators. See [\[R\] suest](#) for details.

## Saved results

hausman saves the following in `r()`:

```
Scalars
 r(chi2) χ^2
 r(df) degrees of freedom for the statistic
 r(p) p -value for the χ^2
 r(rank) rank of (V_b-V_B)^(-1)
```

## Methods and formulas

hausman is implemented as an ado-file.



The Hausman statistic is distributed as  $\chi^2$  and is computed as

$$H = (\beta_c - \beta_e)'(V_c - V_e)^{-1}(\beta_c - \beta_e)$$

where

|           |                                                         |
|-----------|---------------------------------------------------------|
| $\beta_c$ | is the coefficient vector from the consistent estimator |
| $\beta_e$ | is the coefficient vector from the efficient estimator  |
| $V_c$     | is the covariance matrix of the consistent estimator    |
| $V_e$     | is the covariance matrix of the efficient estimator     |

When the difference in the variance matrices is not positive definite, a Moore–Penrose generalized inverse is used. As noted in [Gourieroux and Monfort \(1995, 125–128\)](#), the choice of generalized inverse is not important asymptotically.

The number of degrees of freedom for the statistic is the rank of the difference in the variance matrices. When the difference is positive definite, this is the number of common coefficients in the models being compared.

## Acknowledgment

Portions of `hausman` are based on an early implementation by Jeroen Weesie, Utrecht University, The Netherlands.

## References

- Baltagi, B. H. 2008. *Econometrics*. 4th ed. Berlin: Springer.
- Gourieroux, C., and A. Monfort. 1995. *Statistics and Econometric Models, Vol 2: Testing, Confidence Regions, Model Selection, and Asymptotic Theory*. Trans. Q. Vuong. Cambridge: Cambridge University Press.
- Hausman, J. A. 1978. Specification tests in econometrics. *Econometrica* 46: 1251–1271.
- Hausman, J. A., and D. L. McFadden. 1984. Specification tests for the multinomial logit model. *Econometrica* 52: 1219–1240.
- McFadden, D. L. 1974. Measurement of urban travel demand. *Journal of Public Economics* 3: 303–328.

## Also see

- [\[R\] `lrtest`](#) — Likelihood-ratio test after estimation
- [\[R\] `suest`](#) — Seemingly unrelated estimation
- [\[R\] `test`](#) — Test linear hypotheses after estimation
- [\[XT\] `xtreg`](#) — Fixed-, between-, and random-effects, and population-averaged linear models

## Syntax

### Basic syntax

```
heckman depvar [indepvars], select(varlists) [twostep]
```

or

```
heckman depvar [indepvars], select(depvars = varlists) [twostep]
```

### Full syntax for maximum likelihood estimates only

```
heckman depvar [indepvars] [if] [in] [weight],
 select([depvars =] varlists [, offset(varname) noconstant])
 [heckman_ml_options]
```

### Full syntax for Heckman's two-step consistent estimates only

```
heckman depvar [indepvars] [if] [in], twostep
 select([depvars =] varlists [, noconstant]) [heckman_ts_options]
```

| heckman_ml_options               | description                                                                                                        |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------|
| Model                            |                                                                                                                    |
| * <u>select</u> ()               | specify selection equation: dependent and independent variables; whether to have constant term and offset variable |
| <u>noconstant</u>                | suppress constant term                                                                                             |
| <u>offset</u> (varname)          | include varname in model with coefficient constrained to 1                                                         |
| <u>constraints</u> (constraints) | apply specified linear constraints                                                                                 |
| <u>collinear</u>                 | keep collinear variables                                                                                           |
| SE/Robust                        |                                                                                                                    |
| <u>vce</u> (vcetype)             | vcetype may be oim, <u>robust</u> , <u>cluster</u> clustvar, opg, <u>bootstrap</u> , or <u>jackknife</u>           |
| Reporting                        |                                                                                                                    |
| <u>level</u> (#)                 | set confidence level; default is level(95)                                                                         |
| <u>first</u>                     | report first-step probit estimates                                                                                 |
| <u>noskip</u>                    | perform likelihood-ratio test                                                                                      |
| <u>nshazard</u> (newvar)         | generate nonselection hazard variable                                                                              |
| <u>mills</u> (newvar)            | synonym for nshazard()                                                                                             |
| <u>nocnsreport</u>               | do not display constraints                                                                                         |
| <u>display_options</u>           | control spacing and display of omitted variables and base and empty cells                                          |

## Maximization

|                         |                                                           |
|-------------------------|-----------------------------------------------------------|
| <i>maximize_options</i> | control the maximization process; seldom used             |
| † <i>coeflegend</i>     | display coefficients' legend instead of coefficient table |

\* *select()* is required.

The full specification is *select*( [ *depvar<sub>s</sub>* = ] *varlist<sub>s</sub>* [ , *offset*(*varname*) *noconstant* ] ) .

† *coeflegend* does not appear in the dialog box.

|                           |             |
|---------------------------|-------------|
| <i>heckman_ts_options</i> | description |
|---------------------------|-------------|

## Model

|                   |                                                                                                |
|-------------------|------------------------------------------------------------------------------------------------|
| * <i>select()</i> | specify selection equation: dependent and independent variables; whether to have constant term |
| * <i>twostep</i>  | produce two-step consistent estimate                                                           |
| <i>noconstant</i> | suppress constant term                                                                         |
| <i>rhosigma</i>   | truncate $\rho$ to $[-1, 1]$ with consistent $\sigma$                                          |
| <i>hotrunc</i>    | truncate $\rho$ to $[-1, 1]$                                                                   |
| <i>rholimited</i> | truncate $\rho$ in limited cases                                                               |
| <i>rhoforce</i>   | do not truncate $\rho$                                                                         |

## SE

|                               |                                                                                    |
|-------------------------------|------------------------------------------------------------------------------------|
| <i>vce</i> ( <i>vcetype</i> ) | <i>vcetype</i> may be <i>conventional</i> , <i>bootstrap</i> , or <i>jackknife</i> |
|-------------------------------|------------------------------------------------------------------------------------|

## Reporting

|                                   |                                                                           |
|-----------------------------------|---------------------------------------------------------------------------|
| <i>level</i> (#)                  | set confidence level; default is <i>level</i> (95)                        |
| <i>first</i>                      | report first-step probit estimates                                        |
| <i>nshazard</i> ( <i>newvar</i> ) | generate nonselection hazard variable                                     |
| <i>mills</i> ( <i>newvar</i> )    | synonym for <i>nshazard</i> ()                                            |
| <i>display_options</i>            | control spacing and display of omitted variables and base and empty cells |

|                     |                                                           |
|---------------------|-----------------------------------------------------------|
| † <i>coeflegend</i> | display coefficients' legend instead of coefficient table |
|---------------------|-----------------------------------------------------------|

\* *select()* and *twostep* are required.

The full specification is *select*( [ *depvar<sub>s</sub>* = ] *varlist<sub>s</sub>* [ , *noconstant* ] )

† *coeflegend* does not appear in the dialog box.

*indepvars* and *varlist<sub>s</sub>* may contain factor variables; see [U] 11.4.3 **Factor variables**.

*depvar*, *indepvars*, *varlist<sub>s</sub>*, and *depvar<sub>s</sub>* may contain time-series operators; see [U] 11.4.4 **Time-series varlists**.

*bootstrap*, *by*, *jackknife*, *rolling*, *statsby*, and *svy* are allowed; see [U] 11.1.10 **Prefix commands**.

Weights are not allowed with the *bootstrap* prefix.

*aweight*s are not allowed with the *jackknife* prefix.

*twostep*, *vce*(), *first*, *noskip*, and weights are not allowed with the *svy* prefix.

*pweight*s, *aweight*s, *fweight*s, and *iweight*s are allowed with maximum likelihood estimation;

see [U] 11.1.6 **weight**. No weights are allowed if *twostep* is specified.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

## Menu

### heckman for maximum likelihood estimates

Statistics > Sample-selection models > Heckman selection model (ML)

### heckman for two-step consistent estimates

Statistics > Sample-selection models > Heckman selection model (two-step)

## Description

`heckman` fits regression models with selection by using either Heckman's two-step consistent estimator or full maximum likelihood.

## Options for Heckman selection model (ML)

### Model

`select(...)` specifies the variables and options for the selection equation. It is an integral part of specifying a Heckman model and is required. The selection equation should contain at least one variable that is not in the outcome equation.

If `depvars` is specified, it should be coded as 0 or 1, with 0 indicating an observation not selected and 1 indicating a selected observation. If `depvars` is not specified, observations for which `depvar` is not missing are assumed selected, and those for which `depvar` is missing are assumed not selected.

`noconstant`, `offset(varname)`, `constraints(constraints)`, `collinear`; see [\[R\] estimation options](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [\[R\] vce\\_option](#).

### Reporting

`level(#)`; see [\[R\] estimation options](#).

`first` specifies that the first-step probit estimates of the selection equation be displayed before estimation.

`noskip` specifies that a full maximum-likelihood model with only a constant for the regression equation be fit. This model is not displayed but is used as the base model to compute a likelihood-ratio test for the model test statistic displayed in the estimation header. By default, the overall model test statistic is an asymptotically equivalent Wald test that all the parameters in the regression equation are zero (except the constant). For many models, this option can substantially increase estimation time.

`nshazard(newvar)` and `mills(newvar)` are synonyms; either will create a new variable containing the nonselection hazard—what Heckman (1979) referred to as the inverse of the Mills' ratio—from the selection equation. The nonselection hazard is computed from the estimated parameters of the selection equation.

`nocnsreport`; see [\[R\] estimation options](#).

*display\_options*: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

#### Maximization

*maximize\_options*: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [R] [maximize](#). These options are seldom used.

Setting the optimization type to `technique(bhhh)` resets the default *vcetype* to `vce(opg)`.

The following option is available with `heckman` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Options for Heckman selection model (two-step)

#### Model

`select(...)` specifies the variables and options for the selection equation. It is an integral part of specifying a Heckman model and is required. The selection equation should contain at least one variable that is not in the outcome equation.

If *depvar<sub>s</sub>* is specified, it should be coded as 0 or 1, with 0 indicating an observation not selected and 1 indicating a selected observation. If *depvar<sub>s</sub>* is not specified, observations for which *depvar* is not missing are assumed selected, and those for which *depvar* is missing are assumed not selected.

`twostep` specifies that Heckman's (1979) two-step efficient estimates of the parameters, standard errors, and covariance matrix be produced.

`noconstant`; see [R] [estimation options](#).

`rhosigma`, `rot trunc`, `rho limited`, and `rho force` are rarely used options to specify how the two-step estimator (option `twostep`) handles unusual cases in which the two-step estimate of  $\rho$  is outside the admissible range for a correlation,  $[-1, 1]$ . When  $\text{abs}(\rho) > 1$ , the two-step estimate of the coefficient variance–covariance matrix may not be positive definite and thus may be unusable for testing. The default is `rhosigma`.

`rhosigma` specifies that  $\rho$  be truncated, as with the `rot trunc` option, and that the estimate of  $\sigma$  be made consistent with  $\hat{\rho}$ , the truncated estimate of  $\rho$ . So,  $\hat{\sigma} = \beta_m \hat{\rho}$ ; see [Methods and formulas](#) for the definition of  $\beta_m$ . Both the truncated  $\rho$  and the new estimate of  $\hat{\sigma}$  are used in all computations to estimate the two-step covariance matrix.

`rot trunc` specifies that  $\rho$  be truncated to lie in the range  $[-1, 1]$ . If the two-step estimate is less than  $-1$ ,  $\rho$  is set to  $-1$ ; if the two-step estimate is greater than  $1$ ,  $\rho$  is set to  $1$ . This truncated value of  $\rho$  is used in all computations to estimate the two-step covariance matrix.

`rho limited` specifies that  $\rho$  be truncated only in computing the diagonal matrix **D** as it enters **V**<sub>twostep</sub> and **Q**; see [Methods and formulas](#). In all other computations, the untruncated estimate of  $\rho$  is used.

`rho force` specifies that the two-step estimate of  $\rho$  be retained, even if it is outside the admissible range for a correlation. This option may, in rare cases, lead to a non-positive-definite covariance matrix.

These options have no effect when estimation is by maximum likelihood, the default. They also have no effect when the two-step estimate of  $\rho$  is in the range  $[-1, 1]$ .

SE

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

`vce(conventional)`, the default, uses the two-step variance estimator derived by Heckman.

Reporting

`level(#)`; see [R] [estimation options](#).

`first` specifies that the first-step probit estimates of the selection equation be displayed before estimation.

`nshazard(newvar)` and `mills(newvar)` are synonyms; either will create a new variable containing the nonselection hazard—what Heckman (1979) referred to as the inverse of the Mills’ ratio—from the selection equation. The nonselection hazard is computed from the estimated parameters of the selection equation.

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

The following option is available with `heckman` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

Remarks

The Heckman selection model (Gronau 1974; Lewis 1974; Heckman 1976) assumes that there exists an underlying regression relationship,

$$y_j = \mathbf{x}_j\boldsymbol{\beta} + u_{1j} \qquad \text{regression equation}$$

The dependent variable, however, is not always observed. Rather, the dependent variable for observation  $j$  is observed if

$$\mathbf{z}_j\boldsymbol{\gamma} + u_{2j} > 0 \qquad \text{selection equation}$$

where

$$\begin{aligned} u_1 &\sim N(0, \sigma) \\ u_2 &\sim N(0, 1) \\ \text{corr}(u_1, u_2) &= \rho \end{aligned}$$

When  $\rho \neq 0$ , standard regression techniques applied to the first equation yield biased results. `heckman` provides consistent, asymptotically efficient estimates for all the parameters in such models.

In one classic example, the first equation describes the wages of women. Women choose whether to work, and thus, from our point of view as researchers, whether we observe their wages in our data. If women made this decision randomly, we could ignore that not all wages are observed and use ordinary regression to fit a wage model. Such an assumption of random participation, however, is unlikely to be true; women who would have low wages may be unlikely to choose to work, and thus the sample of observed wages is biased upward. In the jargon of economics, women choose not to work when their personal reservation wage is greater than the wage offered by employers. Thus

women who choose not to work might have even higher offer wages than those who do work—they may have high offer wages, but they have even higher reservation wages. We could tell a story that competency is related to wages, but competency is rewarded more at home than in the labor force.

In any case, in this problem—which is the paradigm for most such problems—a solution can be found if there are some variables that strongly affect the chances for observation (the reservation wage) but not the outcome under study (the offer wage). Such a variable might be the number of children in the home. (Theoretically, we do not need such identifying variables, but without them, we depend on functional form to identify the model. It would be difficult for anyone to take such results seriously because the functional-form assumptions have no firm basis in theory.)

## ► Example 1

In the syntax for `heckman`, *depvar* and *indepvars* are the dependent variable and regressors for the underlying regression model to be fit ( $\mathbf{y} = \mathbf{X}\beta$ ), and *varlist<sub>s</sub>* are the variables ( $\mathbf{Z}$ ) thought to determine whether *depvar* is observed or unobserved (selected or not selected). In our female wage example, the number of children at home would be included in the second list. By default, `heckman` assumes that missing values (see [U] 12.2.1 Missing values) of *depvar* imply that the dependent variable is unobserved (not selected). With some datasets, it is more convenient to specify a binary variable (*depvar<sub>s</sub>*) that identifies the observations for which the dependent is observed/selected (*depvar<sub>s</sub>* ≠ 0) or not observed (*depvar<sub>s</sub>* = 0); `heckman` will accommodate either type of data.

We have a (fictional) dataset on 2,000 women, 1,343 of whom work:

```
. use http://www.stata-press.com/data/r11/womenwk
. summarize age educ married children wage
```

| Variable  | Obs  | Mean     | Std. Dev. | Min     | Max      |
|-----------|------|----------|-----------|---------|----------|
| age       | 2000 | 36.208   | 8.28656   | 20      | 59       |
| education | 2000 | 13.084   | 3.045912  | 10      | 20       |
| married   | 2000 | .6705    | .4701492  | 0       | 1        |
| children  | 2000 | 1.6445   | 1.398963  | 0       | 5        |
| wage      | 1343 | 23.69217 | 6.305374  | 5.88497 | 45.80979 |

We will assume that the hourly wage is a function of education and age, whereas the likelihood of working (the likelihood of the wage being observed) is a function of marital status, the number of children at home, and (implicitly) the wage (via the inclusion of age and education, which we think determine the wage):

(Continued on next page)

```
. heckman wage educ age, select(married children educ age)
Iteration 0: log likelihood = -5178.7009
Iteration 1: log likelihood = -5178.3049
Iteration 2: log likelihood = -5178.3045

Heckman selection model Number of obs = 2000
(regression model with sample selection) Censored obs = 657
 Uncensored obs = 1343
 Wald chi2(2) = 508.44
Log likelihood = -5178.304 Prob > chi2 = 0.0000
```

|                                                                                   | Coef.     | Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|-----------------------------------------------------------------------------------|-----------|-----------|--------|-------|----------------------|-----------|
| wage                                                                              |           |           |        |       |                      |           |
| education                                                                         | .9899537  | .0532565  | 18.59  | 0.000 | .8855729             | 1.094334  |
| age                                                                               | .2131294  | .0206031  | 10.34  | 0.000 | .1727481             | .2535108  |
| _cons                                                                             | .4857752  | 1.077037  | 0.45   | 0.652 | -1.625179            | 2.59673   |
| select                                                                            |           |           |        |       |                      |           |
| married                                                                           | .4451721  | .0673954  | 6.61   | 0.000 | .3130794             | .5772647  |
| children                                                                          | .4387068  | .0277828  | 15.79  | 0.000 | .3842534             | .4931601  |
| education                                                                         | .0557318  | .0107349  | 5.19   | 0.000 | .0346917             | .0767718  |
| age                                                                               | .0365098  | .0041533  | 8.79   | 0.000 | .0283694             | .0446502  |
| _cons                                                                             | -2.491015 | .1893402  | -13.16 | 0.000 | -2.862115            | -2.119915 |
| /athrho                                                                           | .8742086  | .1014225  | 8.62   | 0.000 | .6754241             | 1.072993  |
| /lnsigma                                                                          | 1.792559  | .027598   | 64.95  | 0.000 | 1.738468             | 1.84665   |
| rho                                                                               | .7035061  | .0512264  |        |       | .5885365             | .7905862  |
| sigma                                                                             | 6.004797  | .1657202  |        |       | 5.68862              | 6.338548  |
| lambda                                                                            | 4.224412  | .3992265  |        |       | 3.441942             | 5.006881  |
| LR test of indep. eqns. (rho = 0):   chi2(1) =       61.20   Prob > chi2 = 0.0000 |           |           |        |       |                      |           |

heckman assumes that wage is the dependent variable and that the first variable list (educ and age) are the determinants of wage. The variables specified in the select() option (married, children, educ, and age) are assumed to determine whether the dependent variable is observed (the selection equation). Thus we fit the model

$$\text{wage} = \beta_0 + \beta_1\text{educ} + \beta_2\text{age} + u_1$$

and we assumed that wage is observed if

$$\gamma_0 + \gamma_1\text{married} + \gamma_2\text{children} + \gamma_3\text{educ} + \gamma_4\text{age} + u_2 > 0$$

where  $u_1$  and  $u_2$  have correlation  $\rho$ .

The reported results for the wage equation are interpreted exactly as though we observed wage data for all women in the sample; the coefficients on age and education level represent the estimated marginal effects of the regressors in the underlying regression equation. The results for the two ancillary parameters require some explanation. heckman does not directly estimate  $\rho$ ; to constrain  $\rho$  within its valid limits, and for numerical stability during optimization, it estimates the inverse hyperbolic tangent of  $\rho$ :

$$\text{atanh } \rho = \frac{1}{2} \ln \left( \frac{1 + \rho}{1 - \rho} \right)$$



This estimate is reported as `/athrho`. In the bottom panel of the output, `heckman` undoes this transformation for you: the estimated value of  $\rho$  is 0.7035061. The standard error for  $\rho$  is computed using the delta method, and its confidence intervals are the transformed intervals of `/athrho`.

Similarly,  $\sigma$ , the standard error of the residual in the wage equation, is not directly estimated; for numerical stability, `heckman` instead estimates  $\ln \sigma$ . The untransformed `sigma` is reported at the end of the output: 6.004797.

Finally, some researchers—especially economists—are used to the selectivity effect summarized not by  $\rho$  but by  $\lambda = \rho\sigma$ . `heckman` reports this, too, along with an estimate of the standard error and confidence interval.



## □ Technical note

If each of the equations in the model had contained many regressors, the `heckman` command could have become long. An alternate way of specifying our wage model would be to use Stata's global macros. The following lines are an equivalent way of specifying our model:

```
. global wageeq "wage educ age"
. global seleq "married children educ age"
. heckman $wageeq, select($seleq)
(output omitted)
```



## □ Technical note

The reported model  $\chi^2$  test is a Wald test that all coefficients in the regression model (except the constant) are 0. `heckman` is an estimation command, so you can use `test`, `testnl`, or `lrtest` to perform tests against whatever nested alternate model you choose; see [\[R\] test](#), [\[R\] testnl](#), and [\[R\] lrtest](#).

The estimation of  $\rho$  and  $\sigma$  in the forms  $\text{atanh } \rho$  and  $\ln \sigma$  extends the range of these parameters to infinity in both directions, thus avoiding boundary problems during the maximization. Tests of  $\rho$  must be made in the transformed units. However, because  $\text{atanh}(0) = 0$ , the reported test for  $\text{atanh } \rho = 0$  is equivalent to the test for  $\rho = 0$ .

The likelihood-ratio test reported at the bottom of the output is an equivalent test for  $\rho = 0$  and is computationally the comparison of the joint likelihood of an independent probit model for the selection equation and a regression model on the observed wage data against the Heckman model likelihood. Because  $\chi^2 = 61.20$ , this clearly justifies the Heckman selection equation with these data.



## ▷ Example 2

`heckman` supports the Huber/White/sandwich estimator of variance under the `vce(robust)` and `vce(cluster clustvar)` options or when `pweights` are used for population-weighted data; see [\[U\] 20.16 Obtaining robust variance estimates](#). We can obtain robust standard errors for our wage model by specifying clustering on county of residence (the `county` variable).

```
. heckman wage educ age, select(married children educ age) vce(cluster county)
Iteration 0: log pseudolikelihood = -5178.7009
Iteration 1: log pseudolikelihood = -5178.3049
Iteration 2: log pseudolikelihood = -5178.3045

Heckman selection model Number of obs = 2000
(regression model with sample selection) Censored obs = 657
 Uncensored obs = 1343
 Wald chi2(1) = .
Log pseudolikelihood = -5178.304 Prob > chi2 = .
 (Std. Err. adjusted for 10 clusters in county)
```

|                                                                           | Coef.     | Robust<br>Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|---------------------------------------------------------------------------|-----------|---------------------|--------|-------|----------------------|-----------|
| wage                                                                      |           |                     |        |       |                      |           |
| education                                                                 | .9899537  | .0600061            | 16.50  | 0.000 | .8723438             | 1.107564  |
| age                                                                       | .2131294  | .020995             | 10.15  | 0.000 | .17198               | .2542789  |
| _cons                                                                     | .4857752  | 1.302103            | 0.37   | 0.709 | -2.066299            | 3.03785   |
| select                                                                    |           |                     |        |       |                      |           |
| married                                                                   | .4451721  | .0731472            | 6.09   | 0.000 | .3018062             | .5885379  |
| children                                                                  | .4387068  | .0312386            | 14.04  | 0.000 | .3774802             | .4999333  |
| education                                                                 | .0557318  | .0110039            | 5.06   | 0.000 | .0341645             | .0772991  |
| age                                                                       | .0365098  | .004038             | 9.04   | 0.000 | .0285954             | .0444242  |
| _cons                                                                     | -2.491015 | .1153305            | -21.60 | 0.000 | -2.717059            | -2.264972 |
| /athrho                                                                   | .8742086  | .1403337            | 6.23   | 0.000 | .5991596             | 1.149258  |
| /lnsigma                                                                  | 1.792559  | .0258458            | 69.36  | 0.000 | 1.741902             | 1.843216  |
| rho                                                                       | .7035061  | .0708796            |        |       | .5364513             | .817508   |
| sigma                                                                     | 6.004797  | .155199             |        |       | 5.708189             | 6.316818  |
| lambda                                                                    | 4.224412  | .5186709            |        |       | 3.207835             | 5.240988  |
| Wald test of indep. eqns. (rho = 0): chi2(1) = 61.20 Prob > chi2 = 0.0000 |           |                     |        |       |                      |           |

The robust standard errors tend to be a bit larger, but we notice no systematic differences. This finding is not surprising because the data were not constructed to have any county-specific correlations or any other characteristics that would deviate from the assumptions of the Heckman model.

➤ Example 3

Stata also produces Heckman’s (1979) two-step efficient estimator of the model with the `twostep` option. Maximum likelihood estimation of the parameters can be time consuming with large datasets, and the two-step estimates may provide a good alternative in such cases. Continuing with the women’s wage model, we can obtain the two-step estimates with Heckman’s consistent covariance estimates by typing

```
. heckman wage educ age, select(married children educ age) twostep
Heckman selection model -- two-step estimates Number of obs = 2000
(regression model with sample selection) Censored obs = 657
 Uncensored obs = 1343
 Wald chi2(4) = 442.54
 Prob > chi2 = 0.0000
```

|           | Coef.     | Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|-----------|-----------|-----------|--------|-------|----------------------|-----------|
| wage      |           |           |        |       |                      |           |
| education | .9825259  | .0538821  | 18.23  | 0.000 | .8769189             | 1.088133  |
| age       | .2118695  | .0220511  | 9.61   | 0.000 | .1686502             | .2550888  |
| _cons     | .7340391  | 1.248331  | 0.59   | 0.557 | -1.712645            | 3.180723  |
| select    |           |           |        |       |                      |           |
| married   | .4308575  | .074208   | 5.81   | 0.000 | .2854125             | .5763025  |
| children  | .4473249  | .0287417  | 15.56  | 0.000 | .3909922             | .5036576  |
| education | .0583645  | .0109742  | 5.32   | 0.000 | .0368555             | .0798735  |
| age       | .0347211  | .0042293  | 8.21   | 0.000 | .0264318             | .0430105  |
| _cons     | -2.467365 | .1925635  | -12.81 | 0.000 | -2.844782            | -2.089948 |
| mills     |           |           |        |       |                      |           |
| lambda    | 4.001615  | .6065388  | 6.60   | 0.000 | 2.812821             | 5.19041   |
| rho       | 0.67284   |           |        |       |                      |           |
| sigma     | 5.9473529 |           |        |       |                      |           |
| lambda    | 4.0016155 | .6065388  |        |       |                      |           |

◀

## □ Technical note

The Heckman selection model depends strongly on the model being correct, much more so than ordinary regression. Running a separate probit or logit for sample inclusion followed by a regression, referred to in the literature as the two-part model ([Manning, Duan, and Rogers 1987](#))—not to be confused with Heckman’s two-step procedure—is an especially attractive alternative if the regression part of the model arose because of taking a logarithm of zero values. When the goal is to analyze an underlying regression model or to predict the value of the dependent variable that would be observed in the absence of selection, however, the Heckman model is more appropriate. When the goal is to predict an actual response, the two-part model is usually the better choice.

The Heckman selection model can be unstable when the model is not properly specified or if a specific dataset simply does not support the model’s assumptions. For example, let’s examine the solution to another simulated problem.

(Continued on next page)

```
. use http://www.stata-press.com/data/r11/twopart
. heckman yt x1 x2 x3, select(z1 z2)

Iteration 0: log likelihood = -111.94996
Iteration 1: log likelihood = -110.82258
Iteration 2: log likelihood = -110.17707
Iteration 3: log likelihood = -107.70663 (not concave)
Iteration 4: log likelihood = -107.07729 (not concave)
(output omitted)
Iteration 42: log likelihood = -104.08246
Iteration 43: log likelihood = -104.08246

Heckman selection model Number of obs = 150
(regression model with sample selection) Censored obs = 87
 Uncensored obs = 63
 Wald chi2(3) = 5.31e+10
 Prob > chi2 = 0.0000

Log likelihood = -104.0825
```

|                                                                         | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-------------------------------------------------------------------------|-----------|-----------|-------|-------|----------------------|-----------|
| yt                                                                      |           |           |       |       |                      |           |
| x1                                                                      | .8974192  | .0000272  | .     | 0.000 | .897366              | .8974724  |
| x2                                                                      | -2.525303 | .000016   | .     | 0.000 | -2.525334            | -2.525271 |
| x3                                                                      | 2.855786  | .0000371  | .     | 0.000 | 2.855713             | 2.855858  |
| _cons                                                                   | .6975109  | .0705485  | 9.89  | 0.000 | .5592383             | .8357835  |
| select                                                                  |           |           |       |       |                      |           |
| z1                                                                      | -.6826393 | .0794597  | -8.59 | 0.000 | -.8383775            | -.5269012 |
| z2                                                                      | 1.003665  | .1168273  | 8.59  | 0.000 | .7746873             | 1.232642  |
| _cons                                                                   | -.3605445 | .1506868  | -2.39 | 0.000 | -.6558852            | -.0652039 |
| /athrho                                                                 | 17.66202  | 161.3175  | 0.11  | 0.957 | -298.5144            | 333.8385  |
| /lnsigma                                                                | -.5396747 | .1164007  | -4.64 | 0.000 | -.7678159            | -.315335  |
| rho                                                                     | 1         | 3.22e-13  |       |       | -1                   | 1         |
| sigma                                                                   | .5829739  | .0678544  |       |       | .4870425             | .7323231  |
| lambda                                                                  | .5829739  | .0678544  |       |       | .4781604             | .71593    |
| LR test of indep. eqns. (rho = 0): chi2(1) = 25.67 Prob > chi2 = 0.0000 |           |           |       |       |                      |           |

The model has converged to a value of  $\rho$  that is 1.0—within machine-rounding tolerances. Given the form of the likelihood for the Heckman selection model, this implies a division by zero, and it is surprising that the model solution turns out as well as it does. Reparameterizing  $\rho$  has allowed the estimation to converge, but we clearly have problems with the estimates. Moreover, if this had occurred in a large dataset, waiting 43 iterations for convergence might take considerable time.

This dataset was not intentionally developed to cause problems. It is actually generated by a “Heckman process” and when generated starting from different random values can be easily estimated. The luck of the draw here merely led to data that, despite the source, did not support the assumptions of the Heckman model.

The two-step model is generally more stable when the data are problematic. It even tolerates estimates of  $\rho$  less than  $-1$  and greater than  $1$ . For these reasons, the two-step model may be preferred when exploring a large dataset. Still, if the maximum likelihood estimates cannot converge, or converge to a value of  $\rho$  that is at the boundary of acceptable values, there is scant support for fitting a Heckman selection model on the data. Heckman (1979) discusses the implications of  $\rho$  being exactly  $1$  or  $0$ , together with the implications of other possible covariance relationships among the model’s determinants.



James Joseph Heckman was born in Chicago in 1944 and studied mathematics at Colorado College and economics at Princeton. He has taught economics at Columbia and (since 1973) at the University of Chicago. He has worked on developing a scientific basis for economic policy evaluation, with emphasis on models of individuals or disaggregated groups and the problems and possibilities created by heterogeneity, diversity, and unobserved counterfactual states. In 2000, he shared the Nobel Prize in Economics with Daniel L. McFadden.

## Saved results

`heckman` (maximum likelihood) saves the following in `e()`:

### Scalars

|                            |                                                   |
|----------------------------|---------------------------------------------------|
| <code>e(N)</code>          | number of observations                            |
| <code>e(N_cens)</code>     | number of censored observations                   |
| <code>e(k)</code>          | number of parameters                              |
| <code>e(k_eq)</code>       | number of equations                               |
| <code>e(k_eq_model)</code> | number of equations in model Wald test            |
| <code>e(k_aux)</code>      | number of auxiliary parameters                    |
| <code>e(k_dv)</code>       | number of dependent variables                     |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints    |
| <code>e(df_m)</code>       | model degrees of freedom                          |
| <code>e(ll)</code>         | log likelihood                                    |
| <code>e(ll_0)</code>       | log likelihood, constant-only model               |
| <code>e(N_clust)</code>    | number of clusters                                |
| <code>e(lambda)</code>     | $\lambda$                                         |
| <code>e(selambda)</code>   | standard error of $\lambda$                       |
| <code>e(sigma)</code>      | $\sigma$                                          |
| <code>e(chi2)</code>       | $\chi^2$                                          |
| <code>e(chi2_c)</code>     | $\chi^2$ for comparison test                      |
| <code>e(p_c)</code>        | $p$ -value for comparison test                    |
| <code>e(p)</code>          | significance of comparison test                   |
| <code>e(rho)</code>        | $\rho$                                            |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                         |
| <code>e(rank0)</code>      | rank of <code>e(V)</code> for constant-only model |
| <code>e(ic)</code>         | number of iterations                              |
| <code>e(rc)</code>         | return code                                       |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                       |

*(Continued on next page)*

**Macros**

|                                 |                                                                                 |
|---------------------------------|---------------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>heckman</code>                                                            |
| <code>e(cmdline)</code>         | command as typed                                                                |
| <code>e(depvar)</code>          | names of dependent variables                                                    |
| <code>e(wtype)</code>           | weight type                                                                     |
| <code>e(wexp)</code>            | weight expression                                                               |
| <code>e(title)</code>           | title in estimation output                                                      |
| <code>e(title2)</code>          | secondary title in estimation output                                            |
| <code>e(clustvar)</code>        | name of cluster variable                                                        |
| <code>e(offset1)</code>         | offset for regression equation                                                  |
| <code>e(offset2)</code>         | offset for selection equation                                                   |
| <code>e(mills)</code>           | variable containing nonselection hazard (inverse of Mills')                     |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                         |
| <code>e(chi2_ct)</code>         | Wald or LR; type of model $\chi^2$ test corresponding to <code>e(chi2_c)</code> |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                                  |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                                   |
| <code>e(diparm#)</code>         | display transformed parameter #                                                 |
| <code>e(opt)</code>             | type of optimization                                                            |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization        |
| <code>e(method)</code>          | <code>ml</code>                                                                 |
| <code>e(ml_method)</code>       | type of <code>ml</code> method                                                  |
| <code>e(user)</code>            | name of likelihood-evaluator program                                            |
| <code>e(technique)</code>       | maximization technique                                                          |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular                     |
| <code>e(crittype)</code>        | optimization criterion                                                          |
| <code>e(properties)</code>      | <code>b V</code>                                                                |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                                  |
| <code>e(marginsok)</code>       | predictions allowed by <code>margins</code>                                     |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>                  |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>                  |

**Matrices**

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(gradient)</code>     | gradient vector                              |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

**Functions**

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

`heckman` (two-step) saves the following in `e()`:

#### Scalars

|                          |                                 |
|--------------------------|---------------------------------|
| <code>e(N)</code>        | number of observations          |
| <code>e(N_cens)</code>   | number of censored observations |
| <code>e(df_m)</code>     | model degrees of freedom        |
| <code>e(lambda)</code>   | $\lambda$                       |
| <code>e(selambda)</code> | standard error of $\lambda$     |
| <code>e(sigma)</code>    | $\sigma$                        |
| <code>e(chi2)</code>     | $\chi^2$                        |
| <code>e(p)</code>        | significance of comparison test |
| <code>e(rho)</code>      | $\rho$                          |
| <code>e(rank)</code>     | rank of <code>e(V)</code>       |

#### Macros

|                              |                                                                                                       |
|------------------------------|-------------------------------------------------------------------------------------------------------|
| <code>e(cmd)</code>          | <code>heckman</code>                                                                                  |
| <code>e(cmdline)</code>      | command as typed                                                                                      |
| <code>e(depvar)</code>       | names of dependent variables                                                                          |
| <code>e(title)</code>        | title in estimation output                                                                            |
| <code>e(title2)</code>       | secondary title in estimation output                                                                  |
| <code>e(mills)</code>        | variable containing nonselection hazard (inverse of Mills')                                           |
| <code>e(chi2type)</code>     | Wald or LR; type of model $\chi^2$ test                                                               |
| <code>e(vce)</code>          | <code>vcetype</code> specified in <code>vce()</code>                                                  |
| <code>e(vcetype)</code>      | title used to label Std. Err.                                                                         |
| <code>e(rhometh)</code>      | <code>rhosigma</code> , <code>rho trunc</code> , <code>rho limited</code> , or <code>rho force</code> |
| <code>e(method)</code>       | <code>twostep</code>                                                                                  |
| <code>e(properties)</code>   | <code>b V</code>                                                                                      |
| <code>e(predict)</code>      | program used to implement <code>predict</code>                                                        |
| <code>e(marginsok)</code>    | predictions allowed by <code>margins</code>                                                           |
| <code>e(marginsnotok)</code> | predictions disallowed by <code>margins</code>                                                        |
| <code>e(asbalanced)</code>   | factor variables <code>fvset</code> as <code>asbalanced</code>                                        |
| <code>e(asobserved)</code>   | factor variables <code>fvset</code> as <code>asobserved</code>                                        |

#### Matrices

|                   |                                              |
|-------------------|----------------------------------------------|
| <code>e(b)</code> | coefficient vector                           |
| <code>e(V)</code> | variance–covariance matrix of the estimators |

#### Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`heckman` is implemented as an ado-file. [Cameron and Trivedi \(2009, 541–547\)](#) and [Greene \(2008, 884–889\)](#) provide good introductions to the Heckman selection model. [Adkins and Hill \(2008, 395–400\)](#) describe the two-step estimator with an application using Stata. [Jones \(2007, 35–40\)](#) illustrates Heckman estimation with an application to health economics.

Regression estimates using the nonselection hazard ([Heckman 1979](#)) provide starting values for maximum likelihood estimation.

The regression equation is

$$y_j = \mathbf{x}_j\boldsymbol{\beta} + u_{1j}$$

The selection equation is

$$\mathbf{z}_j\boldsymbol{\gamma} + u_{2j} > 0$$

where

$$\begin{aligned} u_1 &\sim N(0, \sigma) \\ u_2 &\sim N(0, 1) \\ \text{corr}(u_1, u_2) &= \rho \end{aligned}$$

The log likelihood for observation  $j$ ,  $\ln L_j = l_j$ , is

$$l_j = \begin{cases} w_j \ln \Phi \left\{ \frac{\mathbf{z}_j \gamma + (y_j - \mathbf{x}_j \beta) \rho / \sigma}{\sqrt{1 - \rho^2}} \right\} - \frac{w_j}{2} \left( \frac{y_j - \mathbf{x}_j \beta}{\sigma} \right)^2 - w_j \ln(\sqrt{2\pi} \sigma) & y_j \text{ observed} \\ w_j \ln \Phi(-\mathbf{z}_j \gamma) & y_j \text{ not observed} \end{cases}$$

where  $\Phi(\cdot)$  is the standard cumulative normal and  $w_j$  is an optional weight for observation  $j$ .

In the maximum likelihood estimation,  $\sigma$  and  $\rho$  are not directly estimated. Directly estimated are  $\ln \sigma$  and  $\text{atanh } \rho$ :

$$\text{atanh } \rho = \frac{1}{2} \ln \left( \frac{1 + \rho}{1 - \rho} \right)$$

The standard error of  $\lambda = \rho \sigma$  is approximated through the propagation of error (delta) method; that is,

$$\text{Var}(\lambda) \approx \mathbf{D} \text{Var}\{(\text{atanh } \rho \quad \ln \sigma)\} \mathbf{D}'$$

where  $\mathbf{D}$  is the Jacobian of  $\lambda$  with respect to  $\text{atanh } \rho$  and  $\ln \sigma$ .

With maximum likelihood estimation, this command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

The maximum likelihood version of heckman also supports estimation with survey data. For details on VCEs with survey data, see [\[SVY\] variance estimation](#).

The two-step estimates are computed using Heckman's (1979) procedure.

Probit estimates of the selection equation

$$\Pr(y_j \text{ observed} \mid \mathbf{z}_j) = \Phi(\mathbf{z}_j \gamma)$$

are obtained. From these estimates, the nonselection hazard—what Heckman (1979) referred to as the inverse of the Mills' ratio,  $m_j$ —for each observation  $j$  is computed as

$$m_j = \frac{\phi(\mathbf{z}_j \hat{\gamma})}{\Phi(\mathbf{z}_j \hat{\gamma})}$$

where  $\phi$  is the normal density. We also define

$$\delta_j = m_j(m_j + \hat{\gamma} \mathbf{z}_j)$$

Following Heckman, the two-step parameter estimates of  $\beta$  are obtained by augmenting the regression equation with the nonselection hazard  $\mathbf{m}$ . Thus the regressors become  $[\mathbf{X} \quad \mathbf{m}]$ , and we obtain the additional parameter estimate  $\beta_m$  on the variable containing the nonselection hazard.



A consistent estimate of the regression disturbance variance is obtained using the residuals from the augmented regression and the parameter estimate on the nonselection hazard,

$$\hat{\sigma}^2 = \frac{\mathbf{e}'\mathbf{e} + \beta_m^2 \sum_{j=1}^N \delta_j}{N}$$

The two-step estimate of  $\rho$  is then

$$\hat{\rho} = \frac{\beta_m}{\hat{\sigma}}$$

Heckman derived consistent estimates of the coefficient covariance matrix on the basis of the augmented regression.

Let  $\mathbf{W} = [\mathbf{X} \ \mathbf{m}]$  and  $\mathbf{R}$  be a square, diagonal matrix of dimension  $N$ , with  $(1 - \hat{\rho}^2 \delta_j)$  as the diagonal elements. The conventional VCE is

$$\mathbf{V}_{\text{twostep}} = \hat{\sigma}^2 (\mathbf{W}'\mathbf{W})^{-1} (\mathbf{W}'\mathbf{R}\mathbf{W} + \mathbf{Q}) (\mathbf{W}'\mathbf{W})^{-1}$$

where

$$\mathbf{Q} = \hat{\rho}^2 (\mathbf{W}'\mathbf{D}\mathbf{Z}) \mathbf{V}_p (\mathbf{Z}'\mathbf{D}\mathbf{W})$$

where  $\mathbf{D}$  is the square, diagonal matrix of dimension  $N$  with  $\delta_j$  as the diagonal elements;  $\mathbf{Z}$  is the data matrix of selection equation covariates; and  $\mathbf{V}_p$  is the variance–covariance estimate from the probit estimation of the selection equation.

## References

- Adkins, L. C., and R. C. Hill. 2008. *Using Stata for Principles of Econometrics*. 3rd ed. Hoboken, NJ: Wiley.
- Baum, C. F. 2006. *An Introduction to Modern Econometrics Using Stata*. College Station, TX: Stata Press.
- Cameron, A. C., and P. K. Trivedi. 2009. *Microeconometrics Using Stata*. College Station, TX: Stata Press.
- Chiburis, R., and M. Lokshin. 2007. Maximum likelihood and two-step estimation of an ordered-probit selection model. *Stata Journal* 7: 167–182.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- Gronau, R. 1974. Wage comparisons: A selectivity bias. *Journal of Political Economy* 82: 1119–1143.
- Heckman, J. 1976. The common structure of statistical models of truncation, sample selection and limited dependent variables and a simple estimator for such models. *Annals of Economic and Social Measurement* 5: 475–492.
- . 1979. Sample selection bias as a specification error. *Econometrica* 47: 153–161.
- Jones, A. 2007. *Applied Econometrics for Health Economists: A Practical Guide*. 2nd ed. Abingdon, UK: Radcliffe.
- Lewis, H. G. 1974. Comments on selectivity biases in wage comparisons. *Journal of Political Economy* 82: 1145–1155.
- Manning, W. G., N. Duan, and W. H. Rogers. 1987. Monte Carlo evidence on the choice between sample selection and two-part models. *Journal of Econometrics* 35: 59–82.

## Also see

- [R] [heckman postestimation](#) — Postestimation tools for heckman
- [R] [heckprob](#) — Probit model with sample selection
- [R] [regress](#) — Linear regression
- [R] [tobit](#) — Tobit regression
- [SVY] [svy estimation](#) — Estimation commands for survey data
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `heckman`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code> <sup>1</sup>  | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code> <sup>2</sup> | likelihood-ratio test; not available with two-step estimator                                        |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code> <sup>1</sup>  | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `estat ic` and `suest` are not appropriate after `heckman`, `twostep`.

<sup>2</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

## Syntax for predict

After *ML* or *twostep*

```
predict [type] newvar [if] [in] [, statistic nooffset]
```

After *ML*

```
predict [type] { stub*|newvarreg newvarsel newvarathrho newvarlnsigma }
[if] [in] , scores
```

*statistic*

description

Main

|                                 |                                                                |
|---------------------------------|----------------------------------------------------------------|
| <b>xb</b>                       | linear prediction; the default                                 |
| <b>stdp</b>                     | standard error of the prediction                               |
| <b>stdf</b>                     | standard error of the forecast                                 |
| <b>xb<sub>sel</sub></b>         | linear prediction for selection equation                       |
| <b>stdp<sub>sel</sub></b>       | standard error of the linear prediction for selection equation |
| <b>pr(<i>a</i>,<i>b</i>)</b>    | $\Pr(y_j \mid a < y_j < b)$                                    |
| <b>e(<i>a</i>,<i>b</i>)</b>     | $E(y_j \mid a < y_j < b)$                                      |
| <b>ystar(<i>a</i>,<i>b</i>)</b> | $E(y_j^*), y_j^* = \max\{a, \min(y_j, b)\}$                    |
| <b>ycond</b>                    | $E(y_j \mid y_j \text{ observed})$                             |
| <b>yexpected</b>                | $E(y_j^*), y_j$ taken to be 0 where unobserved                 |
| <b>nshazard</b> or <b>mills</b> | nonselection hazard (also called the inverse of Mills' ratio)  |
| <b>psel</b>                     | $\Pr(y_j \text{ observed})$                                    |

These statistics are available both in and out of sample; type `predict ... if e(sample)` ... if wanted only for the estimation sample.

`stdf` is not allowed with `svy` estimation results.

where *a* and *b* may be numbers or variables; *a* missing ( $a \geq .$ ) means  $-\infty$ , and *b* missing ( $b \geq .$ ) means  $+\infty$ ; see [U] [12.2.1 Missing values](#).

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

**xb**, the default, calculates the linear prediction  $\mathbf{x}_j\mathbf{b}$ .

**stdp** calculates the standard error of the prediction, which can be thought of as the standard error of the predicted expected value or mean for the observation's covariate pattern. The standard error of the prediction is also referred to as the standard error of the fitted value.

**stdf** calculates the standard error of the forecast, which is the standard error of the point prediction for 1 observation. It is commonly referred to as the standard error of the future or forecast value. By construction, the standard errors produced by `stdf` are always larger than those produced by `stdp`; see [Methods and formulas](#) in [R] [regress](#).

**xbse1** calculates the linear prediction for the selection equation.

**stdpse1** calculates the standard error of the linear prediction for the selection equation.

**pr(*a*,*b*)** calculates  $\Pr(a < \mathbf{x}_j\mathbf{b} + u_1 < b)$ , the probability that  $y_j|\mathbf{x}_j$  would be observed in the interval  $(a, b)$ .

*a* and *b* may be specified as numbers or variable names; *lb* and *ub* are variable names;

**pr(20,30)** calculates  $\Pr(20 < \mathbf{x}_j\mathbf{b} + u_1 < 30)$ ; **pr(*lb*,*ub*)** calculates  $\Pr(lb < \mathbf{x}_j\mathbf{b} + u_1 < ub)$ ; and **pr(20,*ub*)** calculates  $\Pr(20 < \mathbf{x}_j\mathbf{b} + u_1 < ub)$ .

*a* missing (*a* ≥ .) means  $-\infty$ ; **pr(. ,30)** calculates  $\Pr(-\infty < \mathbf{x}_j\mathbf{b} + u_j < 30)$ ;

**pr(*lb*,30)** calculates  $\Pr(-\infty < \mathbf{x}_j\mathbf{b} + u_j < 30)$  in observations for which *lb* ≥ . and calculates  $\Pr(lb < \mathbf{x}_j\mathbf{b} + u_j < 30)$  elsewhere.

*b* missing (*b* ≥ .) means  $+\infty$ ; **pr(20,.)** calculates  $\Pr(+\infty > \mathbf{x}_j\mathbf{b} + u_j > 20)$ ;

**pr(20,*ub*)** calculates  $\Pr(+\infty > \mathbf{x}_j\mathbf{b} + u_j > 20)$  in observations for which *ub* ≥ . and calculates  $\Pr(20 < \mathbf{x}_j\mathbf{b} + u_j < ub)$  elsewhere.

**e(*a*,*b*)** calculates  $E(\mathbf{x}_j\mathbf{b} + u_1 \mid a < \mathbf{x}_j\mathbf{b} + u_1 < b)$ , the expected value of  $y_j|\mathbf{x}_j$  conditional on  $y_j|\mathbf{x}_j$  being in the interval  $(a, b)$ , meaning that  $y_j|\mathbf{x}_j$  is censored.

*a* and *b* are specified as they are for **pr()**.

**ystar(*a*,*b*)** calculates  $E(y_j^*)$ , where  $y_j^* = a$  if  $\mathbf{x}_j\mathbf{b} + u_j \leq a$ ,  $y_j^* = b$  if  $\mathbf{x}_j\mathbf{b} + u_j \geq b$ , and  $y_j^* = \mathbf{x}_j\mathbf{b} + u_j$  otherwise, meaning that  $y_j^*$  is truncated. *a* and *b* are specified as they are for **pr()**.

**ycond** calculates the expected value of the dependent variable conditional on the dependent variable being observed, i.e., selected;  $E(y_j \mid y_j \text{ observed})$ .

**yexpected** calculates the expected value of the dependent variable ( $y_j^*$ ), where that value is taken to be 0 when it is expected to be unobserved;  $y_j^* = \Pr(y_j \text{ observed})E(y_j \mid y_j \text{ observed})$ .

The assumption of 0 is valid for many cases where nonselection implies nonparticipation (e.g., unobserved wage levels, insurance claims from those who are uninsured) but may be inappropriate for some problems (e.g., unobserved disease incidence).

**nshazard** and **mills** are synonyms; both calculate the nonselection hazard—what Heckman (1979) referred to as the inverse of the Mills' ratio—from the selection equation.

**pse1** calculates the probability of selection (or being observed):

$$\Pr(y_j \text{ observed}) = \Pr(\mathbf{z}_j\boldsymbol{\gamma} + u_{2j} > 0).$$

**nooffset** is relevant when you specify **offset(*varname*)** for **heckman**. It modifies the calculations made by **predict** so that they ignore the offset variable; the linear prediction is treated as  $\mathbf{x}_j\mathbf{b}$  rather than as  $\mathbf{x}_j\mathbf{b} + \text{offset}_j$ .

**scores**, not available with **twostep**, calculates equation-level score variables.

The first new variable will contain  $\partial \ln L / \partial (\mathbf{x}_j\boldsymbol{\beta})$ .

The second new variable will contain  $\partial \ln L / \partial (\mathbf{z}_j\boldsymbol{\gamma})$ .

The third new variable will contain  $\partial \ln L / \partial (\tanh \rho)$ .

The fourth new variable will contain  $\partial \ln L / \partial (\ln \sigma)$ .

## Remarks

### ► Example 1

The default statistic produced by `predict` after `heckman` is the expected value of the dependent variable from the underlying distribution of the regression model. In the [wage model](#) of [R] [heckman](#), this is the expected wage rate among all women, regardless of whether they were observed to participate in the labor force:

```
. use http://www.stata-press.com/data/r11/womenwk
. heckman wage educ age, select(married children educ age) vce(cluster county)
 (output omitted)
. predict heckwage
 (option xb assumed; fitted values)
```

It is instructive to compare these predicted wage values from the Heckman model with an ordinary regression model—a model without the selection adjustment:

```
. regress wage educ age
```

| Source   | SS         | df   | MS         |                 |        |
|----------|------------|------|------------|-----------------|--------|
| Model    | 13524.0337 | 2    | 6762.01687 | Number of obs = | 1343   |
| Residual | 39830.8609 | 1340 | 29.7245231 | F( 2, 1340) =   | 227.49 |
|          |            |      |            | Prob > F =      | 0.0000 |
|          |            |      |            | R-squared =     | 0.2535 |
|          |            |      |            | Adj R-squared = | 0.2524 |
| Total    | 53354.8946 | 1342 | 39.7577456 | Root MSE =      | 5.452  |

| wage      | Coef.    | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|-----------|----------|-----------|-------|-------|----------------------|----------|
| education | .8965829 | .0498061  | 18.00 | 0.000 | .7988765             | .9942893 |
| age       | .1465739 | .0187135  | 7.83  | 0.000 | .109863              | .1832848 |
| _cons     | 6.084875 | .8896182  | 6.84  | 0.000 | 4.339679             | 7.830071 |

```
. predict regwage
 (option xb assumed; fitted values)
. summarize heckwage regwage
```

| Variable | Obs  | Mean     | Std. Dev. | Min      | Max      |
|----------|------|----------|-----------|----------|----------|
| heckwage | 2000 | 21.15532 | 3.83965   | 14.6479  | 32.85949 |
| regwage  | 2000 | 23.12291 | 3.241911  | 17.98218 | 32.66439 |

Since this dataset was concocted, we know the true coefficients of the wage regression equation to be 1, 0.2, and 1, respectively. We can compute the true mean wage for our sample.

```
. generate truewage = 1 + .2*age + 1*educ
. summarize truewage
```

| Variable | Obs  | Mean    | Std. Dev. | Min | Max  |
|----------|------|---------|-----------|-----|------|
| truewage | 2000 | 21.3256 | 3.797904  | 15  | 32.8 |

Whereas the mean of the predictions from `heckman` is within 18 cents of the true mean wage, ordinary regression yields predictions that are on average about \$1.80 per hour too high because of the selection effect. The regression predictions also show somewhat less variation than the true wages.

The coefficients from `heckman` are so close to the true values that they are not worth testing. Conversely, the regression equation is significantly off but seems to give the right sense. Would we be led far astray if we relied on the OLS coefficients? The effect of age is off by more than 5 cents per year of age, and the coefficient on education level is off by about 10%. We can test the OLS coefficient on education level against the true value by using `test`.

```
. test educ = 1
(1) education = 1
 F(1, 1340) = 4.31
 Prob > F = 0.0380
```

Not only is the OLS coefficient on education substantially lower than the true parameter, but the difference from the true parameter is also statistically significant beyond the 5% level. We can perform a similar test for the OLS age coefficient:

```
. test age = .2
(1) age = .2
 F(1, 1340) = 8.15
 Prob > F = 0.0044
```

We find even stronger evidence that the OLS regression results are biased away from the true parameters.



### ► Example 2

Several other interesting aspects of the Heckman model can be explored with `predict`. Continuing with our wage model, we can obtain the expected wages for women conditional on participating in the labor force with the `ycond` option. Let’s get these predictions and compare them with actual wages for women participating in the labor force.

```
. use http://www.stata-press.com/data/r11/womenwk, clear
. heckman wage educ age, select(married children educ age)
 (output omitted)
. predict hcndwage, ycond
. summarize wage hcndwage if wage != .
```

| Variable | Obs  | Mean     | Std. Dev. | Min      | Max      |
|----------|------|----------|-----------|----------|----------|
| wage     | 1343 | 23.69217 | 6.305374  | 5.88497  | 45.80979 |
| hcndwage | 1343 | 23.68239 | 3.335087  | 16.18337 | 33.7567  |

We see that the average predictions from `heckman` are close to the observed levels but do not have the same mean. These conditional wage predictions are available for all observations in the dataset but can be directly compared only with observed wages, where individuals are participating in the labor force.

What if we were interested in making predictions about mean wages for all women? Here the expected wage is 0 for those who are not expected to participate in the labor force, with expected participation determined by the selection equation. These values can be obtained with the `yexpected` option of `predict`. For comparison, a variable can be generated where the wage is set to 0 for nonparticipants.

```
. predict hexpwage, yexpected
. generate wage0 = wage
(657 missing values generated)
. replace wage0 = 0 if wage == .
(657 real changes made)
```

```
. summarize hexpwage wage0
```

| Variable | Obs  | Mean     | Std. Dev. | Min      | Max      |
|----------|------|----------|-----------|----------|----------|
| hexpwage | 2000 | 15.92511 | 5.979336  | 2.492469 | 32.45858 |
| wage0    | 2000 | 15.90929 | 12.27081  | 0        | 45.80979 |

Again we note that the predictions from `heckman` are close to the observed mean hourly wage rate for all women. Why aren't the predictions using `ycond` and `yexpected` equal to their observed sample equivalents? For the Heckman model, unlike linear regression, the sample moments implied by the optimal solution to the model likelihood do not require that these predictions match observed data. Properly accounting for the additional variation from the selection equation requires that the model use more information than just the sample moments of the observed wages.

◀

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Reference

Heckman, J. 1979. Sample selection bias as a specification error. *Econometrica* 47: 153–161.

## Also see

[R] [heckman](#) — Heckman selection model

[U] [20 Estimation and postestimation commands](#)

Syntax

```
heckprob depvar indepvars [if] [in] [weight] ,
 select([depvars =] varlists [, offset(varname) noconstant]) [options]
```

| <i>options</i>                            | description                                                                                                             |
|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| Model                                     |                                                                                                                         |
| * <u>select</u> ()                        | specify selection equation: dependent and independent variables; whether to have constant term and offset variable      |
| <u>noconstant</u>                         | suppress constant term                                                                                                  |
| <u>offset</u> ( <i>varname</i> )          | include <i>varname</i> in model with coefficient constrained to 1                                                       |
| <u>constraints</u> ( <i>constraints</i> ) | apply specified linear constraints                                                                                      |
| <u>collinear</u>                          | keep collinear variables                                                                                                |
| SE/Robust                                 |                                                                                                                         |
| <u>vce</u> ( <i>vcetype</i> )             | <i>vcetype</i> may be oim, <u>robust</u> , <u>cluster</u> <i>clustvar</i> , opg, <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting                                 |                                                                                                                         |
| <u>level</u> (#)                          | set confidence level; default is level(95)                                                                              |
| <u>first</u>                              | report first-step probit estimates                                                                                      |
| <u>noskip</u>                             | perform likelihood-ratio test                                                                                           |
| <u>nocnsreport</u>                        | do not display constraints                                                                                              |
| <u>display_options</u>                    | control spacing and display of omitted variables and base and empty cells                                               |
| Maximization                              |                                                                                                                         |
| <u>maximize_options</u>                   | control the maximization process; seldom used                                                                           |
| † <u>coeflegend</u>                       | display coefficients' legend instead of coefficient table                                                               |

\*select( ) is required.

The full specification is select( [*depvar*<sub>s</sub> = ] *varlist*<sub>s</sub> [ , offset(*varname*) noconstant ] ) .

†coeflegend does not appear in the dialog box.

*indepvars* and *varlist*<sub>s</sub> may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar*, *indepvars*, *depvar*<sub>s</sub>, and *varlist*<sub>s</sub> may contain time-series operators; see [U] 11.4.4 Time-series varlists.

bootstrap, by, jackknife, rolling, statsby, and svy are allowed; see [U] 11.1.10 Prefix commands.

Weights are not allowed with the bootstrap prefix.

vce( ), first, noskip, and weights are not allowed with the svy prefix.

pweights, fweights, and iweights are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.



## Menu

Statistics > Sample-selection models > Probit model with selection

## Description

`heckprob` fits maximum-likelihood probit models with sample selection.

## Options

### Model

`select(...)` specifies the variables and options for the selection equation. It is an integral part of specifying a selection model and is required. The selection equation should contain at least one variable that is not in the outcome equation.

If `depvars` is specified, it should be coded as 0 or 1, 0 indicating an observation not selected and 1 indicating a selected observation. If `depvars` is not specified, observations for which `depvar` is not missing are assumed selected, and those for which `depvar` is missing are assumed not selected.

`noconstant`, `offset(varname)`, `constraints(constraints)`, `collinear`; see [\[R\] estimation options](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [\[R\] vce\\_option](#).

### Reporting

`level(#)`; see [\[R\] estimation options](#).

`first` specifies that the first-step probit estimates of the selection equation be displayed before estimation.

`noskip` specifies that a full maximum-likelihood model with only a constant for the regression equation be fit. This model is not displayed but is used as the base model to compute a likelihood-ratio test for the model test statistic displayed in the estimation header. By default, the overall model test statistic is an asymptotically equivalent Wald test that all the parameters in the regression equation are zero (except the constant). For many models, this option can substantially increase estimation time.

`nocnsreport`; see [\[R\] estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [\[R\] estimation options](#).

### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrntolerance`, `from(init_specs)`; see [\[R\] maximize](#). These options are seldom used.

Setting the optimization type to `technique(bhhh)` resets the default `vcetype` to `vce(opg)`.

The following option is available with `heckprob` but is not shown in the dialog box: `coeflegend`; see [R] [estimation options](#).

## Remarks

The probit model with sample selection (Van de Ven and Van Pragg 1981) assumes that there exists an underlying relationship

$$y_j^* = \mathbf{x}_j\beta + u_{1j} \quad \text{latent equation}$$

such that we observe only the binary outcome

$$y_j^{\text{probit}} = (y_j^* > 0) \quad \text{probit equation}$$

The dependent variable, however, is not always observed. Rather, the dependent variable for observation  $j$  is observed if

$$y_j^{\text{select}} = (\mathbf{z}_j\gamma + u_{2j} > 0) \quad \text{selection equation}$$

where

$$u_1 \sim N(0, 1)$$

$$u_2 \sim N(0, 1)$$

$$\text{corr}(u_1, u_2) = \rho$$

When  $\rho \neq 0$ , standard probit techniques applied to the first equation yield biased results. `heckprob` provides consistent, asymptotically efficient estimates for all the parameters in such models.

For the model to be well identified, the selection equation should have at least one variable that is not in the probit equation. Otherwise, the model is identified only by functional form, and the coefficients have no structural interpretation.

### ► Example 1

We use the data from Pindyck and Rubinfeld (1998). In this dataset, the variables are whether children attend private school (`private`), number of years the family has been at the present residence (`years`), log of property tax (`logptax`), log of income (`loginc`), and whether one voted for an increase in property taxes (`vote`).

In this example, we alter the meaning of the data. Here we assume that we observe whether children attend private school only if the family votes for increasing the property taxes. This assumption is not true in the dataset, and we make it only to illustrate the use of this command.

We observe whether children attend private school only if the head of household voted for an increase in property taxes. We assume that the vote is affected by the number of years in residence, the current property taxes paid, and the household income. We wish to model whether children are sent to private school on the basis of the number of years spent in the current residence and the current property taxes paid.

```

. use http://www.stata-press.com/data/r11/school
. heckprob private years logptax, sel(vote=years loginc logptax)
Fitting probit model:
Iteration 0: log likelihood = -17.122381
Iteration 1: log likelihood = -16.243974
(output omitted)
Iteration 5: log likelihood = -15.883655
Fitting selection model:
Iteration 0: log likelihood = -63.036914
Iteration 1: log likelihood = -58.534843
Iteration 2: log likelihood = -58.497292
Iteration 3: log likelihood = -58.497288
Comparison: log likelihood = -74.380943
Fitting starting values:
Iteration 0: log likelihood = -40.895684
Iteration 1: log likelihood = -16.654497
(output omitted)
Iteration 7: log likelihood = -15.753765
Fitting full model
Iteration 0: log likelihood = -75.010619 (not concave)
Iteration 1: log likelihood = -74.287538
Iteration 2: log likelihood = -74.250197
Iteration 3: log likelihood = -74.245091
Iteration 4: log likelihood = -74.244973
Iteration 5: log likelihood = -74.244973
Probit model with sample selection Number of obs = 95
 Censored obs = 36
 Uncensored obs = 59
 Wald chi2(2) = 1.04
 Prob > chi2 = 0.5935
Log likelihood = -74.24497

```

|                | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------------|-----------|-----------|-------|-------|----------------------|-----------|
| <b>private</b> |           |           |       |       |                      |           |
| years          | -.1142597 | .1461717  | -0.78 | 0.434 | -.400751             | .1722317  |
| logptax        | .3516098  | 1.016485  | 0.35  | 0.729 | -1.640665            | 2.343884  |
| _cons          | -2.780665 | 6.905838  | -0.40 | 0.687 | -16.31586            | 10.75453  |
| <b>vote</b>    |           |           |       |       |                      |           |
| years          | -.0167511 | .0147735  | -1.13 | 0.257 | -.0457067            | .0122045  |
| loginc         | .9923024  | .4430009  | 2.24  | 0.025 | .1240366             | 1.860568  |
| logptax        | -1.278783 | .5717545  | -2.24 | 0.025 | -2.399401            | -.1581647 |
| _cons          | -.545821  | 4.070418  | -0.13 | 0.893 | -8.523694            | 7.432052  |
| /athrho        | -.8663156 | 1.450028  | -0.60 | 0.550 | -3.708319            | 1.975686  |
| rho            | -.6994973 | .7405342  |       |       | -.9987984            | .9622689  |

```

LR test of indep. eqns. (rho = 0): chi2(1) = 0.27 Prob > chi2 = 0.6020

```

The output shows several iteration logs. The first iteration log corresponds to running the probit model for those observations in the sample where we have observed the outcome. The second iteration log corresponds to running the selection probit model, which models whether we observe our outcome of interest. If  $\rho = 0$ , the sum of the log likelihoods from these two models will equal the log likelihood of the probit model with sample selection; this sum is printed in the iteration log as the comparison log likelihood. The third iteration log shows starting values for the iterations.

The final iteration log is for fitting the full probit model with sample selection. A likelihood-ratio test of the log likelihood for this model and the comparison log likelihood is presented at the end of the output. If we had specified the `vce(robust)` option, this test would be presented as a Wald test instead of as a likelihood-ratio test.



➤ Example 2

In [example 1](#), we could have obtained robust standard errors by specifying the `vce(robust)` option. We do this here and also eliminate the iteration logs by using the `nolog` option:

```
. heckprob private years logptax, sel(vote=years loginc logptax) vce(robust) nolog
Probit model with sample selection Number of obs = 95
 Censored obs = 36
 Uncensored obs = 59
 Wald chi2(2) = 2.55
 Prob > chi2 = 0.2798

Log pseudolikelihood = -74.24497
```

|                                                | Coef.     | Robust<br>Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|------------------------------------------------|-----------|---------------------|-------|-------|----------------------|-----------|
| private                                        |           |                     |       |       |                      |           |
| years                                          | -.1142597 | .1113977            | -1.03 | 0.305 | -.3325952            | .1040758  |
| logptax                                        | .3516098  | .7358267            | 0.48  | 0.633 | -1.090584            | 1.793804  |
| _cons                                          | -2.780665 | 4.786679            | -0.58 | 0.561 | -12.16238            | 6.601053  |
| vote                                           |           |                     |       |       |                      |           |
| years                                          | -.0167511 | .0173344            | -0.97 | 0.334 | -.0507259            | .0172237  |
| loginc                                         | .9923024  | .4228045            | 2.35  | 0.019 | .1636209             | 1.820984  |
| logptax                                        | -1.278783 | .5095156            | -2.51 | 0.012 | -2.277415            | -.2801508 |
| _cons                                          | -.545821  | 4.543892            | -0.12 | 0.904 | -9.451687            | 8.360045  |
| /athrho                                        | -.8663156 | 1.630644            | -0.53 | 0.595 | -4.06232             | 2.329689  |
| rho                                            | -.6994973 | .8327757            |       |       | -.9994079            | .981233   |
| Wald test of indep. eqns. (rho = 0): chi2(1) = |           |                     |       | 0.28  | Prob > chi2 = 0.5952 |           |

Regardless of whether we specify the `vce(robust)` option, the outcome is not significantly different from the outcome obtained by fitting the probit and selection models separately. This result is not surprising because the selection mechanism estimated was invented for the example rather than borne from any economic theory.



## Saved results

heckprob saves the following in `e()`:

### Scalars

|                            |                                                   |
|----------------------------|---------------------------------------------------|
| <code>e(N)</code>          | number of observations                            |
| <code>e(N_cens)</code>     | number of censored observations                   |
| <code>e(k)</code>          | number of parameters                              |
| <code>e(k_eq)</code>       | number of equations                               |
| <code>e(k_eq_model)</code> | number of equations in model Wald test            |
| <code>e(k_aux)</code>      | number of auxiliary parameters                    |
| <code>e(k_dv)</code>       | number of dependent variables                     |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints    |
| <code>e(df_m)</code>       | model degrees of freedom                          |
| <code>e(ll)</code>         | log likelihood                                    |
| <code>e(ll_0)</code>       | log likelihood, constant-only model               |
| <code>e(ll_c)</code>       | log likelihood, comparison model                  |
| <code>e(N_clust)</code>    | number of clusters                                |
| <code>e(chi2)</code>       | $\chi^2$                                          |
| <code>e(chi2_c)</code>     | $\chi^2$ for comparison test                      |
| <code>e(p_c)</code>        | $p$ -value for comparison test                    |
| <code>e(p)</code>          | significance of comparison test                   |
| <code>e(rho)</code>        | $\rho$                                            |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                         |
| <code>e(rank0)</code>      | rank of <code>e(V)</code> for constant-only model |
| <code>e(ic)</code>         | number of iterations                              |
| <code>e(rc)</code>         | return code                                       |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                       |

*(Continued on next page)*

## Macros

|                                 |                                                                          |
|---------------------------------|--------------------------------------------------------------------------|
| <code>e(cmd)</code>             | heckprob                                                                 |
| <code>e(cmdline)</code>         | command as typed                                                         |
| <code>e(depvar)</code>          | names of dependent variables                                             |
| <code>e(wtype)</code>           | weight type                                                              |
| <code>e(wexp)</code>            | weight expression                                                        |
| <code>e(title)</code>           | title in estimation output                                               |
| <code>e(clustvar)</code>        | name of cluster variable                                                 |
| <code>e(offset1)</code>         | offset for regression equation                                           |
| <code>e(offset2)</code>         | offset for selection equation                                            |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                  |
| <code>e(chi2_ct)</code>         | type of comparison $\chi^2$ test                                         |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                           |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                            |
| <code>e(diparm#)</code>         | display transformed parameter #                                          |
| <code>e(opt)</code>             | type of optimization                                                     |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>       | type of ml method                                                        |
| <code>e(user)</code>            | name of likelihood-evaluator program                                     |
| <code>e(technique)</code>       | maximization technique                                                   |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular              |
| <code>e(crittype)</code>        | optimization criterion                                                   |
| <code>e(properties)</code>      | b V                                                                      |
| <code>e(predict)</code>         | program used to implement predict                                        |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>           |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>           |

## Matrices

|                              |                                                                          |
|------------------------------|--------------------------------------------------------------------------|
| <code>e(b)</code>            | coefficient vector                                                       |
| <code>e(Cns)</code>          | constraints matrix                                                       |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)                                      |
| <code>e(gradient)</code>     | gradient vector                                                          |
| <code>e(V)</code>            | variance–covariance matrix of the estimators                             |
| <code>e(V_modelbased)</code> | model-based variance                                                     |
| <code>e(ml_h)</code>         | derivative tolerance, $(\text{abs}(\mathbf{b})+1\text{e-}3)*1\text{e-}3$ |
| <code>e(ml_scale)</code>     | derivative scale factor                                                  |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

heckprob is implemented as an ado-file. [Van de Ven and Van Pragg \(1981\)](#) provide an introduction and an explanation of this model.

The probit equation is

$$y_j = (\mathbf{x}_j\boldsymbol{\beta} + u_{1j} > 0)$$

The selection equation is

$$\mathbf{z}_j\boldsymbol{\gamma} + u_{2j} > 0$$

where

$$\begin{aligned}u_1 &\sim N(0, 1) \\u_2 &\sim N(0, 1) \\ \text{corr}(u_1, u_2) &= \rho\end{aligned}$$

The log likelihood is

$$\begin{aligned}\ln L = & \sum_{\substack{j \in S \\ y_j \neq 0}} w_j \ln \left\{ \Phi_2 \left( x_j \beta + \text{offset}_j^\beta, z_j \gamma + \text{offset}_j^\gamma, \rho \right) \right\} \\ & + \sum_{\substack{j \in S \\ y_j = 0}} w_j \ln \left\{ \Phi_2 \left( -x_j \beta + \text{offset}_j^\beta, z_j \gamma + \text{offset}_j^\gamma, -\rho \right) \right\} \\ & + \sum_{j \notin S} w_j \ln \left\{ 1 - \Phi \left( z_j \gamma + \text{offset}_j^\gamma \right) \right\}\end{aligned}$$

where  $S$  is the set of observations for which  $y_j$  is observed,  $\Phi_2(\cdot)$  is the cumulative bivariate normal distribution function (with mean  $[0 \ 0]'$ ),  $\Phi(\cdot)$  is the standard cumulative normal, and  $w_j$  is an optional weight for observation  $j$ .

In the maximum likelihood estimation,  $\rho$  is not directly estimated. Directly estimated is  $\text{atanh } \rho$ :

$$\text{atanh } \rho = \frac{1}{2} \ln \left( \frac{1 + \rho}{1 - \rho} \right)$$

From the form of the likelihood, it is clear that if  $\rho = 0$ , the log likelihood for the probit model with sample selection is equal to the sum of the probit model for the outcome  $y$  and the selection model. We can perform a likelihood-ratio test by comparing the likelihood of the full model with the sum of the log likelihoods for the probit and selection models.

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

`heckprob` also supports estimation with survey data. For details on VCEs with survey data, see [\[SVY\] variance estimation](#).

(Continued on next page)

## References

- Baum, C. F. 2006. *An Introduction to Modern Econometrics Using Stata*. College Station, TX: Stata Press.
- Chiburis, R., and M. Lokshin. 2007. Maximum likelihood and two-step estimation of an ordered-probit selection model. *Stata Journal* 7: 167–182.
- De Luca, G. 2008. SNP and SML estimation of univariate and bivariate binary-choice models. *Stata Journal* 8: 190–220.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- Heckman, J. 1979. Sample selection bias as a specification error. *Econometrica* 47: 153–161.
- Miranda, A., and S. Rabe-Hesketh. 2006. Maximum likelihood estimation of endogenous switching and sample selection models for binary, ordinal, and count variables. *Stata Journal* 6: 285–308.
- Pindyck, R. S., and D. L. Rubinfeld. 1998. *Econometric Models and Economic Forecasts*. 4th ed. New York: McGraw–Hill.
- Van de Ven, W. P. M. M., and B. M. S. Van Pragg. 1981. The demand for deductibles in private health insurance: A probit model with sample selection. *Journal of Econometrics* 17: 229–252.

## Also see

- [R] [heckprob postestimation](#) — Postestimation tools for heckprob
- [R] [heckman](#) — Heckman selection model
- [R] [probit](#) — Probit regression
- [R] [treatreg](#) — Treatment-effects model
- [U] [20 Estimation and postestimation commands](#)



Description

The following postestimation commands are available for `heckprob`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

## Syntax for predict

```
predict [type] newvar [if] [in] [, statistic nooffset]
```

```
predict [type] { stub* | newvarreg newvarsel newvarathrho } [if] [in] , scores
```

| statistic | description |
|-----------|-------------|
|-----------|-------------|

|                |                                                                                                                                                  |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| Main           |                                                                                                                                                  |
| <u>pmargin</u> | $\Phi(\mathbf{x}_j\mathbf{b})$ , success probability; the default                                                                                |
| <u>p11</u>     | $\Phi_2(\mathbf{x}_j\mathbf{b}, \mathbf{z}_j\mathbf{g}, \rho)$ , predicted probability $\Pr(y_j^{\text{probit}} = 1, y_j^{\text{select}} = 1)$   |
| <u>p10</u>     | $\Phi_2(\mathbf{x}_j\mathbf{b}, -\mathbf{z}_j\mathbf{g}, -\rho)$ , predicted probability $\Pr(y_j^{\text{probit}} = 1, y_j^{\text{select}} = 0)$ |
| <u>p01</u>     | $\Phi_2(-\mathbf{x}_j\mathbf{b}, \mathbf{z}_j\mathbf{g}, -\rho)$ , predicted probability $\Pr(y_j^{\text{probit}} = 0, y_j^{\text{select}} = 1)$ |
| <u>p00</u>     | $\Phi_2(-\mathbf{x}_j\mathbf{b}, -\mathbf{z}_j\mathbf{g}, \rho)$ , predicted probability $\Pr(y_j^{\text{probit}} = 0, y_j^{\text{select}} = 0)$ |
| <u>psel</u>    | $\Phi(\mathbf{z}_j\mathbf{g})$ , selection probability                                                                                           |
| <u>pcond</u>   | $\Phi_2(\mathbf{x}_j\mathbf{b}, \mathbf{z}_j\mathbf{g}, \rho)/\Phi(\mathbf{z}_j\mathbf{g})$ , probability of success conditional on selection    |
| <u>xb</u>      | linear prediction                                                                                                                                |
| <u>stdp</u>    | standard error of the linear prediction                                                                                                          |
| <u>xbasel</u>  | linear prediction for selection equation                                                                                                         |
| <u>stdpsel</u> | standard error of the linear prediction for selection equation                                                                                   |

where  $\Phi(\cdot)$  is the standard normal distribution function and  $\Phi_2(\cdot)$  is the bivariate normal distribution function.

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

|                |                                                                                                                                                                   |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Main           |                                                                                                                                                                   |
| <u>pmargin</u> | the default, calculates the univariate (marginal) predicted probability of success $\Pr(y_j^{\text{probit}} = 1)$ .                                               |
| <u>p11</u>     | calculates the bivariate predicted probability $\Pr(y_j^{\text{probit}} = 1, y_j^{\text{select}} = 1)$ .                                                          |
| <u>p10</u>     | calculates the bivariate predicted probability $\Pr(y_j^{\text{probit}} = 1, y_j^{\text{select}} = 0)$ .                                                          |
| <u>p01</u>     | calculates the bivariate predicted probability $\Pr(y_j^{\text{probit}} = 0, y_j^{\text{select}} = 1)$ .                                                          |
| <u>p00</u>     | calculates the bivariate predicted probability $\Pr(y_j^{\text{probit}} = 0, y_j^{\text{select}} = 0)$ .                                                          |
| <u>psel</u>    | calculates the univariate (marginal) predicted probability of selection $\Pr(y_j^{\text{select}} = 1)$ .                                                          |
| <u>pcond</u>   | calculates the conditional (on selection) predicted probability of success $\Pr(y_j^{\text{probit}} = 1, y_j^{\text{select}} = 1)/\Pr(y_j^{\text{select}} = 1)$ . |
| <u>xb</u>      | calculates the probit linear prediction $\mathbf{x}_j\mathbf{b}$ .                                                                                                |

`stdp` calculates the standard error of the prediction, which can be thought of as the standard error of the predicted expected value or mean for the observation's covariate pattern. The standard error of the prediction is also referred to as the standard error of the fitted value.

`xbse1` calculates the linear prediction for the selection equation.

`stdpsel` calculates the standard error of the linear prediction for the selection equation.

`nooffset` is relevant only if you specified `offset(varname)` for `heckprob`. It modifies the calculations made by `predict` so that they ignore the offset variable; the linear prediction is treated as  $\mathbf{x}_j\mathbf{b}$  rather than as  $\mathbf{x}_j\mathbf{b} + \text{offset}_j$ .

`scores` calculates equation-level score variables.

The first new variable will contain  $\partial \ln L / \partial (\mathbf{x}_j\boldsymbol{\beta})$ .

The second new variable will contain  $\partial \ln L / \partial (\mathbf{z}_j\boldsymbol{\gamma})$ .

The third new variable will contain  $\partial \ln L / \partial (\text{atanh } \rho)$ .

## Remarks

### ► Example 1

It is instructive to compare the marginal predicted probabilities with the predicted probabilities that we would obtain by ignoring the selection mechanism. To compare the two approaches, we will synthesize data so that we know the “true” predicted probabilities.

First, we need to generate correlated error terms, which we can do using a standard Cholesky decomposition approach. For our example, we will clear any data from memory and then generate errors that have a correlation of 0.5 by using the following commands. We set the seed so that interested readers can type in these same commands and obtain the same results.

```
. clear
. set seed 12309
. set obs 5000
obs was 0, now 5000
. gen c1 = rnormal()
. gen c2 = rnormal()
. matrix P = (1,.5\ .5,1)
. matrix A = cholesky(P)
. local fac1 = A[2,1]
. local fac2 = A[2,2]
. gen u1 = c1
. gen u2 = 'fac1'*c1 + 'fac2'*c2
```

We can check that the errors have the correct correlation by using the `correlate` command. We will also normalize the errors so that they have a standard deviation of one, so we can generate a bivariate probit model with known coefficients. We do that with the following commands:

```
. correlate u1 u2
(obs=5000)

 | u1 u2
-----|-----
 | | |
 u1 1.0000
 u2 0.4943 1.0000

. summarize u1
(output omitted)
. replace u1 = u1/r(sd)
(5000 real changes made)
. summarize u2
(output omitted)
. replace u2 = u2/r(sd)
(5000 real changes made)
. drop c1 c2
. gen x1 = uniform()-.5
. gen x2 = uniform()+1/3
. gen y1s = 0.5 + 4*x1 + u1
. gen y2s = 3 - 3*x2 + .5*x1 + u2
. gen y1 = (y1s>0)
. gen y2 = (y2s>0)
```

We have now created two dependent variables,  $y_1$  and  $y_2$ , which are defined by our specified coefficients. We also included error terms for each equation, and the error terms are correlated. We run `heckprob` to verify that the data have been correctly generated according to the model

$$y_1 = .5 + 4x_1 + u_1$$
$$y_2 = 3 + .5x_1 - 3x_2 + u_2$$

where we assume that  $y_1$  is observed only if  $y_2 = 1$ .

```
. heckprob y1 x1, sel(y2 = x1 x2) nolog
```

|                                    |                |   |        |
|------------------------------------|----------------|---|--------|
| Probit model with sample selection | Number of obs  | = | 5000   |
|                                    | Censored obs   | = | 1837   |
|                                    | Uncensored obs | = | 3163   |
|                                    | Wald chi2(1)   | = | 914.68 |
| Log likelihood = -3536.96          | Prob > chi2    | = | 0.0000 |

|    | Coef.   | Std. Err. | z        | P> z   | [95% Conf. Interval] |           |           |
|----|---------|-----------|----------|--------|----------------------|-----------|-----------|
| y1 | x1      | 4.18558   | .1383951 | 30.24  | 0.000                | 3.91433   | 4.456829  |
|    | _cons   | .5712076  | .048313  | 11.82  | 0.000                | .4765159  | .6658992  |
| y2 | x1      | .55554    | .0717434 | 7.74   | 0.000                | .4149255  | .6961546  |
|    | x2      | -3.073634 | .082955  | -37.05 | 0.000                | -3.236223 | -2.911046 |
|    | _cons   | 3.050659  | .0783512 | 38.94  | 0.000                | 2.897093  | 3.204224  |
|    | /athrho | .5983892  | .0831184 | 7.20   | 0.000                | .4354801  | .7612983  |
|    | rho     | .5359024  | .0592475 |        |                      | .4098909  | .641841   |

LR test of indep. eqns. (rho = 0): chi2(1) = 54.88 Prob > chi2 = 0.0000

Now that we have verified that we have generated data according to a known model, we can obtain and then compare predicted probabilities from the probit model with sample selection and a (usual) probit model.

```
. predict pmarg
(option pmargin assumed; Pr(y1=1))
. probit y1 x1 if y2==1
(output omitted)
. predict phat
(option pr assumed; Pr(y1))
```

Using the (marginal) predicted probabilities from the probit model with sample selection (`pmarg`) and the predicted probabilities from the (usual) probit model (`phat`), we can also generate the “true” predicted probabilities from the synthesized `y1s` variable and then compare the predicted probabilities:

```
. gen ptrue = normal(y1s)
. summarize pmarg ptrue phat
```

| Variable | Obs  | Mean     | Std. Dev. | Min      | Max      |
|----------|------|----------|-----------|----------|----------|
| pmarg    | 5000 | .6246981 | .3258191  | .0641337 | .9961385 |
| ptrue    | 5000 | .603741  | .3510774  | 1.88e-06 | .9999996 |
| phat     | 5000 | .6794313 | .3086507  | .0964017 | .9985179 |

Here we see that ignoring the selection mechanism (comparing the `phat` variable with the true `ptrue` variable) results in predicted probabilities that are much higher than the true values. Looking at the marginal predicted probabilities from the model with sample selection, however, results in more accurate predictions.



## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [heckprob](#) — Probit model with sample selection

[U] [20 Estimation and postestimation commands](#)

# Title

help — Display online help

## Syntax

*Display help information in Viewer*

```
help [command_or_topic_name] [, new name(viewername) marker(markername)]
```

*Display help information in Results window*

```
chelp [command_or_topic_name]
```

## Menu

Help > Stata Command...

## Description

The `help` command displays help information about the specified command or topic.

Stata for Mac, Stata for Unix(GUI), and Stata for Windows:

`help` launches a new Viewer to display help for the specified command or topic. If `help` is not followed by a command or a topic name, Stata launches the Viewer and displays `help contents`, the table of contents for the online help.

Help may be accessed either by selecting **Help > Stata Command...** and filling in the desired command name or by typing `help` followed by a command or topic name.

`chelp` will display help in the Results window.

Stata for Unix(console):

Typing `help` followed by a command name or a topic name will display help on the console.

If `help` is not followed by a command or a topic name, a description of how to use the `help` system is displayed.

Stata for Unix(both GUI and console):

`man` is a synonym for `chelp`.

## Options

`new` specifies that a new Viewer window not be opened for the help topic if a Viewer window is already open. The default is for a new Viewer window to be opened each time `help` is typed so that multiple help files may be viewed at once. `new` causes the help file to be displayed in the topmost open Viewer.

`name(viewername)` specifies that help be displayed in a Viewer window named *viewername*. If the named window already exists, its contents will be replaced. If the named window does not exist, it will be created.

`marker`(*markername*) specifies that the help file be opened to the position of *markername* within the help file.

## Remarks

To obtain help for any Stata command, type `help command` or select **Help > Stata Command...** and fill in *command*.

`help` is best explained by examples.

| To obtain help for ...                        | type                                                                          |
|-----------------------------------------------|-------------------------------------------------------------------------------|
| <code>regress</code>                          | <code>help regress</code>                                                     |
| postestimation tools for <code>regress</code> | <code>help regress postestimation</code> or<br><code>help regress post</code> |
| graph option <code>xlabel()</code>            | <code>help graph xlabel()</code>                                              |
| Stata function <code>strpos()</code>          | <code>help strpos()</code>                                                    |
| Mata function <code>optimize()</code>         | <code>help mata optimize()</code>                                             |

Tips:

- `help` displays a subject table of contents for the online help.
- `help guide` displays a table of contents for basic Stata concepts.
- `help estimation commands` displays an alphabetical listing of all Stata estimation commands.
- `help functions` displays help on Stata functions by category.
- `help mata functions` displays a subject table of contents for Mata's functions.
- `help ts glossary` displays the glossary for the time-series manual, and similarly for the other Stata specialty manuals.

See [\[U\] 4 Stata's help and search facilities](#) for a complete description of how to use `help`.

### □ Technical note

When you type `help topic`, Stata first looks along the adopath for *topic.sthlp*; see [\[U\] 17.5 Where does Stata look for ado-files?](#). □

## Also see

[\[R\] hsearch](#) — Search help files

[\[R\] search](#) — Search Stata documentation

[\[R\] net search](#) — Search the Internet for installable packages

[\[GSM\] 4 Getting help](#)

[\[GSW\] 4 Getting help](#)

[\[GSU\] 4 Getting help](#)

[\[U\] 4 Stata's help and search facilities](#)

## Syntax

```
hetprob depvar [indepvars] [if] [in] [weight] ,
 het(varlist [, offset(varname)]) [options]
```

| <i>options</i>                            | description                                                                                                             |
|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| Model                                     |                                                                                                                         |
| *het( <i>varlist</i> [...])               | independent variables to model the variance and possible offset variable                                                |
| <u>no</u> constant                        | suppress constant term                                                                                                  |
| <u>offset</u> ( <i>varname</i> )          | include <i>varname</i> in model with coefficient constrained to 1                                                       |
| asis                                      | retain perfect predictor variables                                                                                      |
| <u>constraints</u> ( <i>constraints</i> ) | apply specified linear constraints                                                                                      |
| <u>collinear</u>                          | keep collinear variables                                                                                                |
| SE/Robust                                 |                                                                                                                         |
| vce( <i>vcetype</i> )                     | <i>vcetype</i> may be oim, <u>robust</u> , <u>cluster</u> <i>clustvar</i> , opg, <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting                                 |                                                                                                                         |
| <u>level</u> (#)                          | set confidence level; default is level(95)                                                                              |
| <u>noskip</u>                             | perform likelihood-ratio test                                                                                           |
| <u>nolrtest</u>                           | perform Wald test on variance                                                                                           |
| <u>nocnsreport</u>                        | do not display constraints                                                                                              |
| <i>display_options</i>                    | control spacing and display of omitted variables and base and empty cells                                               |
| Maximization                              |                                                                                                                         |
| <i>maximize_options</i>                   | control the maximization process; seldom used                                                                           |
| <sup>†</sup> <u>coeflegend</u>            | display coefficients' legend instead of coefficient table                                                               |

\*het() is required. The full specification is het(*varlist* [ , *offset*(*varname*) ]).

<sup>†</sup>coeflegend does not appear in the dialog box.

*indepvars* and *varlist* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar*, *indepvars*, and *varlist* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

bootstrap, by, jackknife, rolling, statsby, and svy are allowed; see [U] 11.1.10 Prefix commands.

Weights are not allowed with the bootstrap prefix.

vce(), noskip, and weights are not allowed with the svy prefix.

fweights, iweights, and pweights are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.



## Menu

Statistics > Binary outcomes > Heteroskedastic probit regression

## Description

`hetprob` fits a maximum-likelihood heteroskedastic probit model.

See [\[R\] `logistic`](#) for a list of related estimation commands.

## Options

### Model

`het(varlist [ , offset(varname) ])` specifies the independent variables and the offset variable, if there is one, in the variance function. `het()` is required.

`noconstant`, `offset(varname)`; see [\[R\] `estimation options`](#).

`asis` forces the retention of perfect predictor variables and their associated perfectly predicted observations and may produce instabilities in maximization; see [\[R\] `probit`](#).

`constraints(constraints)`, `collinear`; see [\[R\] `estimation options`](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [\[R\] `vce\_option`](#).

### Reporting

`level(#)`; see [\[R\] `estimation options`](#).

`noskip` requests fitting of the constant-only model and calculation of the corresponding likelihood-ratio  $\chi^2$  statistic for testing significance of the full model. By default, a Wald  $\chi^2$  statistic is computed for testing the significance of the full model.

`nolrtest` specifies that a Wald test of whether `lnsigma2 = 0` be performed instead of the LR test.

`nocnsreport`; see [\[R\] `estimation options`](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [\[R\] `estimation options`](#).

### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrntolerance`, `from(init_specs)`; see [\[R\] `maximize`](#). These options are seldom used.

Setting the optimization type to `technique(bhhh)` resets the default `vcetype` to `vce(opg)`.

The following option is available with `hetprob` but is not shown in the dialog box:

`coeflegend`; see [\[R\] `estimation options`](#).

## Remarks

Remarks are presented under the following headings:

[Introduction](#)

[Robust standard errors](#)

## Introduction

`hetprob` fits a maximum-likelihood heteroskedastic probit model, which is a generalization of the probit model. Let  $y_j, j = 1, \dots, N$ , be a binary outcome variable taking on the value 0 (failure) or 1 (success). In the probit model, the probability that  $y_j$  takes on the value 1 is modeled as a nonlinear function of a linear combination of the  $k$  independent variables  $\mathbf{x}_j = (x_{1j}, x_{2j}, \dots, x_{kj})$ ,

$$\Pr(y_j = 1) = \Phi(\mathbf{x}_j \mathbf{b})$$

in which  $\Phi()$  is the cumulative distribution function (CDF) of a standard normal random variable, that is, a normally distributed (Gaussian) random variable with mean 0 and variance 1. The linear combination of the independent variables,  $\mathbf{x}_j \mathbf{b}$ , is commonly called the *index function*, or *index*. Heteroskedastic probit generalizes the probit model by generalizing  $\Phi()$  to a normal CDF with a variance that is no longer fixed at 1 but can vary as a function of the independent variables. `hetprob` models the variance as a multiplicative function of these  $m$  variables  $\mathbf{z}_j = (z_{1j}, z_{2j}, \dots, z_{mj})$ , following [Harvey \(1976\)](#):

$$\sigma_j^2 = \{\exp(\mathbf{z}_j \boldsymbol{\gamma})\}^2$$

Thus the probability of success as a function of all the independent variables is

$$\Pr(y_j = 1) = \Phi\left\{\mathbf{x}_j \mathbf{b} / \exp(\mathbf{z}_j \boldsymbol{\gamma})\right\}$$

From this expression, it is clear that, unlike the index  $\mathbf{x}_j \mathbf{b}$ , no constant term can be present in  $\mathbf{z}_j \boldsymbol{\gamma}$  if the model is to be identifiable.

Suppose that the binary outcomes  $y_j$  are generated by thresholding an unobserved random variable,  $w$ , which is normally distributed with mean  $\mathbf{x}_j \mathbf{b}$  and variance 1 such that

$$y_j = \begin{cases} 1 & \text{if } w_j > 0 \\ 0 & \text{if } w_j \leq 0 \end{cases}$$

This process gives the probit model:

$$\Pr(y_j = 1) = \Pr(w_j > 0) = \Phi(\mathbf{x}_j \mathbf{b})$$

Now suppose that the unobserved  $w_j$  are heteroskedastic with variance

$$\sigma_j^2 = \{\exp(\mathbf{z}_j \boldsymbol{\gamma})\}^2$$

Relaxing the homoskedastic assumption of the probit model in this manner yields our multiplicative heteroskedastic probit model:

$$\Pr(y_j = 1) = \Phi\left\{\mathbf{x}_j \mathbf{b} / \exp(\mathbf{z}_j \boldsymbol{\gamma})\right\}$$

## ➤ Example 1

For this example, we generate simulated data for a simple heteroskedastic probit model and then estimate the coefficients with `hetprob`:

```
. set obs 1000
obs was 0, now 1000
. set seed 1234567
. gen x = 1-2*runiform()
. gen xhet = runiform()
. gen sigma = exp(1.5*xhet)
. gen p = normal((0.3+2*x)/sigma)
. gen y = cond(runiform()<=p,1,0)
. hetprob y x, het(xhet)
```

Fitting probit model:

```
Iteration 0: log likelihood = -688.53208
Iteration 1: log likelihood = -591.59895
Iteration 2: log likelihood = -591.50674
Iteration 3: log likelihood = -591.50674
```

Fitting full model:

```
Iteration 0: log likelihood = -591.50674
Iteration 1: log likelihood = -572.12219
Iteration 2: log likelihood = -570.7742
Iteration 3: log likelihood = -569.48921
Iteration 4: log likelihood = -569.47828
Iteration 5: log likelihood = -569.47827
```

Heteroskedastic probit model

```
Number of obs = 1000
Zero outcomes = 452
Nonzero outcomes = 548
Wald chi2(1) = 78.66
Prob > chi2 = 0.0000
```

Log likelihood = -569.4783

|          | y     | Coef.    | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|----------|-------|----------|-----------|------|-------|----------------------|----------|
| y        | x     | 2.228031 | .2512073  | 8.87 | 0.000 | 1.735673             | 2.720388 |
|          | _cons | .2493822 | .0862833  | 2.89 | 0.004 | .08027               | .4184943 |
| lnsigma2 |       |          |           |      |       |                      |          |
|          | xhet  | 1.602537 | .2640131  | 6.07 | 0.000 | 1.085081             | 2.119993 |

Likelihood-ratio test of lnsigma2=0: chi2(1) = 44.06 Prob > chi2 = 0.0000

Above we created two variables, `x` and `xhet`, and then simulated the model

$$\Pr(y = 1) = F\left\{(\beta_0 + \beta_1 x) / \exp(\gamma_1 xhet)\right\}$$

for  $\beta_0 = 0.3$ ,  $\beta_1 = 2$ , and  $\gamma_1 = 1.5$ . According to `hetprob`'s output, all coefficients are significant, and, as we would expect, the Wald test of the full model versus the constant-only model—e.g., the index consisting of  $\beta_0 + \beta_1 x$  versus that of just  $\beta_0$ —is significant with  $\chi^2(1) = 79$ . Likewise, the likelihood-ratio test of heteroskedasticity, which tests the full model with heteroskedasticity against the full model without, is significant with  $\chi^2(1) = 44$ . See [R] [maximize](#) for more explanation of the output. For this simple model, `hetprob` took five iterations to converge. As stated elsewhere (Greene 2008, 789), this is a difficult model to fit, and it is not uncommon for it to require many iterations or for the optimizer to print out warnings and informative messages during the optimization. Slow convergence is especially common for models in which one or more of the independent variables appear in both the index and variance functions.

□ Technical note

Stata interprets a value of 0 as a negative outcome (failure) and treats all other values (except missing) as positive outcomes (successes). Thus if your dependent variable takes on the values 0 and 1, then 0 is interpreted as failure and 1 as success. If your dependent variable takes on the values 0, 1, and 2, then 0 is still interpreted as failure, but both 1 and 2 are treated as successes.

Robust standard errors

If you specify the `vce(robust)` option, `hetprob` reports robust standard errors as described in [U] 20.16 Obtaining robust variance estimates. To illustrate the effect of this option, we will reestimate our coefficients by using the same model and data in our example, this time adding `vce(robust)` to our `hetprob` command.

▷ Example 2

```
. hetprob y x, het(xhet) vce(robust) nolog
Heteroskedastic probit model Number of obs = 1000
 Zero outcomes = 452
 Nonzero outcomes = 548
 Wald chi2(1) = 65.23
 Prob > chi2 = 0.0000

Log pseudolikelihood = -569.4783
```

| y        | Coef.    | Robust Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|----------|----------|------------------|------|-------|----------------------|----------|
| y        |          |                  |      |       |                      |          |
| x        | 2.22803  | .2758597         | 8.08 | 0.000 | 1.687355             | 2.768705 |
| _cons    | .2493821 | .0843367         | 2.96 | 0.003 | .0840853             | .4146789 |
| lnsigma2 |          |                  |      |       |                      |          |
| xhet     | 1.602537 | .2671326         | 6.00 | 0.000 | 1.078967             | 2.126107 |

```
Wald test of lnsigma2=0: chi2(1) = 35.99 Prob > chi2 = 0.0000
```

The `vce(robust)` standard errors for two of the three parameters are larger than the previously reported conventional standard errors. This is to be expected, even though (by construction) we have perfect model specification because this option trades off efficient estimation of the coefficient variance–covariance matrix for robustness against misspecification.

Specifying the `vce(cluster clustvar)` option relaxes the usual assumption of independence between observations to the weaker assumption of independence just between clusters; that is, `hetprob, vce(cluster clustvar)` is robust with respect to within-cluster correlation. This option is less efficient than the `xtgee` population-averaged models because `hetprob` inefficiently sums within cluster for the standard-error calculation rather than attempting to exploit what might be assumed about the within-cluster correlation.

## Saved results

hetprob saves the following in `e()`:

### Scalars

|                            |                                                   |
|----------------------------|---------------------------------------------------|
| <code>e(N)</code>          | number of observations                            |
| <code>e(N_f)</code>        | number of zero outcomes                           |
| <code>e(N_s)</code>        | number of nonzero outcomes                        |
| <code>e(k)</code>          | number of parameters                              |
| <code>e(k_eq)</code>       | number of equations                               |
| <code>e(k_eq_model)</code> | number of equations in model Wald test            |
| <code>e(k_dv)</code>       | number of dependent variables                     |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints    |
| <code>e(df_m)</code>       | model degrees of freedom                          |
| <code>e(ll)</code>         | log likelihood                                    |
| <code>e(ll_0)</code>       | log likelihood, constant-only model               |
| <code>e(ll_c)</code>       | log likelihood, comparison model                  |
| <code>e(N_clust)</code>    | number of clusters                                |
| <code>e(chi2)</code>       | $\chi^2$                                          |
| <code>e(chi2_c)</code>     | $\chi^2$ for heteroskedasticity LR test           |
| <code>e(p_c)</code>        | $p$ -value for heteroskedasticity LR test         |
| <code>e(df_m_c)</code>     | degrees of freedom for heteroskedasticity LR test |
| <code>e(p)</code>          | significance                                      |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                         |
| <code>e(rank0)</code>      | rank of <code>e(V)</code> for constant-only model |
| <code>e(ic)</code>         | number of iterations                              |
| <code>e(rc)</code>         | return code                                       |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                       |

*(Continued on next page)*

**Macros**

|                                 |                                                                                 |
|---------------------------------|---------------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>hetprob</code>                                                            |
| <code>e(cmdline)</code>         | command as typed                                                                |
| <code>e(depvar)</code>          | name of dependent variable                                                      |
| <code>e(wtype)</code>           | weight type                                                                     |
| <code>e(wexp)</code>            | weight expression                                                               |
| <code>e(title)</code>           | title in estimation output                                                      |
| <code>e(clustvar)</code>        | name of cluster variable                                                        |
| <code>e(offset1)</code>         | offset for probit equation                                                      |
| <code>e(offset2)</code>         | offset for variance equation                                                    |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                         |
| <code>e(chi2_ct)</code>         | Wald or LR; type of model $\chi^2$ test corresponding to <code>e(chi2_c)</code> |
| <code>e(vce)</code>             | <i>vctype</i> specified in <code>vce()</code>                                   |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                                   |
| <code>e(opt)</code>             | type of optimization                                                            |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization        |
| <code>e(method)</code>          | requested estimation method                                                     |
| <code>e(ml_method)</code>       | type of ml method                                                               |
| <code>e(user)</code>            | name of likelihood-evaluator                                                    |
| <code>e(technique)</code>       | maximization technique                                                          |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular                     |
| <code>e(crittype)</code>        | optimization criterion                                                          |
| <code>e(properties)</code>      | b V                                                                             |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                                  |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>                  |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>                  |

**Matrices**

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(gradient)</code>     | gradient vector                              |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

**Functions**

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`hetprob` is implemented as an ado-file.

The heteroskedastic probit model is a generalization of the probit model because it allows the scale of the inverse link function to vary from observation to observation as a function of the independent variables.

The log-likelihood function for the heteroskedastic probit model is

$$\ln L = \sum_{j \in S} w_j \ln \Phi\{\mathbf{x}_j \boldsymbol{\beta} / \exp(\mathbf{z}_j \boldsymbol{\gamma})\} + \sum_{j \notin S} w_j \ln [1 - \Phi\{\mathbf{x}_j \boldsymbol{\beta} / \exp(\mathbf{z}_j \boldsymbol{\gamma})\}]$$

where  $S$  is the set of all observations  $j$  such that  $y_j \neq 0$  and  $w_j$  denotes the optional weights.  $\ln L$  is maximized as described in [R] [maximize](#).

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in *Maximum likelihood estimators* and *Methods and formulas*.

`hetprob` also supports estimation with survey data. For details on VCEs with survey data, see [SVY] [variance estimation](#).

## References

Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.

Harvey, A. C. 1976. Estimating regression models with multiplicative heteroscedasticity. *Econometrica* 44: 461–465.

## Also see

[R] [hetprob postestimation](#) — Postestimation tools for `hetprob`

[R] [logistic](#) — Logistic regression, reporting odds ratios

[R] [probit](#) — Probit regression

[SVY] [svy estimation](#) — Estimation commands for survey data

[XT] [xtprobit](#) — Random-effects and population-averaged probit models

[U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `hetprob`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>            | link test for model specification                                                                   |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic nooffset]

predict [type] { stub* | newvarreg newvarlnsigma2 } [if] [in] , scores
```

| statistic          | description                                    |
|--------------------|------------------------------------------------|
| Main               |                                                |
| <code>pr</code>    | probability of a positive outcome; the default |
| <code>xb</code>    | linear prediction                              |
| <code>sigma</code> | standard deviation of the error term           |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.



## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

### Main

**pr**, the default, calculates the probability of a positive outcome.

**xb** calculates the linear prediction.

**sigma** calculates the standard deviation of the error term.

**nooffset** is relevant only if you specified **offset** (*varname*) for **hetprob**. It modifies the calculations made by **predict** so that they ignore the offset variable; the linear prediction is treated as  $\mathbf{x}_j \mathbf{b}$  rather than as  $\mathbf{x}_j \mathbf{b} + \text{offset}_j$ .

**scores** calculates equation-level score variables.

The first new variable will contain  $\partial \ln L / \partial (\mathbf{x}_j \boldsymbol{\beta})$ .

The second new variable will contain  $\partial \ln L / \partial (\mathbf{z}_j \boldsymbol{\gamma})$ .

## Remarks

Once you have fit a model, you can use the **predict** command to obtain the predicted probabilities for both the estimation sample and other samples; see [U] 20 [Estimation and postestimation commands](#) and [R] [predict](#). **predict** without arguments calculates the predicted probability of a positive outcome. With the **xb** option, **predict** calculates the index function combination,  $\mathbf{x}_j \mathbf{b}$ , where  $\mathbf{x}_j$  are the independent variables in the  $j$ th observation and  $\mathbf{b}$  is the estimated parameter vector. With the **sigma** option, **predict** calculates the predicted standard deviation,  $\sigma_j = \exp(\mathbf{z}_j \boldsymbol{\gamma})$ .

### ► Example 1

We use **predict** to compute the predicted probabilities and standard deviations based on the model in [example 2](#) in [R] [hetprob](#) to compare these with the actual values:

```
. predict phat
(option pr assumed; Pr(y))
. gen diff_p = phat - p
. summarize diff_p
```

| Variable | Obs  | Mean      | Std. Dev. | Min       | Max     |
|----------|------|-----------|-----------|-----------|---------|
| diff_p   | 1000 | -.0107081 | .0131869  | -.0466331 | .010482 |

```
. predict sigmahat, sigma
. gen diff_s = sigmahat - sigma
. summarize diff_s
```

| Variable | Obs  | Mean     | Std. Dev. | Min      | Max      |
|----------|------|----------|-----------|----------|----------|
| diff_s   | 1000 | .1558881 | .1363698  | .0000417 | .4819107 |

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [hetprob](#) — Heteroskedastic probit model

[U] [20 Estimation and postestimation commands](#)

Syntax

`histogram varname [if] [in] [weight] [ , [continuous_opts | discrete_opts] options ]`

| <i>continuous_opts</i> | description                       |
|------------------------|-----------------------------------|
| Main                   |                                   |
| <code>bin(#)</code>    | set number of bins to #           |
| <code>width(#)</code>  | set width of bins to #            |
| <code>start(#)</code>  | set lower limit of first bin to # |

| <i>discrete_opts</i>  | description                        |
|-----------------------|------------------------------------|
| Main                  |                                    |
| <code>discrete</code> | specify that data are discrete     |
| <code>width(#)</code> | set width of bins to #             |
| <code>start(#)</code> | set theoretical minimum value to # |

| <i>options</i>                                | description                  |
|-----------------------------------------------|------------------------------|
| Main                                          |                              |
| <code>density</code>                          | draw as density; the default |
| <code>fraction</code>                         | draw as fractions            |
| <code>frequency</code>                        | draw as frequencies          |
| <code>percent</code>                          | draw as percentages          |
| <i>bar_options</i>                            | rendition of bars            |
| <code>addlabels</code>                        | add height labels to bars    |
| <code>addlabopts(marker_label_options)</code> | affect rendition of labels   |

|                                         |                                            |
|-----------------------------------------|--------------------------------------------|
| Density plots                           |                                            |
| <code>normal</code>                     | add a normal density to the graph          |
| <code>normopts(line_options)</code>     | affect rendition of normal density         |
| <code>kdensity</code>                   | add a kernel density estimate to the graph |
| <code>kdenopts(kdensity_options)</code> | affect rendition of kernel density         |

|                            |                                  |
|----------------------------|----------------------------------|
| Add plots                  |                                  |
| <code>addplot(plot)</code> | add other plots to the histogram |

|                                             |                                                     |
|---------------------------------------------|-----------------------------------------------------|
| Y axis, X axis, Titles, Legend, Overall, By |                                                     |
| <i>twoway_options</i>                       | any options documented in [G] <i>twoway_options</i> |

See [G] [graph twoway line](#), [G] [graph twoway kdensity](#), [G] [marker\\_label\\_options](#), [G] [graph twoway bar](#), [G] [twoway\\_options](#), and [G] [addplot\\_option](#).  
fweights are allowed; see [U] [11.1.6 weight](#).

## Menu

Graphics > Histogram

## Description

`histogram` draws histograms of *varname*, which is assumed to be the name of a continuous variable unless the `discrete` option is specified.

## Options for use in the continuous case

Main

`bin(#)` and `width(#)` are alternatives. They specify how the data are to be aggregated into bins: `bin()` by specifying the number of bins (from which the width can be derived) and `width()` by specifying the bin width (from which the number of bins can be derived).

If neither option is specified, results are the same as if `bin(k)` had been specified, where

$$k = \min\{\sqrt{N}, 10 \ln(N)/\ln(10)\}$$

and where *N* is the (weighted) number of observations.

`start(#)` specifies the theoretical minimum of *varname*. The default is `start(m)`, where *m* is the observed minimum value of *varname*.

Specify `start()` when you are concerned about sparse data, for instance, if you know that *varname* can have a value of 0, but you are concerned that 0 may not be observed.

`start(#)`, if specified, must be less than or equal to *m*, or else an error will be issued.

## Options for use in the discrete case

Main

`discrete` specifies that *varname* is discrete and that you want each unique value of *varname* to have its own bin (bar of histogram).

`width(#)` is rarely specified in the discrete case; it specifies the width of the bins. The default is `width(d)`, where *d* is the observed minimum difference between the unique values of *varname*.

Specify `width()` if you are concerned that your data are sparse. For example, in theory *varname* could take on the values, say, 1, 2, 3, ..., 9, but because of the sparseness, perhaps only the values 2, 4, 7, and 8 are observed. Here the default width calculation would produce `width(2)`, and you would want to specify `width(1)`.

`start(#)` is also rarely specified in the discrete case; it specifies the theoretical minimum value of *varname*. The default is `start(m)`, where *m* is the observed minimum value.

As with `width()`, specify `start(#)` if you are concerned that your data are sparse. In the previous example, you might also want to specify `start(1)`. `start()` does nothing more than add white space to the left side of the graph.

The value of # in `start()` must be less than or equal to *m*, or an error will be issued.

## Options for use in both the continuous and discrete cases

### Main

`density`, `fraction`, `frequency`, and `percent` specify whether you want the histogram scaled to density units, fractional units, frequencies, or percentages. `density` is the default.

`density` scales the height of the bars so that the sum of their areas equals 1.

`fraction` scales the height of the bars so that the sum of their heights equals 1.

`frequency` scales the height of the bars so that each bar's height is equal to the number of observations in the category. Thus the sum of the heights is equal to the total number of observations.

`percent` scales the height of the bars so that the sum of their heights equals 100.

`bar_options` are any of the labels allowed by `graph twoway bar`; see [G] [graph twoway bar](#).

One of the most useful `bar_options` is `barwidth(#)`, which specifies the width of the bars in `varname` units. By default, `histogram` draws the bars so that adjacent bars just touch. If you want gaps between the bars, do not specify `histogram`'s `width()` option—which would change how the histogram is calculated—but specify the `bar_option` `barwidth()` or the `histogram` option `gap`, both of which affect only how the bar is rendered.

The `bar_option` `horizontal` cannot be used with the `addlabels` option.

`addlabels` specifies that the top of each bar be labeled with the density, fraction, or frequency, as determined by the `density`, `fraction`, and `frequency` options.

`addlabopts(marker_label_options)` specifies how to render the labels atop the bars. See [G] [marker\\_label\\_options](#). Do not specify the `marker_label_option` `mlabel(varname)`, which specifies the variable to be used; this is specified for you by `histogram`.

`addlabopts()` will accept more options than those documented in [G] [marker\\_label\\_options](#). All options allowed by `twoway scatter` are also allowed by `addlabopts()`. One particularly useful option is `yvarformat()`; see [G] [advanced\\_options](#).

### Density plots

`normal` specifies that the histogram be overlaid with an appropriately scaled normal density. The normal will have the same mean and standard deviation as the data.

`normopts(line_options)` specifies details about the rendition of the normal curve, such as the color and style of line used. See [G] [graph twoway line](#).

`kdensity` specifies that the histogram be overlaid with an appropriately scaled kernel density estimate of the density. By default, the estimate will be produced using the Epanechnikov kernel with an “optimal” half-width. This default corresponds to the default of `kdensity`; see [R] [kdensity](#). How the estimate is produced can be controlled using the `kdensityopts()` option described below.

`kdensityopts(kdensity_options)` specifies details about how the kernel density estimate is to be produced along with details about the rendition of the resulting curve, such as the color and style of line used. The kernel density estimate is described in [G] [graph twoway kdensity](#). As an example, if you wanted to produce kernel density estimates by using the Gaussian kernel with optimal half-width, you would specify `kdensityopts(gauss)`, and if you also wanted a half-width of 5, you would specify `kdensityopts(gauss width(5))`.

## Add plots

`addplot(plot)` allows adding more `graph twoway` plots to the graph; see help [\[G\] addplot\\_option](#).

## Y axis, X axis, Titles, Legend, Overall, By

`twoway_options` are any of the options documented in [\[G\] twoway\\_options](#). This includes, most importantly, options for titling the graph (see [\[G\] title\\_options](#)), options for saving the graph to disk (see [\[G\] saving\\_option](#)), and the `by()` option, which will allow you to simultaneously graph histograms for different subsets of the data (see [\[G\] by\\_option](#)).

## Remarks

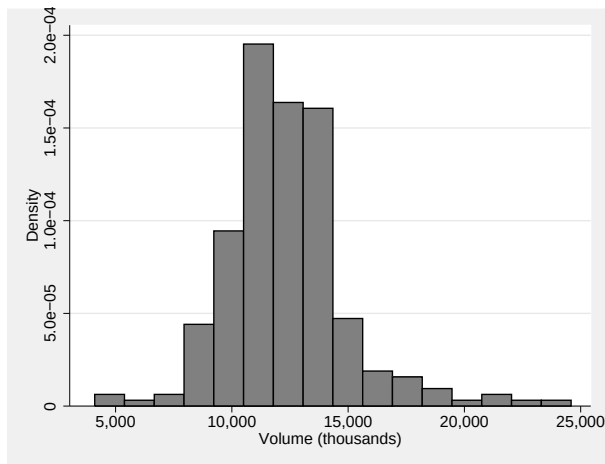
Remarks are presented under the following headings:

[Histograms of continuous variables](#)  
[Overlaying normal and kernel density estimates](#)  
[Histograms of discrete variables](#)  
[Use with `by\(\)`](#)

## Histograms of continuous variables

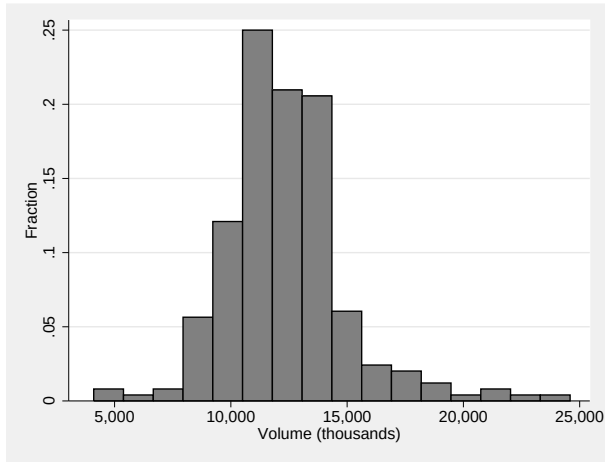
`histogram` assumes that the variable is continuous, so you need type only `histogram` followed by the variable name:

```
. use http://www.stata-press.com/data/r11/sp500
(S&P 500)
. histogram volume
(bin=15, start=4103, width=1280.3533)
```



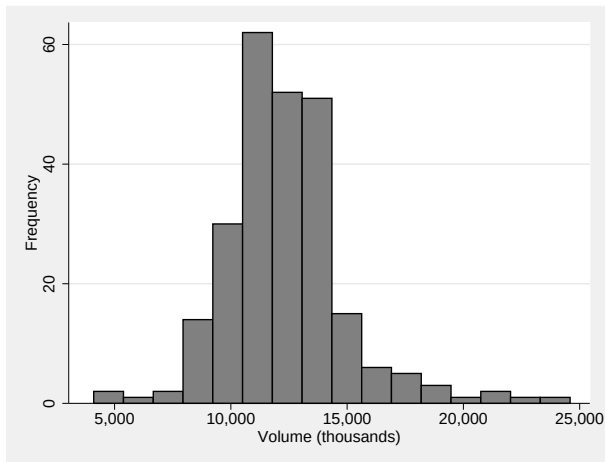
The small values reported for density on the *y* axis are correct; if you added up the area of the bars, you would get 1. Nevertheless, many people are used to seeing histograms scaled so that the bar heights sum to 1,

```
. histogram volume, fraction
(bin=15, start=4103, width=1280.3533)
```



and others are used to seeing histograms so that the bar height reflects the number of observations,

```
. histogram volume, frequency
(bin=15, start=4103, width=1280.3533)
```

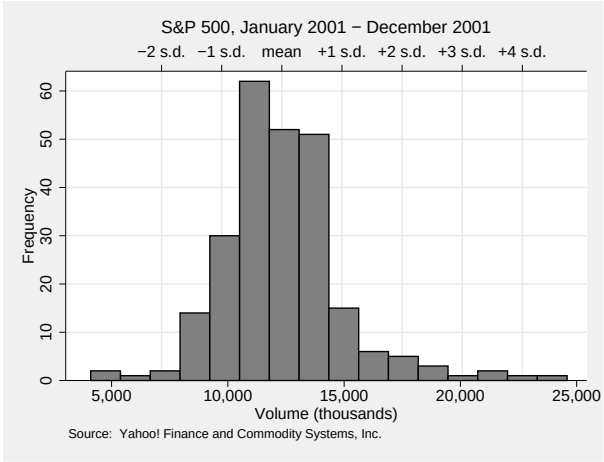


Regardless of the scale you prefer, you can specify other options to make the graph look more impressive:

*(Continued on next page)*

```
. summarize volume
Variable | Obs Mean Std. Dev. Min Max
-----+-----
volume | 248 12320.68 2585.929 4103 23308.3

. histogram volume, freq
> xaxis(1 2)
> ylabel(0(10)60, grid)
> xlabel(12321 "mean"
> 9735 "-1 s.d."
> 14907 "+1 s.d."
> 7149 "-2 s.d."
> 17493 "+2 s.d."
> 20078 "+3 s.d."
> 22664 "+4 s.d."
>
> , axis(2) grid gmax)
> xtitle("", axis(2))
> subtitle("S&P 500, January 2001 - December 2001")
> note("Source: Yahoo! Finance and Commodity Systems, Inc.")
(bin=15, start=4103, width=1280.3533)
```



For an explanation of the `xaxis()` option—it created the upper and lower  $x$  axis—see [G] [axis\\_choice\\_options](#). For an explanation of the `ylabel()` and `xlabel()` options, see [G] [axis\\_label\\_options](#). For an explanation of the `subtitle()` and `note()` options, see [G] [title\\_options](#).

### Overlaying normal and kernel density estimates

Specifying `normal` will overlay a normal density over the histogram. It would be enough to type

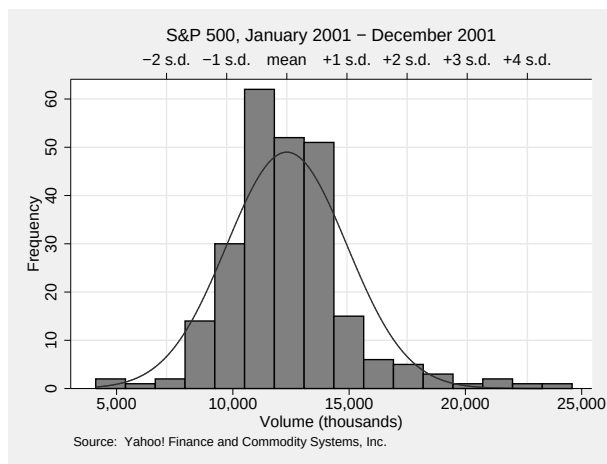
```
. histogram volume, normal
```

but we will add the option to our more impressive rendition:

```
. summarize volume
Variable | Obs Mean Std. Dev. Min Max
-----+-----
volume | 248 12320.68 2585.929 4103 23308.3
```



```
. histogram volume, freq normal
> xaxis(1 2)
> ylabel(0(10)60, grid)
> xlabel(12321 "mean"
> 9735 "-1 s.d."
> 14907 "+1 s.d."
> 7149 "-2 s.d."
> 17493 "+2 s.d."
> 20078 "+3 s.d."
> 22664 "+4 s.d."
> , axis(2) grid gmax)
> xtitle("", axis(2))
> subtitle("S&P 500, January 2001 - December 2001")
> note("Source: Yahoo! Finance and Commodity Systems, Inc.")
(bin=15, start=4103, width=1280.3533)
```



If we instead wanted to overlay a kernel density estimate, we could specify `kdensity` in place of `normal`.

## Histograms of discrete variables

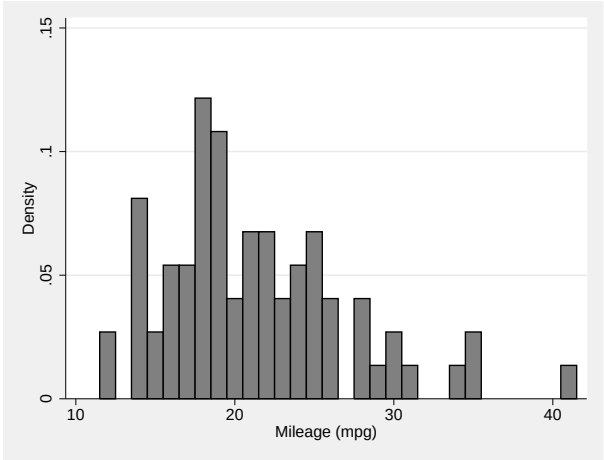
Specify `histogram`'s discrete option when you wish to treat the data as discrete—when you wish each unique value of the variable to be assigned its own bin. For instance, in the automobile data, `mpg` is a continuous variable, but the mileage ratings have been measured to integer precision. If we were to type

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. histogram mpg
(bin=8, start=12, width=3.625)
```

`mpg` would be treated as continuous and categorized into eight bins by the default number-of-bins calculation, which is based on the number of observations, 74.

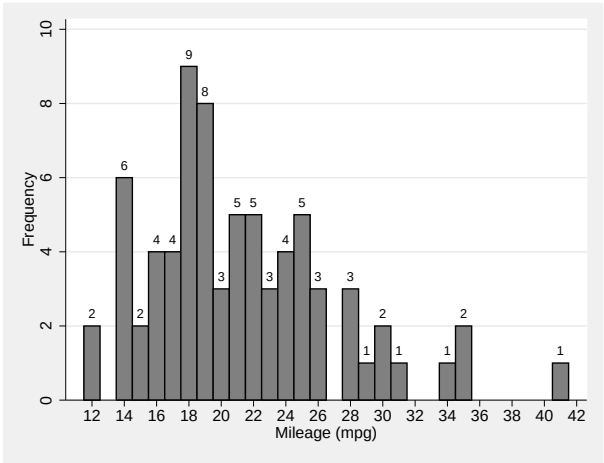
Adding the `discrete` option makes a histogram with a bin for each of the 21 unique values.

```
. histogram mpg, discrete
(start=12, width=1)
```



Just as in the continuous case, the  $y$  axis was reported in density, and we could specify the `fraction` or `frequency` options if we wanted it to be reported differently. Below we specify `frequency`, we specify `addlabels` to add a report of frequencies printed above the bars, we specify `ylabel(,grid)` to add horizontal grid lines, and we specify `xlabel(12(2)42)` to label the values 12, 14, ..., 42 on the  $x$  axis:

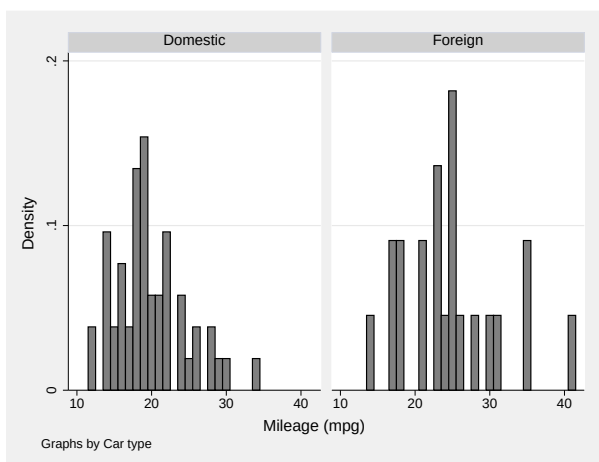
```
. histogram mpg, discrete freq addlabels ylabel(,grid) xlabel(12(2)42)
(start=12, width=1)
```



## Use with by()

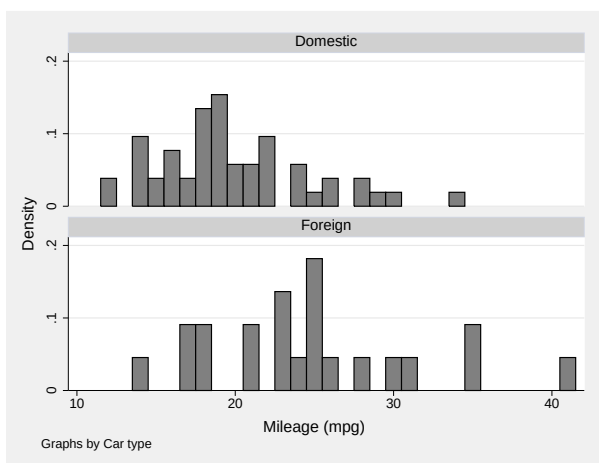
histogram may be used with graph twoway's by(); for example,

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. histogram mpg, discrete by(foreign)
```



Here results would be easier to compare if the graphs were presented in one column:

```
. histogram mpg, discrete by(foreign, col(1))
```



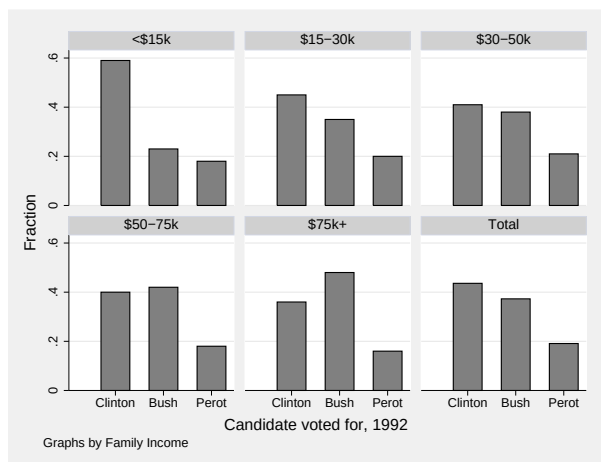
col(1) is a by() suboption—see [G] [by\\_option](#)—and there are other useful suboptions, such as total, which will add an overall total histogram. total is a suboption of by(), not an option of histogram, so you would type

```
. histogram mpg, discrete by(foreign, total)
```

and not histogram mpg, discrete by(foreign) total.

As another example, [Lipset \(1993\)](#) reprinted data from the *New York Times* (November 5, 1992) collected by the Voter Research and Surveys based on questionnaires completed by 15,490 U.S. presidential voters from 300 polling places on election day in 1992.

```
. use http://www.stata-press.com/data/r11/voter
. histogram candi [freq=pop], discrete fraction by(inc, total)
> gap(40) xlabel(2 3 4, valuelabel)
```



We specified `gap(40)` to reduce the width of the bars by 40%. We also used `xlabel()`'s `valuelabel` suboption, which caused our bars to be labeled “Clinton”, “Bush”, and “Perot”, rather than 2, 3, and 4; see [\[G\] \*axis\\_label\\_options\*](#).

## Methods and formulas

`histogram` is implemented as an ado-file.

## References

- Cox, N. J. 2004. [Speaking Stata: Graphing distributions](#). *Stata Journal* 4: 66–88.
- . 2005. [Speaking Stata: Density probability plots](#). *Stata Journal* 5: 259–273.
- Harrison, D. A. 2005. [Stata tip 20: Generating histogram bin variables](#). *Stata Journal* 5: 280–281.
- Lipset, S. M. 1993. The significance of the 1992 election. *PS: Political Science and Politics* 26: 7–16.

## Also see

- [\[R\] \*kdensity\*](#) — Univariate kernel density estimation
- [\[R\] \*spikeplot\*](#) — Spike plots and rootograms
- [\[G\] \*graph twoway histogram\*](#) — Histogram plots

## Syntax

`hsearch word(s)`

`hsearch word(s), build`

`hsearch, build`

## Description

`hsearch word(s)` searches the help files for *word(s)* and presents a clickable list in the Viewer.

`hsearch word(s), build` does the same thing but builds a new index first.

`hsearch, build` rebuilds the index but performs no search.

## Option

`build` forces the index that `hsearch` uses to be built or rebuilt.

The index is automatically built the first time you use `hsearch`, and it is automatically rebuilt if you have recently installed an ado-file update by using `update`; see [\[R\] update](#). Thus the `build` option is rarely specified.

You should specify `build` if you have recently installed user-written ado-files by using `net install` (see [\[R\] net](#)) or `ssc` (see [\[R\] ssc](#)), or if you have recently updated any of your own help files.

## Remarks

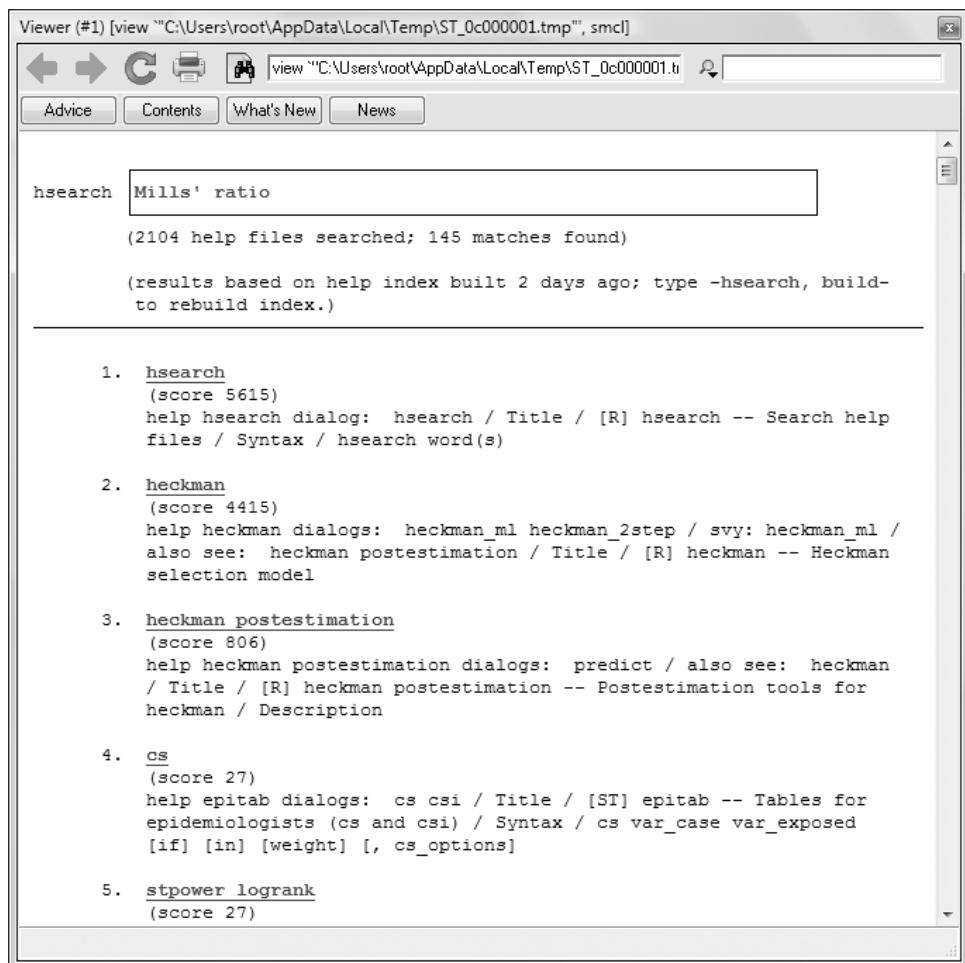
Remarks are presented under the following headings:

[Using hsearch](#)  
[Alternatives to hsearch](#)  
[Recommendations](#)  
[How hsearch works](#)

## Using hsearch

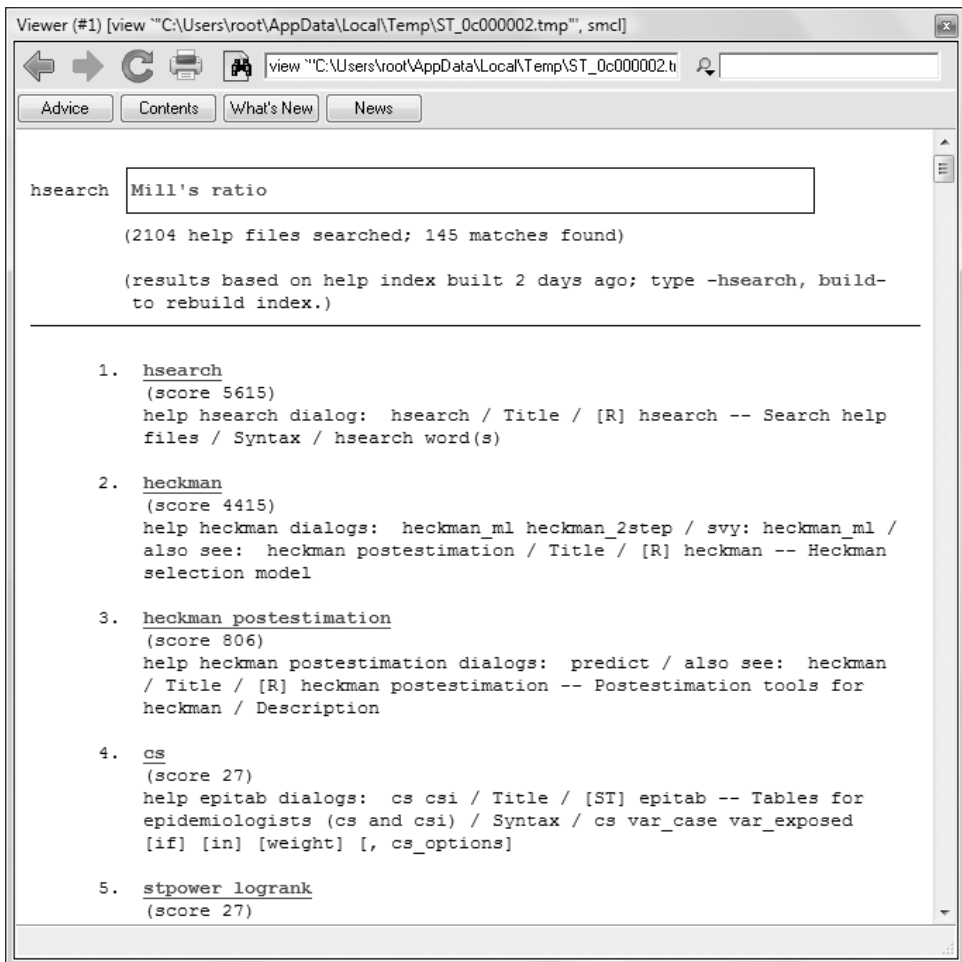
You use `hsearch` to find help for commands and features installed on your computer. If you wanted to find commands related to Mills' ratio, you would type

```
. hsearch Mills' ratio
```



You could just as well type

```
. hsearch Mill's ratio
```



or type any of

- . hsearch Mills ratio
- . hsearch mills ratio

or even

- . hsearch ratio mills

because word order, capitalization, and punctuation do not matter.

## Alternatives to hsearch

Alternatives to hsearch are search and findit:

- . search mills ratio
- . findit mills ratio

search, like hsearch, searches commands already installed on your computer. search searches the keywords; hsearch searches the help files themselves. Hence, hsearch usually finds everything that search finds and more. The fewer things that search finds should be more relevant.

`findit` searches keywords just as `search` does, but `findit` searches the web as well as your computer and so may find commands that you might wish to install.

## Recommendations

- In general, `hsearch` is better than `search`. `hsearch` finds more and better organizes the list of what it finds.
- When you know that Stata can do what you are looking for but you cannot remember the command name or when you know that you installed a relevant user-written package, use `hsearch`.
- When you are unsure whether Stata can do a certain task, use `hsearch` first and then use `findit`.

## How hsearch works

`hsearch` searches the `.sthlp` files.

Finding all those files and then looking through them would take a long time if `hsearch` did that every time you used it. Instead, `hsearch` builds an index of the `.sthlp` files and then searches that.

That file is called `sthlpindex.idx` and is stored in your `PERSONAL` directory.

Every so often, `hsearch` automatically rebuilds the index so that it accurately reflects what is installed on your computer. You can force `hsearch` to rebuild the index at any time by typing

```
. hsearch, build
```

## Methods and formulas

`hsearch` is implemented as an ado-file.

## Also see

[R] [net search](#) — Search the Internet for installable packages

[R] [search](#) — Search Stata documentation



Remarks

Stata does not have commands for inequality measures, except `roctab` has an option to report Gini and Pietra indices; see [R] `roc`. Stata users, however, have developed an excellent suite of commands, many of which have been published in the *Stata Journal* (SJ) and in the *Stata Technical Bulletin* (STB).

| issue  | insert                   | author(s)                       | command                                                                                                                                               | description                                                                     |
|--------|--------------------------|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------|
| STB-48 | <a href="#">gr35</a>     | N. J. Cox                       | <code>psm</code> , <code>qsm</code> ,<br><code>pdagum</code> ,<br><code>qdagum</code>                                                                 | Diagnostic plots for assessing Singh–Maddala and Dagum distributions fit by MLE |
| STB-23 | <a href="#">sg31</a>     | R. Goldstein                    | <code>rspread</code>                                                                                                                                  | Measures of diversity: Absolute and relative                                    |
| STB-48 | <a href="#">sg104</a>    | S. P. Jenkins                   | <code>sumdist</code> ,<br><code>xfrac</code> ,<br><code>ineqdeco</code> ,<br><code>geivars</code> ,<br><code>ineqfac</code> ,<br><code>povdeco</code> | Analysis of income distributions                                                |
| STB-48 | <a href="#">sg106</a>    | S. P. Jenkins                   | <code>smfit</code> ,<br><code>dagumfit</code>                                                                                                         | Fitting Singh–Maddala and Dagum distributions by maximum likelihood             |
| STB-48 | <a href="#">sg107</a>    | S. P. Jenkins,<br>P. Van Kerm   | <code>glcurve</code>                                                                                                                                  | Generalized Lorenz curves and related graphs                                    |
| STB-49 | <a href="#">sg107.1</a>  | S. P. Jenkins,<br>P. Van Kerm   | <code>glcurve</code>                                                                                                                                  | update of <code>sg107</code>                                                    |
| SJ-1-1 | <a href="#">gr0001</a>   | S. P. Jenkins,<br>P. Van Kerm   | <code>glcurve7</code>                                                                                                                                 | update for Stata 7 of <code>sg107.1</code>                                      |
| SJ-7-2 | <a href="#">gr0001_3</a> | S. P. Van Kerm,<br>P. Jenkins   | <code>glcurve</code>                                                                                                                                  | update for Stata 8 of <code>gr0001</code> ; <i>install this version</i>         |
| STB-48 | <a href="#">sg108</a>    | P. Van Kerm                     | <code>poverty</code>                                                                                                                                  | Computing poverty indices                                                       |
| STB-51 | <a href="#">sg115</a>    | D. Jolliffe,<br>B. Krushelnysky | <code>ineqerr</code>                                                                                                                                  | Bootstrap standard errors for indices of inequality                             |
| STB-51 | <a href="#">sg117</a>    | D. Jolliffe,<br>A. Semykina     | <code>sepov</code>                                                                                                                                    | Robust standard errors for the Foster–Greer–Thorbecke class of poverty indices  |
| SJ-8-4 | <a href="#">st0100_1</a> | A. López-Feldman                | <code>descogini</code>                                                                                                                                | Decomposing inequality and obtaining marginal effects                           |
| SJ-6-4 | <a href="#">snp15_7</a>  | R. Newson                       | <code>somersd</code>                                                                                                                                  | Gini coefficient is a special case of Somers’ <i>D</i>                          |
| STB-23 | <a href="#">sg30</a>     | E. Whitehouse                   | <code>lorenz</code> ,<br><code>inequal</code> ,<br><code>atkinson</code> ,<br><code>relsgini</code>                                                   | Measures of inequality in Stata                                                 |

More commands may be available; enter Stata and type `search inequality measure, all`.

Max Otto Lorenz (1876–1959) was born in Iowa and studied at the Universities of Iowa and Wisconsin. He proposed what is now known as the Lorenz curve in 1905. Lorenz worked for the Interstate Commerce Commission between 1911 and 1944, mainly with transportation data. His hobbies included calendar reform and Interlingua, a proposed international language.

To download and install the Jenkins and Van Kerm `glcurve` command from the Internet, for instance, you could

1. Select **Help > SJ and User-written Programs**.
2. Click on *Stata Journal*.
3. Click on *sj4-4*.
4. Click on *gr0001\_1*.
5. Click on *click here to install*.

or you could instead do the following:

1. Navigate to the appropriate SJ issue:
  - a. Type `net` from `http://www.stata-journal.com/software`  
    Type `net cd sj4-4`
  - or
  - b. Type `net` from `http://www.stata-journal.com/software/sj4-4`
2. Type `net describe gr0001_1`
3. Type `net install gr0001_1`

To download and install the Jenkins `sumdist` command from the Internet, for instance, you could

1. Select **Help > SJ and User-written Programs**.
2. Click on *STB*.
3. Click on *stb48*.
4. Click on *sg104*.
5. Click on *click here to install*.

or you could instead do the following:

1. Navigate to the appropriate STB issue:
  - a. Type `net` from `http://www.stata.com`  
    Type `net cd stb`  
    Type `net cd stb48`
  - or
  - b. Type `net` from `http://www.stata.com/stb/stb48`
2. Type `net describe sg104`
3. Type `net install sg104`

## References

- Cox, N. J. 1999. [gr35: Diagnostic plots for assessing Singh–Maddala and Dagum distributions fitted by MLE](#). *Stata Technical Bulletin* 48: 2–4. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 72–74. College Station, TX: Stata Press.
- Goldstein, R. 1995. [sg31: Measures of diversity: Absolute and relative](#). *Stata Technical Bulletin* 23: 23–26. Reprinted in *Stata Technical Bulletin Reprints*, vol. 4, pp. 150–154. College Station, TX: Stata Press.
- Jenkins, S. P. 1999a. [sg104: Analysis of income distributions](#). *Stata Technical Bulletin* 48: 4–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 243–260. College Station, TX: Stata Press.
- . 1999b. [sg106: Fitting Singh–Maddala and Dagum distributions by maximum likelihood](#). *Stata Technical Bulletin* 48: 19–25. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 261–268. College Station, TX: Stata Press.
- Jenkins, S. P., and P. Van Kerm. 1999a. [sg107: Generalized Lorenz curves and related graphs](#). *Stata Technical Bulletin* 48: 25–29. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 269–274. College Station, TX: Stata Press.
- . 1999b. [sg107.1: Generalized Lorenz curves and related graphs](#). *Stata Technical Bulletin* 49: 23. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, p. 171. College Station, TX: Stata Press.
- . 2001. [Generalized Lorenz curves and related graphs: An update for Stata 7](#). *Stata Journal* 1: 107–112.
- . 2004. [gr0001\\_1: Software Update: Generalized Lorenz curves and related graphs](#). *Stata Journal* 4: 490.
- . 2006. [gr0001\\_2: Software Update: Generalized Lorenz curves and related graphs](#). *Stata Journal* 6: 597.
- . 2007. [gr0001\\_3: Software Update: Generalized Lorenz curves and related graphs](#). *Stata Journal* 7: 280.
- Jolliffe, D., and B. Krushelnysky. 1999. [sg115: Bootstrap standard errors for indices of inequality](#). *Stata Technical Bulletin* 51: 28–32. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 191–196. College Station, TX: Stata Press.
- Jolliffe, D., and A. Semykina. 1999. [sg117: Robust standard errors for the Foster–Greer–Thorbecke class of poverty indices](#). *Stata Technical Bulletin* 51: 34–36. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 200–203. College Station, TX: Stata Press.
- Kleiber, C., and S. Kotz. 2003. *Statistical Size Distributions in Economics and Actuarial Sciences*. Hoboken, NJ: Wiley.
- López-Feldman, A. 2006. [Decomposing inequality and obtaining marginal effects](#). *Stata Journal* 6: 106–111.
- . 2008. [Software Update: Decomposing inequality and obtaining marginal effects](#). *Stata Journal* 8: 594.
- Lorenz, M. O. 1905. Methods of measuring the concentration of wealth. *American Statistical Association* 9: 209–219.
- Newson, R. 2006. [Confidence intervals for rank statistics: Percentile slopes, differences, and ratios](#). *Stata Journal* 6: 497–520.
- Van Kerm, P. 1999. [sg108: Computing poverty indices](#). *Stata Technical Bulletin* 48: 29–33. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 274–278. College Station, TX: Stata Press.
- Whitehouse, E. 1995. [sg30: Measures of inequality in Stata](#). *Stata Technical Bulletin* 23: 20–23. Reprinted in *Stata Technical Bulletin Reprints*, vol. 4, pp. 146–150. College Station, TX: Stata Press.

Syntax

```
intreg depvar1 depvar2 [indepvars] [if] [in] [weight] [, options]
```

| <i>options</i>                                               | description                                                                                                                                                                 |
|--------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                                        |                                                                                                                                                                             |
| <code>noconstant</code>                                      | suppress constant term                                                                                                                                                      |
| <code>het(<i>varlist</i> [, <code>noconstant</code>])</code> | independent variables to model the variance; use <code>noconstant</code> to suppress constant term                                                                          |
| <code>offset(<i>varname</i>)</code>                          | include <i>varname</i> in model with coefficient constrained to 1                                                                                                           |
| <code>constraints(<i>constraints</i>)</code>                 | apply specified linear constraints                                                                                                                                          |
| <code>collinear</code>                                       | keep collinear variables                                                                                                                                                    |
| SE/Robust                                                    |                                                                                                                                                                             |
| <code>vce(<i>vcetype</i>)</code>                             | <i>vcetype</i> may be <code>oim</code> , <code>robust</code> , <code>cluster <i>clustvar</i></code> , <code>opg</code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                                                    |                                                                                                                                                                             |
| <code>level(#)</code>                                        | set confidence level; default is <code>level(95)</code>                                                                                                                     |
| <code>nocnsreport</code>                                     | do not display constraints                                                                                                                                                  |
| <code>display_options</code>                                 | control spacing and display of omitted variables and base and empty cells                                                                                                   |
| Maximization                                                 |                                                                                                                                                                             |
| <code>maximize_options</code>                                | control the maximization process; seldom used                                                                                                                               |
| <sup>†</sup> <code>coeflegend</code>                         | display coefficients' legend instead of coefficient table                                                                                                                   |

<sup>†</sup>`coeflegend` does not appear in the dialog box.

*indepvars* and *varlist* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar*<sub>1</sub>, *depvar*<sub>2</sub>, *indepvars*, and *varlist* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

`bootstrap`, `by`, `fracpoly`, `jackknife`, `mfp`, `nestreg`, `rolling`, `statsby`, `stepwise`, and `svy` are allowed; see [U] 11.1.10 Prefix commands.

Weights are not allowed with the `bootstrap` prefix.

`aweight`s are not allowed with the `jackknife` prefix.

`vce()` and weights are not allowed with the `svy` prefix.

`aweight`s, `fweight`s, `ilweight`s, and `pweight`s are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Linear models and related > Censored regression > Interval regression

## Description

`intreg` fits a model of  $y = [depvar_1, depvar_2]$  on *indepvars*, where  $y$  for each observation is point data, interval data, left-censored data, or right-censored data.

$depvar_1$  and  $depvar_2$  should have the following form:

| type of data        |                | $depvar_1$ | $depvar_2$ |
|---------------------|----------------|------------|------------|
| point data          | $a = [a, a]$   | $a$        | $a$        |
| interval data       | $[a, b]$       | $a$        | $b$        |
| left-censored data  | $(-\infty, b]$ | $.$        | $b$        |
| right-censored data | $[a, +\infty)$ | $a$        | $.$        |

## Options

### Model

`noconstant`; see [\[R\] estimation options](#).

`het(varlist [, noconstant])` specifies that *varlist* be included in the specification of the conditional variance. This *varlist* enters the variance specification collectively as multiplicative heteroskedasticity.

`offset(varname)`, `constraints(constraints)`, `collinear`; see [\[R\] estimation options](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [\[R\] vce\\_option](#).

### Reporting

`level(#)`; see [\[R\] estimation options](#).

`nocnsreport`; see [\[R\] estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [\[R\] estimation options](#).

### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrntolerance`, `from(init_specs)`; see [\[R\] maximize](#). These options are seldom used.

Setting the optimization type to `technique(bhhh)` resets the default *vcetype* to `vce(opg)`.

The following option is available with `intreg` but is not shown in the dialog box:

`coeflegend`; see [\[R\] estimation options](#).

## Remarks

`intreg` is a generalization of the models fit by `tobit`. Cameron and Trivedi (2009, 532–535) discuss the differences among censored, truncated, and interval data. If you know that the value for the  $j$ th individual is somewhere in the interval  $[y_{1j}, y_{2j}]$ , then the likelihood contribution from this individual is simply  $\Pr(y_{1j} \leq Y_j \leq y_{2j})$ . For censored data, their likelihoods contain terms of the form  $\Pr(Y_j \leq y_j)$  for left-censored data and  $\Pr(Y_j \geq y_j)$  for right-censored data, where  $y_j$  is the observed censoring value and  $Y_j$  denotes the random variable representing the dependent variable in the model.

Hence, `intreg` can fit models for data where each observation represents interval data, left-censored data, right-censored data, or point data. Regardless of the type of observation, the data should be stored in the dataset as interval data; that is, two dependent variables, `depvar1` and `depvar2`, are used to hold the endpoints of the interval. If the data are left-censored, the lower endpoint is  $-\infty$  and is represented by a missing value, `.'`, or an extended missing value, `._a, .b, . . . , .z`, in `depvar1`. If the data are right-censored, the upper endpoint is  $+\infty$  and is represented by a missing value, `.'` (or an extended missing value), in `depvar2`. Point data are represented by the two endpoints being equal.

| type of data        |                | <i>depvar<sub>1</sub></i> | <i>depvar<sub>2</sub></i> |
|---------------------|----------------|---------------------------|---------------------------|
| point data          | $a = [a, a]$   | $a$                       | $a$                       |
| interval data       | $[a, b]$       | $a$                       | $b$                       |
| left-censored data  | $(-\infty, b]$ | $.$                       | $b$                       |
| right-censored data | $[a, +\infty)$ | $a$                       | $.$                       |

Truly missing values of the dependent variable must be represented by missing values in both `depvar1` and `depvar2`.

Interval data arise naturally in many contexts, such as wage data. Often you know only that, for example, a person’s salary is between \$30,000 and \$40,000. Below we give an example for wage data and show how to set up `depvar1` and `depvar2`.

### ➤ Example 1

We have a dataset that contains the yearly wages of working women. Women were asked via a questionnaire to indicate a category for their yearly income from employment. The categories were less than 5,000, 5,001–10,000, . . . , 25,001–30,000, 30,001–40,000, 40,001–50,000, and more than 50,000. The wage categories are stored in the `wagecat` variable.

```
. use http://www.stata-press.com/data/r11/womenwage
(Wages of women)
. tab wagecat
```

| Wage<br>category<br>(\$1000s) | Freq. | Percent | Cum.   |
|-------------------------------|-------|---------|--------|
| 5                             | 14    | 2.87    | 2.87   |
| 10                            | 83    | 17.01   | 19.88  |
| 15                            | 158   | 32.38   | 52.25  |
| 20                            | 107   | 21.93   | 74.18  |
| 25                            | 57    | 11.68   | 85.86  |
| 30                            | 30    | 6.15    | 92.01  |
| 40                            | 19    | 3.89    | 95.90  |
| 50                            | 14    | 2.87    | 98.77  |
| 51                            | 6     | 1.23    | 100.00 |
| Total                         | 488   | 100.00  |        |

A value of 5 for `wagecat` represents the category less than 5,000, a value of 10 represents 5,001–10,000, ..., and a value of 51 represents greater than 50,000.

To use `intreg`, we must create two variables, `wage1` and `wage2`, containing the lower and upper endpoints of the wage categories. Here is one way to do it. We first create a dataset containing the nine wage categories, lag the wage categories into `wage1`, and match-merge this dataset with nine observations back into the main one.

```
. by wagecat: keep if _n==1
(479 observations deleted)
. generate wage1 = wagecat[_n-1]
(1 missing value generated)
. keep wagecat wage1
. save lagwage
file lagwage.dta saved
. use http://www.stata-press.com/data/r11/womenwage
(Wages of women)
. merge m:1 wagecat using lagwage
```

| Result      | # of obs. |                            |
|-------------|-----------|----------------------------|
| not matched | 0         |                            |
| matched     | 488       | ( <code>_merge==3</code> ) |

Now we create the upper endpoint and list the new variables:

```
. generate wage2 = wagecat
. replace wage2 = . if wagecat == 51
(6 real changes made, 6 to missing)
. sort age, stable
```

```
. list wage1 wage2 in 1/10
```

|     | wage1 | wage2 |
|-----|-------|-------|
| 1.  | .     | 5     |
| 2.  | 5     | 10    |
| 3.  | 5     | 10    |
| 4.  | 10    | 15    |
| 5.  | .     | 5     |
| 6.  | .     | 5     |
| 7.  | .     | 5     |
| 8.  | 5     | 10    |
| 9.  | 5     | 10    |
| 10. | 5     | 10    |

We can now run intreg:

```
. intreg wage1 wage2 age c.age#c.age nev_mar rural school tenure
```

Fitting constant-only model:

```
Iteration 0: log likelihood = -967.24956
Iteration 1: log likelihood = -967.1368
Iteration 2: log likelihood = -967.1368
```

Fitting full model:

```
Iteration 0: log likelihood = -856.65324
Iteration 1: log likelihood = -856.33294
Iteration 2: log likelihood = -856.33293
```

```
Interval regression Number of obs = 488
 LR chi2(6) = 221.61
Log likelihood = -856.33293 Prob > chi2 = 0.0000
```

|             | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-------------|-----------|-----------|-------|-------|----------------------|-----------|
| age         | .7914438  | .4433604  | 1.79  | 0.074 | -.0775265            | 1.660414  |
| c.age#c.age | -.0132624 | .0073028  | -1.82 | 0.069 | -.0275757            | .0010509  |
| nev_mar     | -.2075022 | .8119581  | -0.26 | 0.798 | -1.798911            | 1.383906  |
| rural       | -3.043044 | .7757324  | -3.92 | 0.000 | -4.563452            | -1.522637 |
| school      | 1.334721  | .1357873  | 9.83  | 0.000 | 1.068583             | 1.600859  |
| tenure      | .8000664  | .1045077  | 7.66  | 0.000 | .5952351             | 1.004898  |
| _cons       | -12.70238 | 6.367117  | -1.99 | 0.046 | -25.1817             | -.2230583 |
| /lnsigma    | 1.987823  | .0346543  | 57.36 | 0.000 | 1.919902             | 2.055744  |
| sigma       | 7.299626  | .2529634  |       |       | 6.82029              | 7.81265   |

```
Observation summary: 14 left-censored observations
 0 uncensored observations
 6 right-censored observations
 468 interval observations
```



We could also model these data by using an ordered probit model with `oprobit`:

```
. oprobit wagecat age c.age#c.age nev_mar rural school tenure
Iteration 0: log likelihood = -881.1491
Iteration 1: log likelihood = -764.31729
Iteration 2: log likelihood = -763.31191
Iteration 3: log likelihood = -763.31049
Iteration 4: log likelihood = -763.31049

Ordered probit regression Number of obs = 488
 LR chi2(6) = 235.68
 Prob > chi2 = 0.0000
 Pseudo R2 = 0.1337

Log likelihood = -763.31049
```

| wagecat     | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-------------|-----------|-----------|-------|-------|----------------------|-----------|
| age         | .1674519  | .0620333  | 2.70  | 0.007 | .0458689             | .289035   |
| c.age#c.age | -.0027983 | .0010214  | -2.74 | 0.006 | -.0048001            | -.0007964 |
| nev_mar     | -.0046417 | .1126737  | -0.04 | 0.967 | -.225478             | .2161946  |
| rural       | -.5270036 | .1100449  | -4.79 | 0.000 | -.7426875            | -.3113196 |
| school      | .2010587  | .0201189  | 9.99  | 0.000 | .1616263             | .2404911  |
| tenure      | .0989916  | .0147887  | 6.69  | 0.000 | .0700063             | .127977   |
| /cut1       | 2.650637  | .8957245  |       |       | .8950495             | 4.406225  |
| /cut2       | 3.941018  | .8979167  |       |       | 2.181134             | 5.700903  |
| /cut3       | 5.085205  | .9056582  |       |       | 3.310148             | 6.860263  |
| /cut4       | 5.875534  | .9120933  |       |       | 4.087864             | 7.663204  |
| /cut5       | 6.468723  | .918117   |       |       | 4.669247             | 8.268199  |
| /cut6       | 6.922726  | .9215455  |       |       | 5.11653              | 8.728922  |
| /cut7       | 7.34471   | .9237628  |       |       | 5.534168             | 9.155252  |
| /cut8       | 7.963441  | .9338881  |       |       | 6.133054             | 9.793828  |

We can directly compare the log likelihoods for the `intreg` and `oprobit` models because both likelihoods are discrete. If we had point data in our `intreg` estimation, the likelihood would be a mixture of discrete and continuous terms, and we could not compare it directly with the `oprobit` likelihood.

Here the `oprobit` log likelihood is significantly larger (i.e., less negative), so it fits better than the `intreg` model. The `intreg` model assumes normality, but the distribution of wages is skewed and definitely nonnormal. Normality is more closely approximated if we model the log of wages.

(Continued on next page)

```
. generate logwage1 = log(wage1)
(14 missing values generated)
. generate logwage2 = log(wage2)
(6 missing values generated)
. intreg logwage1 logwage2 age c.age#c.age nev_mar rural school tenure
Fitting constant-only model:
Iteration 0: log likelihood = -889.23647
Iteration 1: log likelihood = -889.06346
Iteration 2: log likelihood = -889.06346
Fitting full model:
Iteration 0: log likelihood = -773.81968
Iteration 1: log likelihood = -773.36566
Iteration 2: log likelihood = -773.36563
Interval regression Number of obs = 488
 LR chi2(6) = 231.40
Log likelihood = -773.36563 Prob > chi2 = 0.0000
```

|             | Coef.     | Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|-------------|-----------|-----------|--------|-------|----------------------|-----------|
| age         | .0645589  | .0249954  | 2.58   | 0.010 | .0155689             | .1135489  |
| c.age#c.age | -.0010812 | .0004115  | -2.63  | 0.009 | -.0018878            | -.0002746 |
| nev_mar     | -.0058151 | .0454867  | -0.13  | 0.898 | -.0949674            | .0833371  |
| rural       | -.2098361 | .0439454  | -4.77  | 0.000 | -.2959675            | -.1237047 |
| school      | .0804832  | .0076783  | 10.48  | 0.000 | .0654341             | .0955323  |
| tenure      | .0397144  | .0058001  | 6.85   | 0.000 | .0283464             | .0510825  |
| _cons       | .7084023  | .3593193  | 1.97   | 0.049 | .0041495             | 1.412655  |
| /lnsigma    | -.906989  | .0356265  | -25.46 | 0.000 | -.9768157            | -.8371623 |
| sigma       | .4037381  | .0143838  |        |       | .3765081             | .4329373  |

Observation summary:      14 left-censored observations  
                             0 uncensored observations  
                             6 right-censored observations  
                             468 interval observations

The log likelihood of this `intreg` model is close to the `oprobit` log likelihood, and the  $z$  statistics for both models are similar.

❑ Technical note

`intreg` has two parameterizations for the log-likelihood function: the transformed parameterization ( $\beta/\sigma, 1/\sigma$ ) and the untransformed parameterization ( $\beta, \ln(\sigma)$ ). By default, the log likelihood for `intreg` is parameterized in the transformed parameter space. This parameterization tends to be more convergent, but it requires that any starting values and constraints have the same parameterization, and it prevents the estimation with multiplicative heteroskedasticity. Therefore, when the `het()` option is specified, `intreg` switches to the untransformed log likelihood for the fit of the conditional-variance model. Similarly, specifying `from()` or `constraints()` causes the optimization in the untransformed parameter space to allow constraints on (and starting values for) the coefficients on the covariates without reference to  $\sigma$ .

The estimation results are all saved in the ( $\beta, \ln(\sigma)$ ) metric.

## Saved results

intreg saves the following in `e()`:

### Scalars

|                            |                                                   |
|----------------------------|---------------------------------------------------|
| <code>e(N)</code>          | number of observations                            |
| <code>e(N_unc)</code>      | number of uncensored observations                 |
| <code>e(N_ltc)</code>      | number of left-censored observations              |
| <code>e(N_rc)</code>       | number of right-censored observations             |
| <code>e(N_int)</code>      | number of interval observations                   |
| <code>e(k)</code>          | number of parameters                              |
| <code>e(k_aux)</code>      | number of auxiliary parameters                    |
| <code>e(k_eq)</code>       | number of equations                               |
| <code>e(k_eq_model)</code> | number of equations in model Wald test            |
| <code>e(k_dv)</code>       | number of dependent variables                     |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints    |
| <code>e(df_m)</code>       | model degrees of freedom                          |
| <code>e(ll)</code>         | log likelihood                                    |
| <code>e(ll_0)</code>       | log likelihood, constant-only model               |
| <code>e(N_clust)</code>    | number of clusters                                |
| <code>e(chi2)</code>       | $\chi^2$                                          |
| <code>e(p)</code>          | $p$ -value for model $\chi^2$ test                |
| <code>e(sigma)</code>      | sigma                                             |
| <code>e(se_sigma)</code>   | standard error of sigma                           |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                         |
| <code>e(rank0)</code>      | rank of <code>e(V)</code> for constant-only model |
| <code>e(ic)</code>         | number of iterations                              |
| <code>e(rc)</code>         | return code                                       |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                       |

*(Continued on next page)*

**Macros**

|                                 |                                                                          |
|---------------------------------|--------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>intreg</code>                                                      |
| <code>e(cmdline)</code>         | command as typed                                                         |
| <code>e(depvar)</code>          | names of dependent variables                                             |
| <code>e(wtype)</code>           | weight type                                                              |
| <code>e(wexp)</code>            | weight expression                                                        |
| <code>e(title)</code>           | title in estimation output                                               |
| <code>e(clustvar)</code>        | name of cluster variable                                                 |
| <code>e(offset)</code>          | offset                                                                   |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                  |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                           |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                            |
| <code>e(diparm#)</code>         | display transformed parameter #                                          |
| <code>e(het)</code>             | heteroskedasticity, if <code>het()</code> specified                      |
| <code>e(ml_score)</code>        | program used to implement <code>scores</code>                            |
| <code>e(opt)</code>             | type of optimization                                                     |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>       | type of ml method                                                        |
| <code>e(user)</code>            | name of likelihood-evaluator program                                     |
| <code>e(technique)</code>       | maximization technique                                                   |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular              |
| <code>e(crittype)</code>        | optimization criterion                                                   |
| <code>e(properties)</code>      | b V                                                                      |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                           |
| <code>e(footnote)</code>        | program used to implement the footnote display                           |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>           |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>           |

**Matrices**

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(gradient)</code>     | gradient vector                              |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

**Functions**

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`intreg` is implemented as an ado-file.

See [Wooldridge \(2009, sec. 17.4\)](#) or [Davidson and MacKinnon \(2004, sec. 11.6\)](#) for an introduction to censored and truncated regression models.

The likelihood for `intreg` subsumes that of the `tobit` models.

Let  $\mathbf{y} = \mathbf{X}\beta + \epsilon$  be the model.  $\mathbf{y}$  represents continuous outcomes—either observed or not observed. Our model assumes  $\epsilon \sim N(\mathbf{0}, \sigma^2 \mathbf{I})$ .

For observations  $j \in \mathcal{C}$ , we observe  $y_j$ , i.e., point data. Observations  $j \in \mathcal{L}$  are left-censored; we know only that the unobserved  $y_j$  is less than or equal to  $y_{\mathcal{L}j}$ , a censoring value that we do know. Similarly, observations  $j \in \mathcal{R}$  are right-censored; we know only that the unobserved  $y_j$  is greater than or equal to  $y_{\mathcal{R}j}$ . Observations  $j \in \mathcal{I}$  are intervals; we know only that the unobserved  $y_j$  is in the interval  $[y_{1j}, y_{2j}]$ .

The log likelihood is

$$\begin{aligned} \ln L = & -\frac{1}{2} \sum_{j \in \mathcal{C}} w_j \left\{ \left( \frac{y_j - \mathbf{x}\beta}{\sigma} \right)^2 + \log 2\pi\sigma^2 \right\} \\ & + \sum_{j \in \mathcal{L}} w_j \log \Phi \left( \frac{y_{\mathcal{L}j} - \mathbf{x}\beta}{\sigma} \right) \\ & + \sum_{j \in \mathcal{R}} w_j \log \left\{ 1 - \Phi \left( \frac{y_{\mathcal{R}j} - \mathbf{x}\beta}{\sigma} \right) \right\} \\ & + \sum_{j \in \mathcal{I}} w_j \log \left\{ \Phi \left( \frac{y_{2j} - \mathbf{x}\beta}{\sigma} \right) - \Phi \left( \frac{y_{1j} - \mathbf{x}\beta}{\sigma} \right) \right\} \end{aligned}$$

where  $\Phi()$  is the standard cumulative normal and  $w_j$  is the weight for the  $j$ th observation. If no weights are specified,  $w_j = 1$ . If `aweight`s are specified,  $w_j = 1$ , and  $\sigma$  is replaced by  $\sigma/\sqrt{a_j}$  in the above, where  $a_j$  are the `aweight`s normalized to sum to  $N$ .

Maximization is as described in [R] [maximize](#); the estimate reported as `_sigma` is  $\hat{\sigma}$ .

See Amemiya (1973) for a generalization of the tobit model to variable, but known, cutoffs.

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

`intreg` also supports estimation with survey data. For details on VCEs with survey data, see [SVY] [variance estimation](#).

## References

- Amemiya, T. 1973. Regression analysis when the dependent variable is truncated normal. *Econometrica* 41: 997–1016.
- Cameron, A. C., and P. K. Trivedi. 2009. *Microeconometrics Using Stata*. College Station, TX: Stata Press.
- Conroy, R. M. 2005. *Stings in the tails: Detecting and dealing with censored data*. *Stata Journal* 5: 395–404.
- Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.
- . 2004. *Econometric Theory and Methods*. New York: Oxford University Press.
- Goldberger, A. S. 1983. Abnormal selection bias. In *Studies in Econometrics, Time Series, and Multivariate Statistics*, ed. S. Karlin, T. Amemiya, and L. A. Goodman, 67–84. New York: Academic Press.
- Hurd, M. 1979. Estimation in truncated samples when there is heteroscedasticity. *Journal of Econometrics* 11: 247–258.
- Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.
- Stewart, M. B. 1983. On least squares estimation when the dependent variable is grouped. *Review of Economic Studies* 50: 737–753.
- Wooldridge, J. M. 2009. *Introductory Econometrics: A Modern Approach*. 4th ed. Cincinnati, OH: South-Western.

## Also see

[R] [intreg postestimation](#) — Postestimation tools for intreg

[R] [tobit](#) — Tobit regression

[R] [regress](#) — Linear regression

[SVY] [svy estimation](#) — Estimation commands for survey data

[XT] [xtintreg](#) — Random-effects interval-data regression models

[XT] [xttobit](#) — Random-effects tobit models

[U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `intreg`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

## Syntax for predict

```
predict [type] newvar [if] [in] [, statistic nooffset]
```

```
predict [type] { stub*|newvarreg newvarlnsigma } [if] [in] , scores
```

| statistic | description |
|-----------|-------------|
|-----------|-------------|

|                              |                                             |
|------------------------------|---------------------------------------------|
| Main                         |                                             |
| xb                           | linear prediction; the default              |
| stdp                         | standard error of the prediction            |
| stdf                         | standard error of the forecast              |
| pr( <i>a</i> , <i>b</i> )    | $\Pr(a < y_j < b)$                          |
| e( <i>a</i> , <i>b</i> )     | $E(y_j   a < y_j < b)$                      |
| ystar( <i>a</i> , <i>b</i> ) | $E(y_j^*), y_j^* = \max\{a, \min(y_j, b)\}$ |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

`stdf` is not allowed with `svy` postestimation results.

where *a* and *b* may be numbers or variables; *a* missing (*a* ≥ .) means  $-\infty$ , and *b* missing (*b* ≥ .) means  $+\infty$ ; see [U] [12.2.1 Missing values](#).

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

`xb`, the default, calculates the linear prediction.

`stdp` calculates the standard error of the prediction, which can be thought of as the standard error of the predicted expected value or mean for the observation’s covariate pattern. The standard error of the prediction is also referred to as the standard error of the fitted value.

`stdf` calculates the standard error of the forecast, which is the standard error of the point prediction for 1 observation. It is commonly referred to as the standard error of the future or forecast value. By construction, the standard errors produced by `stdf` are always larger than those produced by `stdp`; see [Methods and formulas](#) in [R] [regress](#).

`pr(a,b)` calculates  $\Pr(a < \mathbf{x}_j\mathbf{b} + u_j < b)$ , the probability that  $y_j|\mathbf{x}_j$  would be observed in the interval (*a*,*b*).

*a* and *b* may be specified as numbers or variable names; *lb* and *ub* are variable names;

`pr(20,30)` calculates  $\Pr(20 < \mathbf{x}_j\mathbf{b} + u_j < 30)$ ;

`pr(lb,ub)` calculates  $\Pr(lb < \mathbf{x}_j\mathbf{b} + u_j < ub)$ ; and

`pr(20,ub)` calculates  $\Pr(20 < \mathbf{x}_j\mathbf{b} + u_j < ub)$ .

*a* missing (*a* ≥ .) means  $-\infty$ ; `pr(. ,30)` calculates  $\Pr(-\infty < \mathbf{x}_j\mathbf{b} + u_j < 30)$ ;

`pr(lb,30)` calculates  $\Pr(-\infty < \mathbf{x}_j\mathbf{b} + u_j < 30)$  in observations for which *lb* ≥ . and calculates  $\Pr(lb < \mathbf{x}_j\mathbf{b} + u_j < 30)$  elsewhere.



$b$  missing ( $b \geq .$ ) means  $+\infty$ ; `pr(20,.)` calculates  $\Pr(+\infty > \mathbf{x}_j\mathbf{b} + u_j > 20)$ ;  
`pr(20,ub)` calculates  $\Pr(+\infty > \mathbf{x}_j\mathbf{b} + u_j > 20)$  in observations for which  $ub \geq .$   
 and calculates  $\Pr(20 < \mathbf{x}_j\mathbf{b} + u_j < ub)$  elsewhere.

`e(a,b)` calculates  $E(\mathbf{x}_j\mathbf{b} + u_j \mid a < \mathbf{x}_j\mathbf{b} + u_j < b)$ , the expected value of  $y_j \mid \mathbf{x}_j$  conditional on  $y_j \mid \mathbf{x}_j$  being in the interval  $(a, b)$ , meaning that  $y_j \mid \mathbf{x}_j$  is censored.  
 $a$  and  $b$  are specified as they are for `pr()`.

`ystar(a,b)` calculates  $E(y_j^*)$ , where  $y_j^* = a$  if  $\mathbf{x}_j\mathbf{b} + u_j \leq a$ ,  $y_j^* = b$  if  $\mathbf{x}_j\mathbf{b} + u_j \geq b$ , and  $y_j^* = \mathbf{x}_j\mathbf{b} + u_j$  otherwise, meaning that  $y_j^*$  is truncated.  $a$  and  $b$  are specified as they are for `pr()`.

`nooffset` is relevant only if you specified `offset(varname)`. It modifies the calculations made by `predict` so that they ignore the offset variable; the linear prediction is treated as  $\mathbf{x}_j\mathbf{b}$  rather than as  $\mathbf{x}_j\mathbf{b} + \text{offset}_j$ .

`scores` calculates equation-level score variables.

The first new variable will contain  $\partial \ln L / \partial (\mathbf{x}_j\beta)$ .

The second new variable will contain  $\partial \ln L / \partial \ln \sigma$ .

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [intreg](#) — Interval regression

[U] [20 Estimation and postestimation commands](#)

Syntax

Maximum likelihood estimator

```
ivprobit depvar [varlist1] (varlist2 = varlistiv) [if] [in] [weight] [, mle_options]
```

Two-step estimator

```
ivprobit depvar [varlist1] (varlist2 = varlistiv) [if] [in] [weight], twostep
[tse_options]
```

| <i>mle_options</i>                        | description                                                                                                               |
|-------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| Model                                     |                                                                                                                           |
| <u>m</u> le                               | use conditional maximum-likelihood estimator; the default                                                                 |
| <u>a</u> sis                              | retain perfect predictor variables                                                                                        |
| <u>c</u> onstraints( <i>constraints</i> ) | apply specified linear constraints                                                                                        |
| SE/Robust                                 |                                                                                                                           |
| vce( <i>vcetype</i> )                     | <i>vcetype</i> may be oim, <u>r</u> obust, <u>c</u> luster <i>clustvar</i> , opg, <u>b</u> ootstrap, or <u>j</u> ackknife |
| Reporting                                 |                                                                                                                           |
| <u>l</u> evel(#)                          | set confidence level; default is level(95)                                                                                |
| <u>f</u> irst                             | report first-stage regression                                                                                             |
| <u>n</u> ocnsreport                       | do not display constraints                                                                                                |
| <i>display_options</i>                    | control spacing and display of omitted variables and base and empty cells                                                 |
| Maximization                              |                                                                                                                           |
| <i>maximize_options</i>                   | control the maximization process                                                                                          |
| <sup>†</sup> <u>c</u> oefflegend          | display coefficients' legend instead of coefficient table                                                                 |

<sup>†</sup> coeflegend does not appear in the dialog box.

| <i>tse_options</i>             | description                                                                                     |
|--------------------------------|-------------------------------------------------------------------------------------------------|
| Model                          |                                                                                                 |
| * <b>twostep</b>               | use Newey's two-step estimator; the default is <code>mle</code>                                 |
| <b>asis</b>                    | retain perfect predictor variables                                                              |
| SE                             |                                                                                                 |
| <b>vce</b> ( <i>vcetype</i> )  | <i>vcetype</i> may be <code>twostep</code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                      |                                                                                                 |
| <b>level</b> (#)               | set confidence level; default is <code>level(95)</code>                                         |
| <b>first</b>                   | report first-stage regression                                                                   |
| <i>display_options</i>         | control spacing and display of omitted variables and base and empty cells                       |
| <sup>†</sup> <b>coeflegend</b> | display coefficients' legend instead of coefficient table                                       |

\***twostep** is required.

<sup>†</sup>**coeflegend** does not appear in the dialog box.

*varlist<sub>1</sub>* and *varlist<sub>1v</sub>* may contain factor variables; see [U] 11.4.3 [Factor variables](#).

*devar*, *varlist<sub>1</sub>*, *varlist<sub>2</sub>*, and *varlist<sub>1v</sub>* may contain time-series operators; see [U] 11.4.4 [Time-series varlists](#).

`bootstrap`, `by`, `jackknife`, `rolling`, `statsby`, and `svy` are allowed; see [U] 11.1.10 [Prefix commands](#).

Weights are not allowed with the `bootstrap` prefix.

`vce()`, `first`, `twostep`, and `weights` are not allowed with the `svy` prefix.

`fweights`, `iweights`, and `pweights` are allowed with the maximum likelihood estimator. `fweights` are allowed with Newey's two-step estimator. See [U] 11.1.6 [weight](#).

See [U] 20 [Estimation and postestimation commands](#) for more capabilities of estimation commands.

## Menu

Statistics > Endogenous covariates > Probit model with endogenous covariates

## Description

`ivprobit` fits probit models where one or more of the regressors are endogenously determined. By default, `ivprobit` uses maximum likelihood estimation. Alternatively, Newey's (1987) minimum chi-squared estimator can be invoked with the `twostep` option. Both estimators assume that the endogenous regressors are continuous and are not appropriate for use with discrete endogenous regressors. See [R] [ivtobit](#) for tobit estimation with endogenous regressors and [R] [probit](#) for probit estimation when the model contains no endogenous regressors.

## Options for ML estimator

### Model

`mle` requests that the conditional maximum-likelihood estimator be used. This is the default.

`asis` requests that all specified variables and observations be retained in the maximization process.

This option is typically not used and may introduce numerical instability. Normally, `ivprobit` drops any endogenous or exogenous variables that perfectly predict success or failure in the dependent variable. The associated observations are also dropped. For more information, see the discussion of model identification in [R] [probit](#).

`constraints(constraints)`; see [R] [estimation options](#).

#### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

#### Reporting

`level(#)`; see [R] [estimation options](#).

`first` requests that the parameters for the reduced-form equations showing the relationships between the endogenous variables and instruments be displayed. For the two-step estimator, `first` shows the first-stage regressions. For the maximum likelihood estimator, these parameters are estimated jointly with the parameters of the probit equation. The default is not to show these parameter estimates.

`nocnsreport`; see [R] [estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

#### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [R] [maximize](#). This model's likelihood function can be difficult to maximize, especially with multiple endogenous variables. The `difficult` and `technique(bfgs)` options may be helpful in achieving convergence.

Setting the optimization type to `technique(bhhh)` resets the default `vcetype` to `vce(opg)`.

The following option is available with `ivprobit` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Options for two-step estimator

#### Model

`twostep` is required and requests that Newey's (1987) efficient two-step estimator be used to obtain the coefficient estimates.

`asis` requests that all specified variables and observations be retained in the maximization process. This option is typically not used and may introduce numerical instability. Normally, `ivprobit` drops any endogenous or exogenous variables that perfectly predict success or failure in the dependent variable. The associated observations are also dropped. For more information, see the discussion of model identification in [R] [probit](#).

#### SE

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

## Reporting

`level(#)`; see [R] [estimation options](#).

`first` requests that the parameters for the reduced-form equations showing the relationships between the endogenous variables and instruments be displayed. For the two-step estimator, `first` shows the first-stage regressions. For the maximum likelihood estimator, these parameters are estimated jointly with the parameters of the probit equation. The default is not to show these parameter estimates.

*display\_options*: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

The following option is available with `ivprobit` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

Remarks are presented under the following headings:

*Model setup*

*Model identification*

## Model setup

`ivprobit` fits models with dichotomous dependent variables and endogenous regressors. You can use it to fit a probit model when you suspect that one or more of the regressors are correlated with the error term. `ivprobit` is to probit modeling what `ivregress` is to linear regression analysis; see [R] [ivregress](#) for more information.

Formally, the model is

$$\begin{aligned} y_{1i}^* &= \mathbf{y}_{2i}\boldsymbol{\beta} + \mathbf{x}_{1i}\boldsymbol{\gamma} + u_i \\ \mathbf{y}_{2i} &= \mathbf{x}_{1i}\boldsymbol{\Pi}_1 + \mathbf{x}_{2i}\boldsymbol{\Pi}_2 + \mathbf{v}_i \end{aligned}$$

where  $i = 1, \dots, N$ ,  $\mathbf{y}_{2i}$  is a  $1 \times p$  vector of endogenous variables,  $\mathbf{x}_{1i}$  is a  $1 \times k_1$  vector of exogenous variables,  $\mathbf{x}_{2i}$  is a  $1 \times k_2$  vector of additional instruments, and the equation for  $\mathbf{y}_{2i}$  is written in reduced form. By assumption,  $(u_i, \mathbf{v}_i) \sim N(\mathbf{0}, \boldsymbol{\Sigma})$ , where  $\sigma_{11}$  is normalized to one to identify the model.  $\boldsymbol{\beta}$  and  $\boldsymbol{\gamma}$  are vectors of structural parameters, and  $\boldsymbol{\Pi}_1$  and  $\boldsymbol{\Pi}_2$  are matrices of reduced-form parameters. This is a recursive model:  $\mathbf{y}_{2i}$  appears in the equation for  $y_{1i}^*$ , but  $y_{1i}^*$  does not appear in the equation for  $\mathbf{y}_{2i}$ . We do not observe  $y_{1i}^*$ ; instead, we observe

$$y_{1i} = \begin{cases} 0 & y_{1i}^* < 0 \\ 1 & y_{1i}^* \geq 0 \end{cases}$$

The order condition for identification of the structural parameters requires that  $k_2 \geq p$ . Presumably,  $\boldsymbol{\Sigma}$  is not block diagonal between  $u_i$  and  $\mathbf{v}_i$ ; otherwise,  $\mathbf{y}_{2i}$  would not be endogenous.

## □ Technical note

This model is derived under the assumption that  $(u_i, \mathbf{v}_i)$  is independent and identically distributed multivariate normal for all  $i$ . The `vce(cluster clustvar)` option can be used to control for a lack of independence. As with most probit models, if  $u_i$  is heteroskedastic, point estimates will be inconsistent. □

► Example 1

We have hypothetical data on 500 two-parent households, and we wish to model whether the woman is employed. We have a variable, `fem_work`, that is equal to one if she has a job and zero otherwise. Her decision to work is a function of the number of children at home (`kids`), number of years of schooling completed (`fem_educ`), and other household income measured in thousands of dollars (`other_inc`). We suspect that unobservable shocks affecting the woman’s decision to hold a job also affect the household’s other income. Therefore, we treat `other_inc` as endogenous. As an instrument, we use the number of years of schooling completed by the man (`male_educ`).

The syntax for specifying the exogenous, endogenous, and instrumental variables is identical to that used in `ivregress`; see [R] [ivregress](#) for details.

```
. use http://www.stata-press.com/data/r11/laborsup
. ivprobit fem_work fem_educ kids (other_inc = male_educ)

Fitting exogenous probit model
Iteration 0: log likelihood = -344.63508
Iteration 1: log likelihood = -255.36855
Iteration 2: log likelihood = -255.31444
Iteration 3: log likelihood = -255.31444

Fitting full model
Iteration 0: log likelihood = -2371.4753
Iteration 1: log likelihood = -2369.3178
Iteration 2: log likelihood = -2368.2198
Iteration 3: log likelihood = -2368.2062
Iteration 4: log likelihood = -2368.2062

Probit model with endogenous regressors
Log likelihood = -2368.2062

Number of obs = 500
Wald chi2(3) = 163.88
Prob > chi2 = 0.0000
```

|           | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-----------|-----------|-----------|-------|-------|----------------------|-----------|
| other_inc | -.0542756 | .0060854  | -8.92 | 0.000 | -.0662027            | -.0423485 |
| fem_educ  | .2111111  | .0268648  | 7.86  | 0.000 | .1584569             | .2637651  |
| kids      | -.1820929 | .0478267  | -3.81 | 0.000 | -.2758316            | -.0883543 |
| _cons     | .3672083  | .4480724  | 0.82  | 0.412 | -.5109975            | 1.245414  |
| <hr/>     |           |           |       |       |                      |           |
| /athrho   | .3907858  | .1509443  | 2.59  | 0.010 | .0949403             | .6866313  |
| /lnsigma  | 2.813383  | .0316228  | 88.97 | 0.000 | 2.751404             | 2.875363  |
| <hr/>     |           |           |       |       |                      |           |
| rho       | .3720374  | .1300519  |       |       | .0946561             | .5958135  |
| sigma     | 16.66621  | .5270318  |       |       | 15.66461             | 17.73186  |

```
Instrumented: other_inc
Instruments: fem_educ kids male_educ

Wald test of exogeneity (/athrho = 0): chi2(1) = 6.70 Prob > chi2 = 0.0096
```

Because we did not specify `mle` or `twostep`, `ivprobit` used the maximum likelihood estimator by default. At the top of the output, we see the iteration log. `ivprobit` fits a probit model ignoring endogeneity to obtain starting values for the endogenous model. The header of the output contains the sample size as well as a Wald statistic and *p*-value for the test of the hypothesis that all the slope coefficients are jointly zero. Below the table of coefficients, Stata reminds us that the endogenous variable is `other_inc` and that `fem_educ`, `kids`, and `male_educ` were used as instruments.

At the bottom of the output is a Wald test of the exogeneity of the instrumented variables. If the test statistic is not significant, there is not sufficient information in the sample to reject the null that

there is no endogeneity. Then a regular probit regression may be appropriate; the point estimates from `ivprobit` are consistent, though those from `probit` are likely to have smaller standard errors.

Various two-step estimators have also been proposed for the endogenous probit model, and Newey's (1987) minimum chi-squared estimator is available with the `twostep` option.

## ► Example 2

Refitting our labor-supply model with the two-step estimator yields

```
. ivprobit fem_work fem_educ kids (other_inc = male_educ), twostep
Checking reduced-form model...
```

|                                            |               |   |        |
|--------------------------------------------|---------------|---|--------|
| Two-step probit with endogenous regressors | Number of obs | = | 500    |
|                                            | Wald chi2(3)  | = | 93.97  |
|                                            | Prob > chi2   | = | 0.0000 |

|           | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval]   |
|-----------|-----------|-----------|-------|-------|------------------------|
| other_inc | -.058473  | .0093364  | -6.26 | 0.000 | -.0767719    -.040174  |
| fem_educ  | .227437   | .0281628  | 8.08  | 0.000 | .1722389    .282635    |
| kids      | -.1961748 | .0496323  | -3.95 | 0.000 | -.2934522    -.0988973 |
| _cons     | .3956061  | .4982649  | 0.79  | 0.427 | -.5809752    1.372187  |

```
Instrumented: other_inc
Instruments: fem_educ kids male_educ
```

|                          |           |      |                      |
|--------------------------|-----------|------|----------------------|
| Wald test of exogeneity: | chi2(1) = | 6.50 | Prob > chi2 = 0.0108 |
|--------------------------|-----------|------|----------------------|

All the coefficients have the same signs as their counterparts in the maximum likelihood model. The Wald test at the bottom of the output confirms our earlier finding of endogeneity.

## □ Technical note

In a standard probit model, the error term is assumed to have a variance of one. In the probit model with endogenous regressors, we assume that  $(u_i, v_i)$  is multivariate normal with covariance matrix

$$\text{Var}(u_i, v_i) = \Sigma = \begin{bmatrix} 1 & \Sigma'_{21} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$$

With the properties of the multivariate normal distribution,  $\text{Var}(u_i|v_i) = 1 - \Sigma'_{21}\Sigma_{22}^{-1}\Sigma_{21}$ . As a result, Newey's estimator and other two-step probit estimators do not yield estimates of  $\beta$  and  $\gamma$  but rather  $\beta/\sigma$  and  $\gamma/\sigma$ , where  $\sigma$  is the square root of  $\text{Var}(u_i|v_i)$ . Hence, we cannot directly compare the estimates obtained from Newey's estimator with those obtained via maximum likelihood or with those obtained from `probit`. See Wooldridge (2002, 472–477) for a discussion of Rivers and Vuong's (1988) two-step estimator. The issues raised pertaining to the interpretation of the coefficients of that estimator are identical to those that arise with Newey's estimator. Wooldridge also discusses ways to obtain marginal effects from two-step estimators.

Despite the coefficients not being directly comparable to their maximum likelihood counterparts, the two-step estimator is nevertheless useful. The maximum likelihood estimator may have difficulty

converging, especially with multiple endogenous variables. The two-step estimator, consisting of nothing more complicated than a probit regression, will almost certainly converge. Moreover, although the coefficients from the two models are not directly comparable, the two-step estimates can still be used to test for statistically significant relationships.

## Model identification

As in the linear simultaneous-equation model, the order condition for identification requires that the number of excluded exogenous variables (i.e., the additional instruments) be at least as great as the number of included endogenous variables. `ivprobit` checks this for you and issues an error message if the order condition is not met.

Like `probit`, `logit`, and `logistic`, `ivprobit` checks the exogenous and endogenous variables to see if any of them predict the outcome variable perfectly. It will then drop offending variables and observations and fit the model on the remaining data. Instruments that are perfect predictors do not affect estimation, so they are not checked. See [Model identification](#) in [\[R\] probit](#) for more information.

`ivprobit` will also occasionally display messages such as

Note: 4 failures and 0 successes completely determined.

For an explanation of this message, see [\[R\] logit](#).

## Saved results

`ivprobit`, `mle` saves the following in `e()`:

### Scalars

|                            |                                                                            |
|----------------------------|----------------------------------------------------------------------------|
| <code>e(N)</code>          | number of observations                                                     |
| <code>e(N_cds)</code>      | number of completely determined successes                                  |
| <code>e(N_cdf)</code>      | number of completely determined failures                                   |
| <code>e(k)</code>          | number of parameters                                                       |
| <code>e(k_eq)</code>       | number of equations                                                        |
| <code>e(k_eq_model)</code> | number of equations in model Wald test                                     |
| <code>e(k_aux)</code>      | number of auxiliary parameters                                             |
| <code>e(k_dv)</code>       | number of dependent variables                                              |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints                             |
| <code>e(df_m)</code>       | model degrees of freedom                                                   |
| <code>e(ll)</code>         | log likelihood                                                             |
| <code>e(N_clust)</code>    | number of clusters                                                         |
| <code>e(k_eq_skip)</code>  | identifies which equations should not be reported in the coefficient table |
| <code>e(endog_ct)</code>   | number of endogenous regressors                                            |
| <code>e(p)</code>          | model Wald $p$ -value                                                      |
| <code>e(p_exog)</code>     | exogeneity test Wald $p$ -value                                            |
| <code>e(chi2)</code>       | model Wald $\chi^2$                                                        |
| <code>e(chi2_exog)</code>  | Wald $\chi^2$ test of exogeneity                                           |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                                                  |
| <code>e(ic)</code>         | number of iterations                                                       |
| <code>e(rc)</code>         | return code                                                                |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                                                |



## Macros

|                                 |                                                                                        |
|---------------------------------|----------------------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>ivprobit</code>                                                                  |
| <code>e(cmdline)</code>         | command as typed                                                                       |
| <code>e(depvar)</code>          | name of dependent variable                                                             |
| <code>e(instd)</code>           | instrumented variables                                                                 |
| <code>e(insts)</code>           | instruments                                                                            |
| <code>e(wtype)</code>           | weight type                                                                            |
| <code>e(wexp)</code>            | weight expression                                                                      |
| <code>e(title)</code>           | title in estimation output                                                             |
| <code>e(clustvar)</code>        | name of cluster variable                                                               |
| <code>e(chi2type)</code>        | Wald; type of model $\chi^2$ test                                                      |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                                         |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                                          |
| <code>e(diparm#)</code>         | display transformed parameter #                                                        |
| <code>e(asis)</code>            | <i>asis</i> , if specified                                                             |
| <code>e(method)</code>          | <code>ml</code>                                                                        |
| <code>e(opt)</code>             | type of optimization                                                                   |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization               |
| <code>e(ml_method)</code>       | type of <code>ml</code> method                                                         |
| <code>e(user)</code>            | name of likelihood-evaluator program                                                   |
| <code>e(technique)</code>       | maximization technique                                                                 |
| <code>e(singularHmethod)</code> | <code>m-marquardt</code> or <code>hybrid</code> ; method used when Hessian is singular |
| <code>e(crittype)</code>        | optimization criterion                                                                 |
| <code>e(properties)</code>      | <code>b V</code>                                                                       |
| <code>e(estat_cmd)</code>       | program used to implement <code>estat</code>                                           |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                                         |
| <code>e(footnote)</code>        | program used to implement the footnote display                                         |
| <code>e(marginsok)</code>       | predictions allowed by <code>margins</code>                                            |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>                         |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>                         |

## Matrices

|                              |                                                   |
|------------------------------|---------------------------------------------------|
| <code>e(b)</code>            | coefficient vector                                |
| <code>e(Cns)</code>          | constraints matrix                                |
| <code>e(rules)</code>        | information about perfect predictors              |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)               |
| <code>e(gradient)</code>     | gradient vector                                   |
| <code>e(Sigma)</code>        | $\hat{\Sigma}$                                    |
| <code>e(V)</code>            | variance–covariance matrix of the estimators      |
| <code>e(V_modelbased)</code> | model-based variance                              |
| <code>e(ml_h)</code>         | derivative tolerance, $(\text{abs}(b)+1e-3)*1e-3$ |
| <code>e(ml_scale)</code>     | derivative scale factor                           |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

(Continued on next page)

`ivprobit`, `twostep` saves the following in `e()`:

#### Scalars

|                           |                                                    |
|---------------------------|----------------------------------------------------|
| <code>e(N)</code>         | number of observations                             |
| <code>e(N_cds)</code>     | number of completely determined successes          |
| <code>e(N_cdf)</code>     | number of completely determined failures           |
| <code>e(df_m)</code>      | model degrees of freedom                           |
| <code>e(df_exog)</code>   | degrees of freedom for $\chi^2$ test of exogeneity |
| <code>e(p)</code>         | model Wald $p$ -value                              |
| <code>e(p_exog)</code>    | exogeneity test Wald $p$ -value                    |
| <code>e(chi2)</code>      | model Wald $\chi^2$                                |
| <code>e(chi2_exog)</code> | Wald $\chi^2$ test of exogeneity                   |
| <code>e(rank)</code>      | rank of <code>e(V)</code>                          |

#### Macros

|                            |                                                                |
|----------------------------|----------------------------------------------------------------|
| <code>e(cmd)</code>        | <code>ivprobit</code>                                          |
| <code>e(cmdline)</code>    | command as typed                                               |
| <code>e(depvar)</code>     | name of dependent variable                                     |
| <code>e(instd)</code>      | instrumented variables                                         |
| <code>e(insts)</code>      | instruments                                                    |
| <code>e(wtype)</code>      | weight type                                                    |
| <code>e(wexp)</code>       | weight expression                                              |
| <code>e(chi2type)</code>   | Wald; type of model $\chi^2$ test                              |
| <code>e(vce)</code>        | <i>vcetype</i> specified in <code>vce()</code>                 |
| <code>e(vcetype)</code>    | title used to label Std. Err.                                  |
| <code>e(asis)</code>       | <i>asis</i> , if specified                                     |
| <code>e(method)</code>     | <code>twostep</code>                                           |
| <code>e(properties)</code> | <code>b V</code>                                               |
| <code>e(estat_cmd)</code>  | program used to implement <code>estat</code>                   |
| <code>e(predict)</code>    | program used to implement <code>predict</code>                 |
| <code>e(footnote)</code>   | program used to implement the footnote display                 |
| <code>e(marginsok)</code>  | predictions allowed by <code>margins</code>                    |
| <code>e(asbalanced)</code> | factor variables <code>fvset</code> as <code>asbalanced</code> |
| <code>e(asobserved)</code> | factor variables <code>fvset</code> as <code>asobserved</code> |

#### Matrices

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>e(b)</code>     | coefficient vector                           |
| <code>e(rules)</code> | information about perfect predictors         |
| <code>e(V)</code>     | variance–covariance matrix of the estimators |

#### Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`ivprobit` is implemented as an ado-file.

Fitting limited-dependent variable models with endogenous regressors has received considerable attention in the econometrics literature. Building on the results of Amemiya (1978, 1979), Newey (1987) developed an efficient method of estimation that encompasses both Rivers and Vuong's (1988) simultaneous-equations probit model and Smith and Blundell's (1986) simultaneous-equations tobit model. With modern computers, maximum likelihood estimation is feasible as well. For compactness, we write the model as

$$y_{1i}^* = \mathbf{z}_i \boldsymbol{\delta} + u_i \quad (1a)$$

$$\mathbf{y}_{2i} = \mathbf{x}_i \boldsymbol{\Pi} + v_i \quad (1b)$$

where  $\mathbf{z}_i = (\mathbf{y}_{2i}, \mathbf{x}_{1i})$ ,  $\mathbf{x}_i = (\mathbf{x}_{1i}, \mathbf{x}_{2i})$ ,  $\boldsymbol{\delta} = (\boldsymbol{\beta}', \boldsymbol{\gamma}')'$ , and  $\boldsymbol{\Pi} = (\boldsymbol{\Pi}'_1, \boldsymbol{\Pi}'_2)'$ .

Deriving the likelihood function is straightforward because we can write the joint density  $f(y_{1i}, \mathbf{y}_{2i} | \mathbf{x}_i)$  as  $f(y_{1i} | \mathbf{y}_{2i}, \mathbf{x}_i) f(\mathbf{y}_{2i} | \mathbf{x}_i)$ . When there is an endogenous regressor, the log likelihood for observation  $i$  is

$$\ln L_i = w_i \left[ y_{1i} \ln \Phi(m_i) + (1 - y_{1i}) \ln \{1 - \Phi(m_i)\} + \ln \phi\left(\frac{y_{2i} - \mathbf{x}_i \boldsymbol{\Pi}}{\sigma}\right) - \ln \sigma \right]$$

where

$$m_i = \frac{\mathbf{z}_i \boldsymbol{\delta} + \rho(y_{2i} - \mathbf{x}_i \boldsymbol{\Pi}) / \sigma}{(1 - \rho^2)^{\frac{1}{2}}}$$

$\Phi(\cdot)$  and  $\phi(\cdot)$  are the standard normal distribution and density functions, respectively;  $\sigma$  is the standard deviation of  $v_i$ ;  $\rho$  is the correlation coefficient between  $u_i$  and  $v_i$ ; and  $w_i$  is the weight for observation  $i$  or one if no weights were specified. Instead of estimating  $\sigma$  and  $\rho$ , we estimate  $\ln \sigma$  and  $\text{atanh } \rho$ , where

$$\text{atanh } \rho = \frac{1}{2} \ln \left( \frac{1 + \rho}{1 - \rho} \right)$$

For multiple endogenous regressors, let

$$\text{Var}(u_i, \mathbf{v}_i) = \boldsymbol{\Sigma} = \begin{bmatrix} 1 & \boldsymbol{\Sigma}'_{21} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}$$

As in any probit model, we have imposed the normalization  $\text{Var}(u_i) = 1$  to identify the model. The log likelihood for observation  $i$  is

$$\ln L_i = w_i \left[ y_{1i} \ln \Phi(m_i) + (1 - y_{1i}) \ln \{1 - \Phi(m_i)\} + \ln f(\mathbf{y}_{2i} | \mathbf{x}_i) \right]$$

where

$$\ln f(\mathbf{y}_{2i} | \mathbf{x}_i) = -\frac{p}{2} \ln 2\pi - \frac{1}{2} \ln |\boldsymbol{\Sigma}_{22}| - \frac{1}{2} (\mathbf{y}_{2i} - \mathbf{x}_i \boldsymbol{\Pi}) \boldsymbol{\Sigma}_{22}^{-1} (\mathbf{y}_{2i} - \mathbf{x}_i \boldsymbol{\Pi})'$$

and

$$m_i = (1 - \boldsymbol{\Sigma}'_{21} \boldsymbol{\Sigma}_{22}^{-1} \boldsymbol{\Sigma}_{21})^{-\frac{1}{2}} \{ \mathbf{z}_i \boldsymbol{\delta} + (\mathbf{y}_{2i} - \mathbf{x}_i \boldsymbol{\Pi}) \boldsymbol{\Sigma}_{22}^{-1} \boldsymbol{\Sigma}_{21} \}$$

Instead of maximizing the log-likelihood function with respect to  $\boldsymbol{\Sigma}$ , we maximize with respect to the Cholesky decomposition  $\mathbf{S}$  of  $\boldsymbol{\Sigma}$ ; that is, there exists a lower triangular matrix,  $\mathbf{S}$ , such that  $\mathbf{S}\mathbf{S}' = \boldsymbol{\Sigma}$ . This maximization ensures that  $\boldsymbol{\Sigma}$  is positive definite, as a covariance matrix must be. Let

$$\mathbf{S} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ s_{21} & s_{22} & 0 & \dots & 0 \\ s_{31} & s_{32} & s_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{p+1,1} & s_{p+1,2} & s_{p+1,3} & \dots & s_{p+1,p+1} \end{bmatrix}$$

With maximum likelihood estimation, this command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

The maximum likelihood version of `heckman` also supports estimation with survey data. For details on VCEs with survey data, see [SVY] [variance estimation](#).

The two-step estimates are obtained using Newey's (1987) minimum chi-squared estimator. The reduced-form equation for  $y_{1i}^*$  is

$$\begin{aligned} y_{1i}^* &= (x_i \Pi + v_i) \beta + x_{1i} \gamma + u_i \\ &= x_i \alpha + v_i \beta + u_i \\ &= x_i \alpha + \nu_i \end{aligned}$$

where  $\nu_i = v_i \beta + u_i$ . Because  $u_i$  and  $v_i$  are jointly normal,  $\nu_i$  is also normal. Note that

$$\alpha = \begin{bmatrix} \Pi_1 \\ \Pi_2 \end{bmatrix} \beta + \begin{bmatrix} I \\ 0 \end{bmatrix} \gamma = D(\Pi) \delta$$

where  $D(\Pi) = (\Pi, I_1)$  and  $I_1$  is defined such that  $x_i I_1 = x_{1i}$ . Letting  $\hat{z}_i = (x_i \hat{\Pi}, x_{1i})$ ,  $\hat{z}_i \delta = x_i D(\hat{\Pi}) \delta$ , where  $D(\hat{\Pi}) = (\hat{\Pi}, I_1)$ . Thus one estimator of  $\alpha$  is  $D(\hat{\Pi}) \delta$ ; denote this estimator by  $\hat{D} \delta$ .

$\alpha$  could also be estimated directly as the solution to

$$\max_{\alpha, \lambda} \sum_{i=1}^N l(y_{1i}, x_i \alpha + \hat{v}_i \lambda) \quad (2)$$

where  $l(\cdot)$  is the log likelihood for probit. Denote this estimator by  $\tilde{\alpha}$ . The inclusion of the  $\hat{v}_i \lambda$  term follows because the multivariate normality of  $(u_i, v_i)$  implies that, conditional on  $y_{2i}$ , the expected value of  $u_i$  is nonzero. Because  $v_i$  is unobservable, the least-squares residuals from fitting equation (1b) are used.

Amemiya (1978) shows that the estimator of  $\delta$  defined by

$$\max_{\delta} (\tilde{\alpha} - \hat{D} \delta)' \hat{\Omega}^{-1} (\tilde{\alpha} - \hat{D} \delta)$$

where  $\hat{\Omega}$  is a consistent estimator of the covariance of  $\sqrt{N}(\tilde{\alpha} - \hat{D} \delta)$ , is asymptotically efficient relative to all other estimators that minimize the distance between  $\tilde{\alpha}$  and  $D(\hat{\Pi}) \delta$ . Thus an efficient estimator of  $\delta$  is

$$\hat{\delta} = (\hat{D}' \hat{\Omega}^{-1} \hat{D})^{-1} \hat{D}' \hat{\Omega}^{-1} \tilde{\alpha} \quad (3)$$

and

$$\text{Var}(\hat{\delta}) = (\hat{D}' \hat{\Omega}^{-1} \hat{D})^{-1} \quad (4)$$

To implement this estimator, we need  $\hat{\Omega}^{-1}$ .

Consider the two-step maximum likelihood estimator that results from first fitting equation (1b) by OLS and computing the residuals  $\hat{v}_i = y_{2i} - x_i \hat{\Pi}$ . The estimator is then obtained by solving

$$\max_{\delta, \lambda} \sum_{i=1}^N l(y_{1i}, z_i \delta + \hat{v}_i \lambda)$$

This is the two-step instrumental variables (2SIV) estimator proposed by Rivers and Vuong (1988), and its role will become apparent shortly.

From Proposition 5 of Newey (1987),  $\sqrt{N}(\tilde{\alpha} - \hat{D}\delta) \xrightarrow{d} N(\mathbf{0}, \Omega)$ , where

$$\Omega = J_{\alpha\alpha}^{-1} + (\lambda - \beta)' \Sigma_{22} (\lambda - \beta) Q^{-1}$$

and  $\Sigma_{22} = E\{v_i' v_i\}$ .  $J_{\alpha\alpha}^{-1}$  is simply the covariance matrix of  $\tilde{\alpha}$ , ignoring that  $\hat{\Pi}$  is an estimated parameter matrix. Moreover, Newey shows that the covariance matrix from an OLS regression of  $y_{2i}(\hat{\lambda} - \hat{\beta})$  on  $x_i$  is a consistent estimator of the second term.  $\hat{\lambda}$  can be obtained from solving (2), and the 2SIV estimator yields a consistent estimate,  $\hat{\beta}$ .

Mechanically, estimation proceeds in several steps.

1. Each of the endogenous right-hand-side variables is regressed on all the exogenous variables, and the fitted values and residuals are calculated. The matrix  $\hat{D} = D(\hat{\Pi})$  is assembled from the estimated coefficients.
2. `probit` is used to solve (2) and obtain  $\tilde{\alpha}$  and  $\hat{\lambda}$ . The portion of the covariance matrix corresponding to  $\alpha$ ,  $J_{\alpha\alpha}^{-1}$ , is also saved.
3. The 2SIV estimator is evaluated, and the parameters  $\hat{\beta}$  corresponding to  $y_{2i}$  are collected.
4.  $y_{2i}(\hat{\lambda} - \hat{\beta})$  is regressed on  $x_i$ . The covariance matrix of the parameters from this regression is added to  $J_{\alpha\alpha}^{-1}$ , yielding  $\hat{\Omega}$ .
5. Evaluating equations (3) and (4) yields the estimates  $\hat{\delta}$  and  $\text{Var}(\hat{\delta})$ .
6. A Wald test of the null hypothesis  $H_0: \lambda = \mathbf{0}$ , using the 2SIV estimates, serves as our test of exogeneity.

The two-step estimates are not directly comparable to those obtained from the maximum likelihood estimator or from `probit`. The argument is the same for Newey's efficient estimator as for Rivers and Vuong's (1988) 2SIV estimator, so we consider the simpler 2SIV estimator. From the properties of the normal distribution,

$$E(u_i | v_i) = v_i \Sigma_{22}^{-1} \Sigma_{21} \quad \text{and} \quad \text{Var}(u_i | v_i) = 1 - \Sigma_{21}' \Sigma_{22}^{-1} \Sigma_{21}$$

We write  $u_i$  as  $u_i = v_i \Sigma_{22}^{-1} \Sigma_{21} + e_i = v_i \lambda + e_i$ , where  $e_i \sim N(0, 1 - \rho^2)$ ,  $\rho^2 = \Sigma_{21}' \Sigma_{22}^{-1} \Sigma_{21}$ , and  $e_i$  is independent of  $v_i$ . In the second stage of 2SIV, we use a probit regression to estimate the parameters of

$$y_{1i} = z_i \delta + v_i \lambda + e_i$$

Because  $v_i$  is unobservable, we use the sample residuals from the first-stage regressions.

$$\Pr(y_{1i} = 1 | z_i, v_i) = \Pr(z_i \delta + v_i \lambda + e_i > 0 | z_i, v_i) = \Phi \left\{ (1 - \rho^2)^{-\frac{1}{2}} (z_i \delta + v_i \lambda) \right\}$$

Hence, as mentioned previously, 2SIV and Newey's estimator do not estimate  $\delta$  and  $\lambda$  but rather

$$\delta_\rho = \frac{1}{(1 - \rho^2)^{\frac{1}{2}}} \delta \quad \text{and} \quad \lambda_\rho = \frac{1}{(1 - \rho^2)^{\frac{1}{2}}} \lambda$$

## Acknowledgments

The two-step estimator is based on the `probitiv` command written by Jonah Gelbach and the `ivprob` command written by Joe Harkness.

## References

- Amemiya, T. 1978. The estimation of a simultaneous equation generalized probit model. *Econometrica* 46: 1193–1205.
- . 1979. The estimation of a simultaneous-equation tobit model. *International Economic Review* 20: 169–181.
- Miranda, A., and S. Rabe-Hesketh. 2006. Maximum likelihood estimation of endogenous switching and sample selection models for binary, ordinal, and count variables. *Stata Journal* 6: 285–308.
- Newey, W. K. 1987. Efficient estimation of limited dependent variable models with endogenous explanatory variables. *Journal of Econometrics* 36: 231–250.
- Rivers, D., and Q. H. Vuong. 1988. Limited information estimators and exogeneity tests for simultaneous probit models. *Journal of Econometrics* 39: 347–366.
- Smith, R. J., and R. Blundell. 1986. An exogeneity test for the simultaneous equation tobit model with an application to labor supply. *Econometrica* 54: 679–685.
- Wooldridge, J. M. 2002. *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA: MIT Press.

## Also see

- [R] [ivprobit postestimation](#) — Postestimation tools for `ivprobit`
- [R] [gmm](#) — Generalized method of moments estimation
- [R] [ivregress](#) — Single-equation instrumental-variables regression
- [R] [ivtobit](#) — Tobit model with continuous endogenous regressors
- [R] [probit](#) — Probit regression
- [XT] [xtprobit](#) — Random-effects and population-averaged probit models
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are of special interest after `ivprobit`:

| command                 | description                                                                                              |
|-------------------------|----------------------------------------------------------------------------------------------------------|
| <code>estat clas</code> | <code>estat classification</code> reports various summary statistics, including the classification table |
| <code>lroc</code>       | graphs the ROC curve and calculates the area under the curve                                             |
| <code>lsens</code>      | graphs sensitivity and specificity versus probability cutoff                                             |

These commands are not appropriate after the two-step estimator or the `svy` prefix.

For information about these commands, see [\[R\] logistic postestimation](#).

The following standard postestimation commands are also available:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code> <sup>1</sup>  | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>hausman</code>             | Hausman’s specification test                                                                        |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code> <sup>2</sup> | likelihood-ratio test; not available with two-step estimator                                        |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code> <sup>1</sup>  | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `estat ic` and `suest` are not appropriate after `ivprobit`, `twostep`.

<sup>2</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Syntax for predict

After *ML* or *twostep*

```
predict [type] newvar [if] [in] [, statistic rules asif]
```

After ML

```
predict [type] { stub* | newvarlist } [if] [in] , scores
```

| statistic | description                                                              |
|-----------|--------------------------------------------------------------------------|
| Main      |                                                                          |
| xb        | linear prediction; the default                                           |
| stdp      | standard error of the linear prediction                                  |
| pr        | probability of a positive outcome; not available with two-step estimator |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

Menu

Statistics > Postestimation > Predictions, residuals, etc.

Options for predict

- Main
- `xb`, the default, calculates the linear prediction.
  - `stdp` calculates the standard error of the linear prediction.
  - `pr` calculates the probability of a positive outcome. `pr` is not available with the two-step estimator.
  - `rules` requests that Stata use any rules that were used to identify the model when making the prediction. By default, Stata calculates missing for excluded observations. `rules` is not available with the two-step estimator.
  - `asif` requests that Stata ignore the rules and the exclusion criteria and calculate predictions for all observations possible using the estimated parameters from the model. `asif` is not available with the two-step estimator.
  - `scores`, not available with `twostep`, calculates equation-level score variables.

For models with one endogenous regressor, four new variables are created.

- The first new variable will contain  $\partial \ln L / \partial (z_i \delta)$ .
- The second new variable will contain  $\partial \ln L / \partial (x_i \Pi)$ .
- The third new variable will contain  $\partial \ln L / \partial \operatorname{atanh} \rho$ .
- The fourth new variable will contain  $\partial \ln L / \partial \ln \sigma$ .

For models with  $p$  endogenous regressors,  $p + \{(p + 1)(p + 2)\} / 2$  new variables are created.

- The first new variable will contain  $\partial \ln L / \partial (z_i \delta)$ .
- The second through  $(p + 1)$ th new variables will contain  $\partial \ln L / \partial (x_i \Pi_k)$ ,  $k = 1, \dots, p$ , where  $\Pi_k$  is the  $k$ th column of  $\Pi$ .
- The remaining score variables will contain the partial derivatives of  $\ln L$  with respect to  $s_{21}, s_{31}, \dots, s_{p+1,1}, s_{22}, \dots, s_{p+1,2}, \dots, s_{p+1,p+1}$ , where  $s_{m,n}$  denotes the  $(m, n)$  element of the Cholesky decomposition of the error covariance matrix.



## Remarks

Remarks are presented under the following headings:

*Marginal effects*

*Obtaining predicted values*

## Marginal effects

### ► Example 1

We can obtain marginal effects by using the `margins` command after `ivprobit`. We will calculate average marginal effects by using the labor-supply model of [example 1](#) in [\[R\] ivprobit](#).

```
. use http://www.stata-press.com/data/r11/laborsup
. ivprobit fem_work fem_educ kids (other_inc = male_educ)
 (output omitted)
. margins, dydx(*) predict(pr)

Average marginal effects Number of obs = 500
Model VCE : OIM

Expression : Probability of positive outcome, predict(pr)
dy/dx w.r.t. : other_inc fem_educ kids male_educ
```

|           | Delta-method |           |        | P> z  | [95% Conf. Interval] |           |
|-----------|--------------|-----------|--------|-------|----------------------|-----------|
|           | dy/dx        | Std. Err. | z      |       |                      |           |
| other_inc | -.014015     | .0009836  | -14.25 | 0.000 | -.0159428            | -.0120872 |
| fem_educ  | .0545129     | .0066007  | 8.26   | 0.000 | .0415758             | .06745    |
| kids      | -.0470199    | .0123397  | -3.81  | 0.000 | -.0712052            | -.0228346 |
| male_educ | (omitted)    |           |        |       |                      |           |

Here we see that a \$1,000 increase in `other_inc` leads to an average decrease of 0.014 in the probability that the woman has a job. `male_edu` has no effect because it appears only as an instrument.



## Obtaining predicted values

After fitting your model with `ivprobit`, you can obtain the linear prediction and its standard error for both the estimation sample and other samples by using the `predict` command; see [\[U\] 20 Estimation and postestimation commands](#) and [\[R\] predict](#). If you had used the maximum likelihood estimator, you could also obtain the probability of a positive outcome.

`predict`'s `p` option calculates the probability of a positive outcome, remembering any rules used to identify the model, and calculates missing for excluded observations. `predict`'s `rules` option uses the rules in predicting probabilities, whereas `predict`'s `asif` option ignores both the rules and the exclusion criteria and calculates probabilities for all possible observations by using the estimated parameters from the model. See [Obtaining predicted values](#) in [\[R\] probit postestimation](#) for an example.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

The linear prediction is calculated as  $z_i\hat{\delta}$ , where  $\hat{\delta}$  is the estimated value of  $\delta$ , and  $z_i$  and  $\delta$  are defined in (1a) of [R] [ivprobit](#). The probability of a positive outcome is  $\Phi(z_i\hat{\delta})$ , where  $\Phi(\cdot)$  is the standard normal distribution function.

## Also see

[R] [ivprobit](#) — Probit model with continuous endogenous regressors

[U] [20 Estimation and postestimation commands](#)

Syntax

```
ivregress estimator depvar [varlist1] (varlist2 = varlistiv) [if] [in] [weight]
[, options]
```

| estimator | description                                   |
|-----------|-----------------------------------------------|
| 2sls      | two-stage least squares (2SLS)                |
| liml      | limited-information maximum likelihood (LIML) |
| gmm       | generalized method of moments (GMM)           |

| options | description |
|---------|-------------|
|---------|-------------|

|                                          |                                                                                                                                                       |
|------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                    |                                                                                                                                                       |
| <u>no</u> constant                       | suppress constant term                                                                                                                                |
| <u>has</u> cons                          | has user-supplied constant                                                                                                                            |
| GMM <sup>1</sup>                         |                                                                                                                                                       |
| <u>w</u> matrix( <i>wmtype</i> )         | <i>wmtype</i> may be <u>robust</u> , <u>cluster</u> <i>clustvar</i> , <i>hac kernel</i> , or <u>unadjusted</u>                                        |
| <u>c</u> enter                           | center moments in weight matrix computation                                                                                                           |
| <u>i</u> gmm                             | use iterative instead of two-step GMM estimator                                                                                                       |
| <u>eps</u> (#) <sup>2</sup>              | specify # for parameter convergence criterion; default is <u>eps</u> (1e-6)                                                                           |
| <u>weps</u> (#) <sup>2</sup>             | specify # for weight matrix convergence criterion; default is <u>weps</u> (1e-6)                                                                      |
| <u>optimization_options</u> <sup>2</sup> | control the optimization process; seldom used                                                                                                         |
| SE/Robust                                |                                                                                                                                                       |
| <u>vce</u> ( <i>vcetype</i> )            | <i>vcetype</i> may be <u>unadjusted</u> , <u>robust</u> , <u>cluster</u> <i>clustvar</i> , <u>bootstrap</u> , <u>jackknife</u> , or <i>hac kernel</i> |
| Reporting                                |                                                                                                                                                       |
| <u>l</u> evel(#)                         | set confidence level; default is <u>level</u> (95)                                                                                                    |
| <u>f</u> irst                            | report first-stage regression                                                                                                                         |
| <u>s</u> mall                            | make degrees-of-freedom adjustments and report small-sample statistics                                                                                |
| <u>no</u> header                         | display only the coefficient table                                                                                                                    |
| <u>de</u> pname( <i>depname</i> )        | substitute dependent variable name                                                                                                                    |
| <u>e</u> form( <i>string</i> )           | report exponentiated coefficients and use <i>string</i> to label them                                                                                 |
| <u>display_options</u>                   | control spacing and display of omitted variables and base and empty cells                                                                             |
| <sup>†</sup> <u>p</u> erfect             | do not check for collinearity between endogenous regressors and excluded instruments                                                                  |
| <sup>†</sup> <u>c</u> oefflegend         | display coefficients' legend instead of coefficient table                                                                                             |

<sup>1</sup>These options may be specified only when `gmm` is specified.

<sup>2</sup>These options may be specified only when `igmm` is specified.

†`perfect` and `coeflegend` do not appear in the dialog box.

`varlist1` and `varlistiv` may contain factor variables; see [U] 11.4.3 Factor variables.

`depvar`, `varlist1`, `varlist2`, and `varlistiv` may contain time-series operators; see [U] 11.4.4 Time-series varlists.

`bootstrap`, `by`, `jackknife`, `rolling`, `statsby`, and `svy` are allowed; see [U] 11.1.10 Prefix commands.

Weights are not allowed with the `bootstrap` prefix.

`aweight`s are not allowed with the `jackknife` prefix.

`hascons`, `vce()`, `noheader`, `depname()`, and `weights` are not allowed with the `svy` prefix.

`aweight`s, `fweight`s, `iweight`s, and `pweight`s are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

## Menu

Statistics > Endogenous covariates > Single-equation instrumental-variables regression

## Description

`ivregress` fits a linear regression of `depvar` on `varlist1` and `varlist2`, using `varlistiv` (along with `varlist1`) as instruments for `varlist2`. `ivregress` supports estimation via two-stage least squares (2SLS), limited-information maximum likelihood (LIML), and generalized method of moments (GMM).

In the language of instrumental variables, `varlist1` and `varlistiv` are the exogenous variables, and `varlist2` are the endogenous variables.

## Options

### Model

`noconstant`; see [R] estimation options.

`hascons` indicates that a user-defined constant or its equivalent is specified among the independent variables.

### GMM

`wmatrix(wmtype)` specifies the type of weighting matrix to be used in conjunction with the GMM estimator.

Specifying `wmatrix(robust)` requests a weighting matrix that is optimal when the error term is heteroskedastic. `wmatrix(robust)` is the default.

Specifying `wmatrix(cluster clustvar)` requests a weighting matrix that accounts for arbitrary correlation among observations within clusters identified by `clustvar`.

Specifying `wmatrix(hac kernel #)` requests a heteroskedasticity- and autocorrelation-consistent (HAC) weighting matrix using the specified kernel (see below) with `#` lags. The bandwidth of a kernel is equal to `# + 1`.

Specifying `wmatrix(hac kernel opt)` requests an HAC weighting matrix using the specified kernel, and the lag order is selected using Newey and West's (1994) optimal lag-selection algorithm.

Specifying `wmatrix(hac kernel)` requests an HAC weighting matrix using the specified kernel and  $N - 2$  lags, where  $N$  is the sample size.

There are three kernels available for HAC weighting matrices, and you may request each one by using the name used by statisticians or the name perhaps more familiar to economists:

`bartlett` or `nwest` requests the Bartlett (Newey–West) kernel;

`parzen` or `gallant` requests the Parzen (Gallant 1987) kernel; and

`quadraticspectral` or `andrews` requests the quadratic spectral (Andrews 1991) kernel.

Specifying `wmatrix(unadjusted)` requests a weighting matrix that is suitable when the errors are homoskedastic. The two-step GMM estimator with this weighting matrix is equivalent to the 2SLS estimator.

`center` requests that the sample moments be centered (demeaned) when computing GMM weight matrices. By default, centering is not done.

`igmm` requests that the iterative GMM estimator be used instead of the default two-step GMM estimator. Convergence is declared when the relative change in the parameter vector from one iteration to the next is less than `eps()` or the relative change in the weight matrix is less than `weps()`.

`eps(#)` specifies the convergence criterion for successive parameter estimates when the iterative GMM estimator is used. The default is `eps(1e-6)`. Convergence is declared when the relative difference between successive parameter estimates is less than `eps()` and the relative difference between successive estimates of the weighting matrix is less than `weps()`.

`weps(#)` specifies the convergence criterion for successive estimates of the weighting matrix when the iterative GMM estimator is used. The default is `weps(1e-6)`. Convergence is declared when the relative difference between successive parameter estimates is less than `eps()` and the relative difference between successive estimates of the weighting matrix is less than `weps()`.

*optimization\_options*: `iterate(#)`, `[no]log`. `iterate()` specifies the maximum number of iterations to perform in conjunction with the iterative GMM estimator. The default is 16,000 or the number set using `set maxiter`. `log/nolog` specifies whether to show the iteration log. These options are seldom used.

---

#### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

`vce(unadjusted)`, the default for `2sls` and `liml`, specifies that an unadjusted (nonrobust) VCE matrix be used. The default for `gmm` is based on the `wmtype` specified in the `wmatrix()` option; see `wmatrix(wmtype)` above. If `wmatrix()` is specified with `gmm` but `vce()` is not, then `vcetype` is set equal to `wmtype`. To override this behavior and obtain an unadjusted (nonrobust) VCE matrix, specify `vce(unadjusted)`.

`ivregress` also allows the following:

`vce(hac kernel [#|opt])` specifies that an HAC covariance matrix be used. The syntax used with `vce(hac kernel ...)` is identical to that used with `wmatrix(hac kernel ...)`; see `wmatrix(wmtype)` above.

---

#### Reporting

`level(#)`; see [R] [estimation options](#).

`first` requests that the first-stage regression results be displayed.

`small` requests that the degrees-of-freedom adjustment  $N/(N - k)$  be made to the variance–covariance matrix of parameters and that small-sample  $F$  and  $t$  statistics be reported, where  $N$  is the sample

size and  $k$  is the number of parameters estimated. By default, no degrees-of-freedom adjustment is made, and Wald and  $z$  statistics are reported. Even with this option, no degrees-of-freedom adjustment is made to the weighting matrix when the GMM estimator is used.

`noheader` suppresses the display of the summary statistics at the top of the output, displaying only the coefficient table.

`depname(depname)` is used only in programs and ado-files that use `ivregress` to fit models other than instrumental-variables regression. `depname()` may be specified only at estimation time. *depname* is recorded as the identity of the dependent variable, even though the estimates are calculated using *depvar*. This method affects the labeling of the output—not the results calculated—but could affect later calculations made by `predict`, where the residual would be calculated as deviations from *depname* rather than *depvar*. `depname()` is most typically used when *depvar* is a temporary variable (see [P] [macro](#)) used as a proxy for *depname*.

`eform(string)` is used only in programs and ado-files that use `ivregress` to fit models other than instrumental-variables regression. `eform()` specifies that the coefficient table be displayed in “exponentiated form”, as defined in [R] [maximize](#), and that *string* be used to label the exponentiated coefficients in the table.

*display\_options*: [noomitted](#), [vsquish](#), [noemptycells](#), [baselevels](#), [allbaselevels](#); see [R] [estimation options](#).

The following options are available with `ivregress` but are not shown in the dialog box:

`perfect` requests that `ivregress` not check for collinearity between the endogenous regressors and excluded instruments, allowing one to specify “perfect” instruments. This option cannot be used with the LIML estimator. This option may be required when using `ivregress` to implement other estimators.

`coeflegend`; see [R] [estimation options](#).

## Remarks

`ivregress` performs instrumental-variables regression and weighted instrumental-variables regression. For a general discussion of instrumental variables, see [Baum \(2006\)](#), [Cameron and Trivedi \(2005; 2009, chap. 6\)](#) [Davidson and MacKinnon \(1993, 2004\)](#), [Greene \(2008, chap. 12\)](#), and [Wooldridge \(2002, 2009\)](#). See [Hall \(2005\)](#) for a lucid presentation of GMM estimation. [Angrist and Pischke \(2009, chap. 4\)](#) offer a casual yet thorough introduction to instrumental-variables estimators, including their use in estimating treatment effects. Some of the earliest work on simultaneous systems can be found in Cowles Commission monographs—[Koopmans and Marschak \(1950\)](#) and [Koopmans and Hood \(1953\)](#)—with the first developments of 2SLS appearing in [Theil \(1953\)](#) and [Basmann \(1957\)](#). However, [Stock and Watson \(2007, 424–428\)](#) present an example of the method of instrumental variables that was first published in 1928 by Philip Wright.

The syntax for `ivregress` assumes that you want to fit one equation from a system of equations or an equation for which you do not want to specify the functional form for the remaining equations of the system. To fit a full system of equations, using either 2SLS equation-by-equation or three-stage least squares, see [R] [reg3](#). An advantage of `ivregress` is that you can fit one equation of a multiple-equation system without specifying the functional form of the remaining equations.

Formally, the model fit by `ivregress` is

$$y_i = \mathbf{y}_i\beta_1 + \mathbf{x}_{1i}\beta_2 + u_i \quad (1)$$

$$\mathbf{y}_i = \mathbf{x}_{1i}\Pi_1 + \mathbf{x}_{2i}\Pi_2 + \mathbf{v}_i \quad (2)$$

Here  $y_i$  is the dependent variable for the  $i$ th observation,  $\mathbf{y}_i$  represents the endogenous regressors (*varlist<sub>2</sub>* in the syntax diagram),  $\mathbf{x}_{1i}$  represents the included exogenous regressors (*varlist<sub>1</sub>* in the syntax diagram), and  $\mathbf{x}_{2i}$  represents the excluded exogenous regressors (*varlist<sub>iv</sub>* in the syntax diagram).  $\mathbf{x}_{1i}$  and  $\mathbf{x}_{2i}$  are collectively called the instruments.  $u_i$  and  $\mathbf{v}_i$  are zero-mean error terms, and the correlations between  $u_i$  and the elements of  $\mathbf{v}_i$  are presumably nonzero.

The rest of the discussion is presented under the following headings:

*2SLS and LIML estimators*

*GMM estimator*

## 2SLS and LIML estimators

The most common instrumental-variables estimator is 2SLS.

### ► Example 1: 2SLS estimator

We have state data from the 1980 census on the median dollar value of owner-occupied housing (*hsgval*) and the median monthly gross rent (*rent*). We want to model *rent* as a function of *hsgval* and the percentage of the population living in urban areas (*pcturban*):

$$\text{rent}_i = \beta_0 + \beta_1 \text{hsgval}_i + \beta_2 \text{pcturban}_i + u_i$$

where  $i$  indexes states and  $u_i$  is an error term.

Because random shocks that affect rental rates in a state probably also affect housing values, we treat *hsgval* as endogenous. We believe that the correlation between *hsgval* and  $u$  is not equal to zero. On the other hand, we have no reason to believe that the correlation between *pcturban* and  $u$  is nonzero, so we assume that *pcturban* is exogenous.

Because we are treating *hsgval* as an endogenous regressor, we must have one or more additional variables available that are correlated with *hsgval* but uncorrelated with  $u$ . Moreover, these excluded exogenous variables must not affect *rent* directly, because if they do then they should be included in the regression equation we specified above. In our dataset, we have a variable for family income (*faminc*) and for region of the country (*region*) that we believe are correlated with *hsgval* but not the error term. Together, *pcturban*, *faminc*, and factor variables *2.region*, *3.region*, and *4.region* constitute our set of instruments.

To fit the equation in Stata, we specify the dependent variable and the list of included exogenous variables. In parentheses, we specify the endogenous regressors, an equal sign, and the excluded exogenous variables. Only the additional exogenous variables must be specified to the right of the equal sign; the exogenous variables that appear in the regression equation are automatically included as instruments.

(Continued on next page)

Here we fit our model with the 2SLS estimator:

```
. use http://www.stata-press.com/data/r11/hsng
(1980 Census housing data)
. ivregress 2sls rent pcturban (hsngval = faminc i.region)
Instrumental variables (2SLS) regression
Number of obs = 50
Wald chi2(2) = 90.76
Prob > chi2 = 0.0000
R-squared = 0.5989
Root MSE = 22.166
```

| rent     | Coef.    | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|----------|----------|-----------|------|-------|----------------------|----------|
| hsngval  | .0022398 | .0003284  | 6.82 | 0.000 | .0015961             | .0028836 |
| pcturban | .081516  | .2987652  | 0.27 | 0.785 | -.504053             | .667085  |
| _cons    | 120.7065 | 15.22839  | 7.93 | 0.000 | 90.85942             | 150.5536 |

Instrumented: hsngval  
Instruments: pcturban faminc 2.region 3.region 4.region

As we would expect, states with higher housing values have higher rental rates. The proportion of a state’s population that is urban does not have a significant effect on rents.



□ Technical note

In a simultaneous-equations framework, we could write the model we just fit as

$$\begin{aligned} \text{hsngval}_i &= \pi_0 + \pi_1 \text{faminc}_i + \pi_2 \text{2.region}_i + \pi_3 \text{3.region}_i + \pi_4 \text{4.region}_i + v_i \\ \text{rent}_i &= \beta_0 + \beta_1 \text{hsngval}_i + \beta_2 \text{pcturban}_i + u_i \end{aligned}$$

which here happens to be recursive (triangular), because `hsngval` appears in the equation for `rent` but `rent` does not appear in the equation for `hsngval`. In general, however, systems of simultaneous equations are not recursive. Because this system is recursive, we could fit the two equations individually via OLS if we were willing to assume that  $u$  and  $v$  were independent. For a more detailed discussion of triangular systems, see [Kmenta \(1997, 719–720\)](#).

Historically, instrumental-variables estimation and systems of simultaneous equations were taught concurrently, and older textbooks describe instrumental-variables estimation solely in the context of simultaneous equations. However, in recent decades, the treatment of endogeneity and instrumental-variables estimation has taken on a much broader scope, while interest in the specification of complete systems of simultaneous equations has waned. Most recent textbooks, such as [Cameron and Trivedi \(2005\)](#), [Davidson and MacKinnon \(1993, 2004\)](#), and [Wooldridge \(2002, 2009\)](#), treat instrumental-variables estimation as an integral part of the modern economists’ toolkit and introduce it long before shorter discussions on simultaneous equations.



In addition to the 2SLS member of the  $\kappa$ -class estimators, `ivregress` implements the LIML estimator. Both theoretical and Monte Carlo exercises indicate that the LIML estimator may yield less bias and confidence intervals with better coverage rates than the 2SLS estimator. See [Poi \(2006\)](#) and [Stock, Wright, and Yogo \(2002\)](#) (and the papers cited therein) for Monte Carlo evidence.



## ► Example 2: LIML estimator

Here we refit our model with the LIML estimator:

```
. ivregress liml rent pcturban (hsngval = faminc i.region)
```

|                                          |                 |        |
|------------------------------------------|-----------------|--------|
| Instrumental variables (LIML) regression | Number of obs = | 50     |
|                                          | Wald chi2(2) =  | 75.71  |
|                                          | Prob > chi2 =   | 0.0000 |
|                                          | R-squared =     | 0.4901 |
|                                          | Root MSE =      | 24.992 |

| rent     | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|----------|-----------|-----------|-------|-------|----------------------|----------|
| hsngval  | .0026686  | .0004173  | 6.39  | 0.000 | .0018507             | .0034865 |
| pcturban | -.1827391 | .3571132  | -0.51 | 0.609 | -.8826681            | .5171899 |
| _cons    | 117.6087  | 17.22625  | 6.83  | 0.000 | 83.84587             | 151.3715 |

Instrumented: hsngval

Instruments: pcturban faminc 2.region 3.region 4.region

These results are qualitatively similar to the 2SLS results, although the coefficient on `hsngval` is about 19% higher.

◄

## GMM estimator

Since the celebrated paper of [Hansen \(1982\)](#), the GMM has been a popular method of estimation in economics and finance, and it lends itself well to instrumental-variables estimation. The basic principle is that we have some *moment* or *orthogonality* conditions of the form

$$E(\mathbf{z}_i u_i) = \mathbf{0} \quad (3)$$

From (1), we have  $u_i = y_i - \mathbf{y}_i \beta_1 - \mathbf{x}_{1i} \beta_2$ . What are the elements of the instrument vector  $\mathbf{z}_i$ ? By assumption,  $\mathbf{x}_{1i}$  is uncorrelated with  $u_i$ , as are the excluded exogenous variables  $\mathbf{x}_{2i}$ , and so we use  $\mathbf{z}_i = [\mathbf{x}_{1i} \ \mathbf{x}_{2i}]$ . The moment conditions are simply the mathematical representation of the assumption that the instruments are exogenous—that is, the instruments are orthogonal to (uncorrelated with)  $u_i$ .

If the number of elements in  $\mathbf{z}_i$  is just equal to the number of unknown parameters, then we can apply the analogy principle to (3) and solve

$$\frac{1}{N} \sum_i \mathbf{z}_i u_i = \frac{1}{N} \sum_i \mathbf{z}_i (y_i - \mathbf{y}_i \beta_1 - \mathbf{x}_{1i} \beta_2) = \mathbf{0} \quad (4)$$

This equation is known as the method of moments estimator. Here where the number of instruments equals the number of parameters, the method of moments estimator coincides with the 2SLS estimator, which also coincides with what has historically been called the indirect least-squares estimator (Judge et al. 1985, 595).

The “generalized” in GMM addresses the case in which the number of instruments (columns of  $\mathbf{z}_i$ ) exceeds the number of parameters to be estimated. Here there is no unique solution to the population moment conditions defined in (3), so we cannot use (4). Instead, we define the objective function

$$Q(\beta_1, \beta_2) = \left( \frac{1}{N} \sum_i \mathbf{z}_i u_i \right)' \mathbf{W} \left( \frac{1}{N} \sum_i \mathbf{z}_i u_i \right) \quad (5)$$

where  $\mathbf{W}$  is a positive-definite matrix with the same number of rows and columns as the number of columns of  $\mathbf{z}_i$ .  $\mathbf{W}$  is known as the weighting matrix, and we specify its structure with the `wmatrix()` option. The GMM estimator of  $(\beta_1, \beta_2)$  minimizes  $Q(\beta_1, \beta_2)$ ; that is, the GMM estimator chooses  $\beta_1$  and  $\beta_2$  to make the moment conditions as close to zero as possible for a given  $\mathbf{W}$ . For a more general GMM estimator, see [R] `gmm`. `gmm` does not restrict you to fitting a single linear equation, though the syntax is more complex.

A well-known result is that if we define the matrix  $\mathbf{S}_0$  to be the covariance of  $\mathbf{z}_i u_i$  and set  $\mathbf{W} = \mathbf{S}_0^{-1}$ , then we obtain the optimal two-step GMM estimator, where by optimal estimator we mean the one that results in the smallest variance given the moment conditions defined in (3).

Suppose that the errors  $u_i$  are heteroskedastic but independent among observations. Then

$$\mathbf{S}_0 = E(\mathbf{z}_i u_i u_i \mathbf{z}_i') = E(u_i^2 \mathbf{z}_i \mathbf{z}_i')$$

and the sample analogue is

$$\hat{\mathbf{S}} = \frac{1}{N} \sum_i \hat{u}_i^2 \mathbf{z}_i \mathbf{z}_i' \tag{6}$$

To implement this estimator, we need estimates of the sample residuals  $\hat{u}_i$ . `ivregress gmm` obtains the residuals by estimating  $\beta_1$  and  $\beta_2$  by 2SLS and then evaluates (6) and sets  $\mathbf{W} = \hat{\mathbf{S}}^{-1}$ . Equation (6) is the same as the center term of the “sandwich” robust covariance matrix available from most Stata estimation commands through the `vce(robust)` option.

► Example 3: GMM estimator

Here we refit our model of rents by using the GMM estimator, allowing for heteroskedasticity in  $u_i$ :

```
. ivregress gmm rent pcturban (hsngval = faminc i.region), wmatrix(robust)
Instrumental variables (GMM) regression Number of obs = 50
 Wald chi2(2) = 112.09
 Prob > chi2 = 0.0000
 R-squared = 0.6616
 Root MSE = 20.358

GMM weight matrix: Robust
```

| rent     | Coef.    | Robust<br>Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|----------|----------|---------------------|-------|-------|----------------------|----------|
| hsngval  | .0014643 | .0004473            | 3.27  | 0.001 | .0005877             | .002341  |
| pcturban | .7615482 | .2895105            | 2.63  | 0.009 | .1941181             | 1.328978 |
| _cons    | 112.1227 | 10.80234            | 10.38 | 0.000 | 90.95052             | 133.2949 |

```
Instrumented: hsngval
Instruments: pcturban faminc 2.region 3.region 4.region
```

Because we requested that a heteroskedasticity-consistent weighting matrix be used during estimation but did not specify the `vce()` option, `ivregress` reported standard errors that are robust to heteroskedasticity. Had we specified `vce(unadjusted)`, we would have obtained standard errors that would be correct only if the weighting matrix  $\mathbf{W}$  does in fact converge to  $\mathbf{S}_0^{-1}$ .

## □ Technical note

Many software packages that implement GMM estimation use the same heteroskedasticity-consistent weighting matrix we used in the previous example to obtain the optimal two-step estimates but do not use a heteroskedasticity-consistent VCE, even though they may label the standard errors as being “robust”. To replicate results obtained from other packages, you may have to use the `vce(unadjusted)` option. See [Methods and formulas](#) below for a discussion of robust covariance matrix estimation in the GMM framework.

□

By changing our definition of  $\mathbf{S}_0$ , we can obtain GMM estimators suitable for use with other types of data that violate the assumption that the errors are independent and identically distributed. For example, you may have a dataset that consists of multiple observations for each person in a sample. The observations that correspond to the same person are likely to be correlated, and the estimation technique should account for that lack of independence. Say that in your dataset, people are identified by the variable `personid` and you type

```
. ivregress gmm ..., wmatrix(cluster personid)
```

Here `ivregress` estimates  $\mathbf{S}_0$  as

$$\hat{\mathbf{S}} = \frac{1}{N} \sum_{c \in C} \mathbf{q}_c \mathbf{q}_c'$$

where  $C$  denotes the set of clusters and

$$\mathbf{q}_c = \sum_{i \in c_j} \hat{u}_i \mathbf{z}_i$$

where  $c_j$  denotes the  $j$ th cluster. This weighting matrix accounts for the within-person correlation among observations, so the GMM estimator that uses this version of  $\mathbf{S}_0$  will be more efficient than the estimator that ignores this correlation.

## ▷ Example 4: GMM estimator with clustering

We have data from the National Longitudinal Survey on young women’s wages as reported in a series of interviews from 1968 through 1988, and we want to fit a model of wages as a function of each woman’s age and age squared, job tenure, birth year, and level of education. We believe that random shocks that affect a woman’s wage also affect her job tenure, so we treat tenure as endogenous. As additional instruments, we use her union status, number of weeks worked in the past year, and a dummy indicating whether she lives in a metropolitan area. Because we have several observations for each woman (corresponding to interviews done over several years), we want to control for clustering on each person.

(Continued on next page)

```
. use http://www.stata-press.com/data/r11/nlswork
(National Longitudinal Survey. Young Women 14-26 years of age in 1968)
. ivregress gmm ln_wage age c.age#c.age birth_yr grade
> (tenure = union wks_work msp), wmatrix(cluster idcode)

Instrumental variables (GMM) regression Number of obs = 18625
 Wald chi2(5) = 1807.17
 Prob > chi2 = 0.0000
 R-squared = .
GMM weight matrix: Cluster (idcode) Root MSE = .46951
 (Std. Err. adjusted for 4110 clusters in idcode)
```

| ln_wage     | Coef.     | Robust Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-------------|-----------|------------------|-------|-------|----------------------|-----------|
| tenure      | .099221   | .0037764         | 26.27 | 0.000 | .0918194             | .1066227  |
| age         | .0171146  | .0066895         | 2.56  | 0.011 | .0040034             | .0302259  |
| c.age#c.age | -.0005191 | .000111          | -4.68 | 0.000 | -.0007366            | -.0003016 |
| birth_yr    | -.0085994 | .0021932         | -3.92 | 0.000 | -.012898             | -.0043008 |
| grade       | .071574   | .0029938         | 23.91 | 0.000 | .0657062             | .0774417  |
| _cons       | .8575071  | .1616274         | 5.31  | 0.000 | .5407231             | 1.174291  |

```
Instrumented: tenure
Instruments: age c.age#c.age birth_yr grade union wks_work msp
```

Both job tenure and years of schooling have significant positive effects on wages. 4

Time-series data are often plagued by serial correlation. In these cases, we can construct a weighting matrix to account for the fact that the error in period  $t$  is probably correlated with the errors in periods  $t - 1$ ,  $t - 2$ , etc. An HAC weighting matrix can be used to account for both serial correlation and potential heteroskedasticity.

To request an HAC weighting matrix, you specify the `wmatrix(hac kernel [#|opt])` option. *kernel* specifies which of three kernels to use: `bartlett`, `parzen`, or `quadraticspectral`. *kernel* determines the amount of weight given to lagged values when computing the HAC matrix, and `#` denotes the maximum number of lags to use. Many texts refer to the bandwidth of the kernel instead of the number of lags; the bandwidth is equal to the number of lags plus one. If neither `opt` nor `#` is specified, then  $N - 2$  lags are used, where  $N$  is the sample size.

If you specify `wmatrix(hac kernel opt)`, then `ivregress` uses Newey and West's (1994) algorithm for automatically selecting the number of lags to use. Although the authors' Monte Carlo simulations do show that the procedure may result in size distortions of hypothesis tests, the procedure is still useful when little other information is available to help choose the number of lags.

For more on GMM estimation, see [Baum \(2006\)](#); [Baum, Schaffer, and Stillman \(2003, 2007\)](#); [Cameron and Trivedi \(2005\)](#); [Davidson and MacKinnon \(1993, 2004\)](#); [Hayashi \(2000\)](#); or [Wooldridge \(2002\)](#). See [Newey and West \(1987\)](#) for an introduction to HAC covariance matrix estimation.

## Saved results

`ivregress` saves the following in `e()`:

### Scalars

|                            |                                                               |
|----------------------------|---------------------------------------------------------------|
| <code>e(N)</code>          | number of observations                                        |
| <code>e(mss)</code>        | model sum of squares                                          |
| <code>e(df_m)</code>       | model degrees of freedom                                      |
| <code>e(rss)</code>        | residual sum of squares                                       |
| <code>e(df_r)</code>       | residual degrees of freedom                                   |
| <code>e(r2)</code>         | $R^2$                                                         |
| <code>e(r2_a)</code>       | adjusted $R^2$                                                |
| <code>e(F)</code>          | $F$ statistic                                                 |
| <code>e(rmse)</code>       | root mean squared error                                       |
| <code>e(N_clust)</code>    | number of clusters                                            |
| <code>e(chi2)</code>       | $\chi^2$ statistic                                            |
| <code>e(kappa)</code>      | $\kappa$ used in LIML estimator                               |
| <code>e(J)</code>          | value of GMM objective function                               |
| <code>e(wlagopt)</code>    | lags used in HAC weight matrix (if Newey–West algorithm used) |
| <code>e(vcelagopt)</code>  | lags used in HAC VCE matrix (if Newey–West algorithm used)    |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                                     |
| <code>e(iterations)</code> | number of GMM iterations (0 if not applicable)                |

### Macros

|                               |                                                                                                     |
|-------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>e(cmd)</code>           | <code>ivregress</code>                                                                              |
| <code>e(cmdline)</code>       | command as typed                                                                                    |
| <code>e(depvar)</code>        | name of dependent variable                                                                          |
| <code>e(instd)</code>         | instrumented variable                                                                               |
| <code>e(insts)</code>         | instruments                                                                                         |
| <code>e(constant)</code>      | <code>noconstant</code> or <code>hasconstant</code> if specified                                    |
| <code>e(wtype)</code>         | weight type                                                                                         |
| <code>e(wexp)</code>          | weight expression                                                                                   |
| <code>e(title)</code>         | title in estimation output                                                                          |
| <code>e(clustvar)</code>      | name of cluster variable                                                                            |
| <code>e(hac_kernel)</code>    | HAC kernel                                                                                          |
| <code>e(hac_lag)</code>       | HAC lag                                                                                             |
| <code>e(vce)</code>           | <i>vcetype</i> specified in <code>vce()</code>                                                      |
| <code>e(vcetype)</code>       | title used to label Std. Err.                                                                       |
| <code>e(robust_prolog)</code> | program to prepare estimates for linearized VCE computations                                        |
| <code>e(robust_epilog)</code> | program to finalize estimates after linearized VCE computations                                     |
| <code>e(estimator)</code>     | 2sls, liml, or gmm                                                                                  |
| <code>e(exogr)</code>         | exogenous regressors                                                                                |
| <code>e(wmatrix)</code>       | <i>wmtype</i> specified in <code>wmatrix()</code>                                                   |
| <code>e(moments)</code>       | <code>centered</code> if <code>center</code> specified                                              |
| <code>e(small)</code>         | <code>small</code> if small-sample statistics                                                       |
| <code>e(depname)</code>       | <i>depname</i> if <code>depname(depname)</code> specified; otherwise same as <code>e(depvar)</code> |
| <code>e(properties)</code>    | <code>b V</code>                                                                                    |
| <code>e(estat_cmd)</code>     | program used to implement <code>estat</code>                                                        |
| <code>e(predict)</code>       | program used to implement <code>predict</code>                                                      |
| <code>e(footnote)</code>      | program used to implement footnote display                                                          |
| <code>e(marginsok)</code>     | predictions allowed by <code>margins</code>                                                         |
| <code>e(marginsnotok)</code>  | predictions disallowed by <code>margins</code>                                                      |

`e(asbalanced)` factor variables `fvset` as `asbalanced`  
`e(asobserved)` factor variables `fvset` as `asobserved`

#### Matrices

`e(b)` coefficient vector  
`e(W)` weight matrix used to compute GMM estimates  
`e(S)` moment covariance matrix used to compute GMM variance–covariance matrix  
`e(V)` variance–covariance matrix of the estimators  
`e(V_modelbased)` model-based variance

#### Functions

`e(sample)` marks estimation sample

## Methods and formulas

`ivregress` is implemented as an ado-file.

Methods and formulas are presented under the following headings:

*Notation*  
*2SLS and LIML estimators*  
*GMM estimator*

## Notation

Items printed in lowercase and italicized (e.g.,  $x$ ) are scalars. Items printed in lowercase and boldfaced (e.g.,  $\mathbf{x}$ ) are vectors. Items printed in uppercase and boldfaced (e.g.,  $\mathbf{X}$ ) are matrices.

The model is

$$\mathbf{y} = \mathbf{Y}\beta_1 + \mathbf{X}_1\beta_2 + \mathbf{u} = \mathbf{X}\beta + \mathbf{u}$$

$$\mathbf{Y} = \mathbf{X}_1\boldsymbol{\Pi}_1 + \mathbf{X}_2\boldsymbol{\Pi}_2 + \mathbf{v} = \mathbf{Z}\boldsymbol{\Pi} + \mathbf{V}$$

where  $\mathbf{y}$  is an  $N \times 1$  vector of the left-hand-side variable;  $N$  is the sample size;  $\mathbf{Y}$  is an  $N \times p$  matrix of  $p$  endogenous regressors;  $\mathbf{X}_1$  is an  $N \times k_1$  matrix of  $k_1$  included exogenous regressors;  $\mathbf{X}_2$  is an  $N \times k_2$  matrix of  $k_2$  excluded exogenous variables,  $\mathbf{X} = [\mathbf{Y} \ \mathbf{X}_1]$ ,  $\mathbf{Z} = [\mathbf{X}_1 \ \mathbf{X}_2]$ ;  $\mathbf{u}$  is an  $N \times 1$  vector of errors;  $\mathbf{V}$  is an  $N \times p$  matrix of errors;  $\beta = [\beta_1 \ \beta_2]$  is a  $k = (p + k_1) \times 1$  vector of parameters; and  $\boldsymbol{\Pi}$  is a  $(k_1 + k_2) \times p$  vector of parameters. If a constant term is included in the model, then one column of  $\mathbf{X}_1$  contains all ones.

Let  $\mathbf{v}$  be a column vector of weights specified by the user. If no weights are specified,  $\mathbf{v} = \mathbf{1}$ . Let  $\mathbf{w}$  be a column vector of normalized weights. If no weights are specified or if the user specified `fweights` or `iweights`,  $\mathbf{w} = \mathbf{v}$ ; otherwise,  $\mathbf{w} = \{\mathbf{v}/(\mathbf{1}'\mathbf{v})\}(\mathbf{1}'\mathbf{1})$ . Let  $\mathbf{D}$  denote the  $N \times N$  matrix with  $\mathbf{w}$  on the main diagonal and zeros elsewhere. If no weights are specified,  $\mathbf{D}$  is the identity matrix.

The weighted number of observations  $n$  is defined as  $\mathbf{1}'\mathbf{w}$ . For `iweights`, this is truncated to an integer. The *sum of the weights* is  $\mathbf{1}'\mathbf{v}$ . Define  $c = 1$  if there is a constant in the regression and zero otherwise.

The order condition for identification requires that  $k_2 \geq p$ : the number of excluded exogenous variables must be at least as great as the number of endogenous regressors.

In the following formulas, if weights are specified,  $\mathbf{X}_1'\mathbf{X}_1$ ,  $\mathbf{X}'\mathbf{X}$ ,  $\mathbf{X}'\mathbf{y}$ ,  $\mathbf{y}'\mathbf{y}$ ,  $\mathbf{Z}'\mathbf{Z}$ ,  $\mathbf{Z}'\mathbf{X}$ , and  $\mathbf{Z}'\mathbf{y}$  are replaced with  $\mathbf{X}_1'\mathbf{D}\mathbf{X}_1$ ,  $\mathbf{X}'\mathbf{D}\mathbf{X}$ ,  $\mathbf{X}'\mathbf{D}\mathbf{y}$ ,  $\mathbf{y}'\mathbf{D}\mathbf{y}$ ,  $\mathbf{Z}'\mathbf{D}\mathbf{Z}$ ,  $\mathbf{Z}'\mathbf{D}\mathbf{X}$ , and  $\mathbf{Z}'\mathbf{D}\mathbf{y}$ , respectively. We suppress the  $\mathbf{D}$  below to simplify the notation.

## 2SLS and LIML estimators

Define the  $\kappa$ -class estimator of  $\beta$  as

$$\mathbf{b} = \{\mathbf{X}'(\mathbf{I} - \kappa\mathbf{M}_Z)^{-1}\mathbf{X}\}^{-1}\mathbf{X}'(\mathbf{I} - \kappa\mathbf{M}_Z)^{-1}\mathbf{y}$$

where  $\mathbf{M}_Z = \mathbf{I} - \mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'$ . The 2SLS estimator results from setting  $\kappa = 1$ . The LIML estimator results from selecting  $\kappa$  to be the minimum eigenvalue of  $(\mathbf{Y}'\mathbf{M}_Z\mathbf{Y})^{-1/2}\mathbf{Y}'\mathbf{M}_{X_1}\mathbf{Y}(\mathbf{Y}'\mathbf{M}_Z\mathbf{Y})^{-1/2}$ , where  $\mathbf{M}_{X_1} = \mathbf{I} - \mathbf{X}_1(\mathbf{X}_1'\mathbf{X}_1)^{-1}\mathbf{X}_1'$ .

The total sum of squares (TSS) equals  $\mathbf{y}'\mathbf{y}$  if there is no intercept and  $\mathbf{y}'\mathbf{y} - \{(\mathbf{1}'\mathbf{y})^2/n\}$  otherwise. The degrees of freedom are  $n - c$ . The error sum of squares (ESS) is defined as  $\mathbf{y}'\mathbf{y} - 2\mathbf{b}\mathbf{X}'\mathbf{y} + \mathbf{b}'\mathbf{X}'\mathbf{X}\mathbf{b}$ . The model sum of squares (MSS) equals TSS - ESS. The degrees of freedom are  $k - c$ .

The mean squared error,  $s^2$ , is defined as  $\text{ESS}/(n - k)$  if `small` is specified and  $\text{ESS}/n$  otherwise. The root mean squared error is  $s$ , its square root.

If  $c = 1$  and `small` is not specified, a Wald statistic,  $W$ , of the joint significance of the  $k - 1$  parameters of  $\beta$  except the constant term is calculated;  $W \sim \chi^2(k - 1)$ . If  $c = 1$  and `small` is specified, then an  $F$  statistic is calculated as  $F = W/(k - 1)$ ;  $F \sim F(k - 1, n - k)$ .

The  $R$ -squared is defined as  $R^2 = 1 - \text{ESS}/\text{TSS}$ .

The adjusted  $R$ -squared is  $R_a^2 = 1 - (1 - R^2)(n - c)/(n - k)$ .

If `robust` is not specified, then  $\text{Var}(\mathbf{b}) = s^2\{\mathbf{X}'(\mathbf{I} - \kappa\mathbf{M}_Z)^{-1}\mathbf{X}\}^{-1}$ . For a discussion of robust variance estimates in regression and regression with instrumental variables, see [Methods and formulas](#) in [\[R\] regress](#). If `small` is not specified, then  $k = 0$  in the formulas given there.

This command also supports estimation with survey data. For details on VCEs with survey data, see [\[SVY\] variance estimation](#).

## GMM estimator

We obtain an initial consistent estimate of  $\beta$  by using the 2SLS estimator; see above. Using this estimate of  $\beta$ , we compute the weighting matrix  $\mathbf{W}$  and calculate the GMM estimator

$$\mathbf{b}_{\text{GMM}} = \{\mathbf{X}'\mathbf{Z}\mathbf{W}\mathbf{Z}'\mathbf{X}\}^{-1}\mathbf{X}'\mathbf{Z}\mathbf{W}\mathbf{Z}'\mathbf{y}$$

The variance of  $\mathbf{b}_{\text{GMM}}$  is

$$\text{Var}(\mathbf{b}_{\text{GMM}}) = n\{\mathbf{X}'\mathbf{Z}\mathbf{W}\mathbf{Z}'\mathbf{X}\}^{-1}\mathbf{X}'\mathbf{Z}\mathbf{W}\widehat{\mathbf{S}}\mathbf{W}\mathbf{Z}'\mathbf{X}\{\mathbf{X}'\mathbf{Z}\mathbf{W}\mathbf{Z}'\mathbf{X}\}^{-1}$$

$\text{Var}(\mathbf{b}_{\text{GMM}})$  is of the sandwich form **DMD**; see [\[P\] \\_robust](#). If the user specifies the `small` option, `ivregress` implements a small-sample adjustment by multiplying the VCE by  $N/(N - k)$ .

If `vce(unadjusted)` is specified, then we set  $\widehat{\mathbf{S}} = \mathbf{W}^{-1}$  and the VCE reduces to the “optimal” GMM variance estimator

$$\text{Var}(\beta_{\text{GMM}}) = n\{\mathbf{X}'\mathbf{Z}\mathbf{W}\mathbf{Z}'\mathbf{X}\}^{-1}$$

However, if  $\mathbf{W}^{-1}$  is not a good estimator of  $E(\mathbf{z}_i\mathbf{u}_i\mathbf{u}_i'\mathbf{z}_i')$ , then the optimal GMM estimator is inefficient, and inference based on the optimal variance estimator could be misleading.

$\mathbf{W}$  is calculated using the residuals from the initial 2SLS estimates, whereas  $\mathbf{S}$  is estimated using the residuals based on  $\mathbf{b}_{\text{GMM}}$ . The `wmatrix()` option affects the form of  $\mathbf{W}$ , whereas the `vce()` option affects the form of  $\mathbf{S}$ . Except for different residuals being used, the formulas for  $\mathbf{W}^{-1}$  and  $\mathbf{S}$  are identical, so we focus on estimating  $\mathbf{W}^{-1}$ .

If `wmatrix(unadjusted)` is specified, then

$$\mathbf{W}^{-1} = \frac{s^2}{n} \sum_i \mathbf{z}_i \mathbf{z}_i'$$

where  $s^2 = \sum_i u_i^2/n$ . This weight matrix is appropriate if the errors are homoskedastic.

If `wmatrix(robust)` is specified, then

$$\mathbf{W}^{-1} = \frac{1}{n} \sum_i u_i^2 \mathbf{z}_i \mathbf{z}_i'$$

which is appropriate if the errors are heteroskedastic.

If `wmatrix(cluster clustvar)` is specified, then

$$\mathbf{W}^{-1} = \frac{1}{n} \sum_c \mathbf{q}_c \mathbf{q}_c'$$

where  $c$  indexes clusters,

$$\mathbf{q}_c = \sum_{i \in c_j} u_i \mathbf{z}_i$$

and  $c_j$  denotes the  $j$ th cluster.

If `wmatrix(hac kernel [#])` is specified, then

$$\mathbf{W}^{-1} = \frac{1}{n} \sum_i u_i^2 \mathbf{z}_i \mathbf{z}_i' + \frac{1}{n} \sum_{l=1}^{l=n-1} \sum_{i=l+1}^{i=n} K(l, m) u_i u_{i-l} (\mathbf{z}_i \mathbf{z}_{i-l}' + \mathbf{z}_{i-l} \mathbf{z}_i')$$

where  $m = \#$  if  $\#$  is specified and  $m = n - 2$  otherwise. Define  $z = l/(m + 1)$ . If *kernel* is `nwest`, then

$$K(l, m) = \begin{cases} 1 - z & 0 \leq z \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

If *kernel* is `gallant`, then

$$K(l, m) = \begin{cases} 1 - 6z^2 + 6z^3 & 0 \leq z \leq 0.5 \\ 2(1 - z)^3 & 0.5 < z \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

If *kernel* is `quadraticspectral`, then

$$K(l, m) = \begin{cases} 1 & z = 0 \\ 3 \{ \sin(\theta)/\theta - \cos(\theta) \} / \theta^2 & \text{otherwise} \end{cases}$$

where  $\theta = 6\pi z/5$ .

If `wmatrix(hac kernel opt)` is specified, then `ivregress` uses Newey and West's (1994) automatic lag-selection algorithm, which proceeds as follows. Define  $\mathbf{h}$  to be a  $(k_1 + k_2) \times 1$  vector containing ones in all rows except for the row corresponding to the constant term (if present); that row contains a zero. Define



$$\begin{aligned}
 f_i &= (u_i \mathbf{z}_i) \mathbf{h} \\
 \hat{\sigma}_j &= \frac{1}{n} \sum_{i=j+1}^n f_i f_{i-j} \quad j = 0, \dots, m^* \\
 \hat{s}^{(q)} &= 2 \sum_{j=1}^{m^*} \hat{\sigma}_j j^q \\
 \hat{s}^{(0)} &= \hat{\sigma}_0 + 2 \sum_{j=1}^{m^*} \hat{\sigma}_j \\
 \hat{\gamma} &= c_\gamma \left\{ \left( \frac{\hat{s}^{(q)}}{\hat{s}^{(0)}} \right)^2 \right\}^{1/2q+1} \\
 m &= \hat{\gamma} n^{1/(2q+1)}
 \end{aligned}$$

where  $q$ ,  $m^*$ , and  $c_\gamma$  depend on the kernel specified:

| Kernel             | $q$ | $m^*$                             | $c_\gamma$ |
|--------------------|-----|-----------------------------------|------------|
| Bartlett           | 1   | $\text{int} \{20(T/100)^{2/9}\}$  | 1.1447     |
| Parzen             | 2   | $\text{int} \{20(T/100)^{4/25}\}$ | 2.6614     |
| Quadratic spectral | 2   | $\text{int} \{20(T/100)^{2/25}\}$ | 1.3221     |

where  $\text{int}(x)$  denotes the integer obtained by truncating  $x$  toward zero. For the Bartlett and Parzen kernels, the optimal lag is  $\min\{\text{int}(m), m^*\}$ . For the quadratic spectral, the optimal lag is  $\min\{m, m^*\}$ .

If `center` is specified, when computing weighting matrices `ivregress` replaces the term  $u_i \mathbf{z}_i$  in the formulas above with  $u_i \mathbf{z}_i - \bar{u}\bar{\mathbf{z}}$ , where  $\bar{u}\bar{\mathbf{z}} = \sum_i u_i \mathbf{z}_i / N$ .

## References

- Andrews, D. W. K. 1991. Heteroskedasticity and autocorrelation consistent covariance matrix estimation. *Econometrica* 59: 817–858.
- Angrist, J. D., and J.-S. Pischke. 2009. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton, NJ: Princeton University Press.
- Basman, R. L. 1957. A generalized classical method of linear estimation of coefficients in a structural equation. *Econometrica* 25: 77–83.
- Baum, C. F. 2006. *An Introduction to Modern Econometrics Using Stata*. College Station, TX: Stata Press.
- Baum, C. F., M. E. Schaffer, and S. Stillman. 2003. Instrumental variables and GMM: Estimation and testing. *Stata Journal* 3: 1–31.
- . 2007. Enhanced routines for instrumental variables/generalized method of moments estimation and testing. *Stata Journal* 7: 465–506.
- Cameron, A. C., and P. K. Trivedi. 2005. *Microeconometrics: Methods and Applications*. New York: Cambridge University Press.
- . 2009. *Microeconometrics Using Stata*. College Station, TX: Stata Press.

- Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.
- . 2004. *Econometric Theory and Methods*. New York: Oxford University Press.
- Gallant, A. R. 1987. *Nonlinear Statistical Models*. New York: Wiley.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- Hall, A. R. 2005. *Generalized Method of Moments*. Oxford: Oxford University Press.
- Hansen, L. P. 1982. Large sample properties of generalized method of moments estimators. *Econometrica* 50: 1029–1054.
- Hayashi, F. 2000. *Econometrics*. Princeton, NJ: Princeton University Press.
- Judge, G. G., W. E. Griffiths, R. C. Hill, H. Lütkepohl, and T.-C. Lee. 1985. *The Theory and Practice of Econometrics*. 2nd ed. New York: Wiley.
- Kmenta, J. 1997. *Elements of Econometrics*. 2nd ed. Ann Arbor: University of Michigan Press.
- Koopmans, T. C., and W. C. Hood. 1953. *Studies in Econometric Method*. New York: Wiley.
- Koopmans, T. C., and J. Marschak. 1950. *Statistical Inference in Dynamic Economic Models*. New York: Wiley.
- Newey, W. K., and K. D. West. 1987. A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica* 55: 703–708.
- . 1994. Automatic lag selection in covariance matrix estimation. *Review of Economic Studies* 61: 631–653.
- Nichols, A. 2007. Causal inference with observational data. *Stata Journal* 7: 507–541.
- Poi, B. P. 2006. Jackknife instrumental variables estimation in Stata. *Stata Journal* 6: 364–376.
- Stock, J. H., and M. W. Watson. 2007. *Introduction to Econometrics*. 2nd ed. Boston: Addison–Wesley.
- Stock, J. H., J. H. Wright, and M. Yogo. 2002. A survey of weak instruments and weak identification in generalized method of moments. *Journal of Business and Economic Statistics* 20: 518–529.
- Theil, H. 1953. *Repeated Least Squares Applied to Complete Equation Systems*. Mimeograph from the Central Planning Bureau, The Hague.
- Wooldridge, J. M. 2002. *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA: MIT Press.
- . 2009. *Introductory Econometrics: A Modern Approach*. 4th ed. Cincinnati, OH: South-Western.
- Wright, P. G. 1928. *The Tariff on Animal and Vegetable Oils*. New York: Macmillan.

## Also see

- [R] [ivregress postestimation](#) — Postestimation tools for ivregress
- [R] [gmm](#) — Generalized method of moments estimation
- [R] [ivprobit](#) — Probit model with continuous endogenous regressors
- [R] [ivtobit](#) — Tobit model with continuous endogenous regressors
- [R] [regress](#) — Linear regression
- [XT] [xtivreg](#) — Instrumental variables and two-stage least squares for panel-data models
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are of special interest after `ivregress`:

| command                       | description                                   |
|-------------------------------|-----------------------------------------------|
| <code>estat endogenous</code> | perform tests of endogeneity                  |
| <code>estat firststage</code> | report “first-stage” regression statistics    |
| <code>estat overid</code>     | perform tests of overidentifying restrictions |

These commands are not appropriate after the `svy` prefix.

For information about these commands, see below.

The following postestimation commands are also available:

| command                  | description                                                                                         |
|--------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>       | VCE and estimation sample summary                                                                   |
| <code>estat (svy)</code> | postestimation statistics for survey data                                                           |
| <code>estimates</code>   | cataloging estimation results                                                                       |
| <code>hausman</code>     | Hausman’s specification test                                                                        |
| <code>lincom</code>      | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>margins</code>     | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>       | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>     | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>   | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>        | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>      | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Special-interest postestimation commands

`estat endogenous` performs tests to determine whether endogenous regressors in the model are in fact exogenous. After GMM estimation, the *C* (difference-in-Sargan) statistic is reported. After 2SLS estimation with an unadjusted VCE, the Durbin (1954) and Wu–Hausman (Wu 1974; Hausman 1978) statistics are reported. After 2SLS estimation with a robust VCE, Wooldridge’s (1995) robust score test and a robust regression-based test are reported. In all cases, if the test statistic is significant, then the variables being tested must be treated as endogenous. `estat endogenous` is not available after LIML estimation.

`estat firststage` reports various statistics that measure the relevance of the excluded exogenous variables. By default, whether the equation has one or more than one endogenous regressor determines what statistics are reported.

`estat overid` performs tests of overidentifying restrictions. If the 2SLS estimator was used, Sargan’s (1958) and Basmann’s (1960)  $\chi^2$  tests are reported, as is Wooldridge’s (1995) robust score test; if the LIML estimator was used, Anderson and Rubin’s (1950)  $\chi^2$  test and Basmann’s  $F$  test are reported; and if the GMM estimator was used, Hansen’s (1982)  $J$  statistic  $\chi^2$  test is reported. A statistically significant test statistic always indicates that the instruments may not be valid.

## Syntax for predict

```
predict [type] newvar [if] [in] [, statistic]

predict [type] { stub*| newvarlist } [if] [in], scores
```

| statistic                             | description                                 |
|---------------------------------------|---------------------------------------------|
| Main                                  |                                             |
| <code>xb</code>                       | linear prediction; the default              |
| <code>_residuals</code>               | residuals                                   |
| <code>stdp</code>                     | standard error of the prediction            |
| <code>stdf</code>                     | standard error of the forecast              |
| <code>pr(<i>a</i>,<i>b</i>)</code>    | $\Pr(a < y_j < b)$                          |
| <code>e(<i>a</i>,<i>b</i>)</code>     | $E(y_j   a < y_j < b)$                      |
| <code>ystar(<i>a</i>,<i>b</i>)</code> | $E(y_j^*), y_j^* = \max\{a, \min(y_j, b)\}$ |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

`stdf` is not allowed with `svy` estimation results.

where *a* and *b* may be numbers or variables; *a* missing (*a* ≥ .) means  $-\infty$ , and *b* missing (*b* ≥ .) means  $+\infty$ ; see [U] 12.2.1 Missing values.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

- `xb`, the default, calculates the linear prediction.
- `residuals` calculates the residuals, that is,  $y_j - \mathbf{x}_j \mathbf{b}$ . These are based on the estimated equation when the observed values of the endogenous variables are used—not the projections of the instruments onto the endogenous variables.
- `stdp` calculates the standard error of the prediction, which can be thought of as the standard error of the predicted expected value or mean for the observation’s covariate pattern. This is also referred to as the standard error of the fitted value.

**stdf** calculates the standard error of the forecast, which is the standard error of the point prediction for 1 observation. It is commonly referred to as the standard error of the future or forecast value. By construction, the standard errors produced by **stdf** are always larger than those produced by **stdp**; see [Methods and formulas](#) in [R] **regress**.

**pr(*a*,*b*)** calculates  $\Pr(a < \mathbf{x}_j \mathbf{b} + u_j < b)$ , the probability that  $y_j | \mathbf{x}_j$  would be observed in the interval  $(a, b)$ .

*a* and *b* may be specified as numbers or variable names; *lb* and *ub* are variable names;

**pr(20,30)** calculates  $\Pr(20 < \mathbf{x}_j \mathbf{b} + u_j < 30)$ ;

**pr(*lb*,*ub*)** calculates  $\Pr(lb < \mathbf{x}_j \mathbf{b} + u_j < ub)$ ; and

**pr(20,*ub*)** calculates  $\Pr(20 < \mathbf{x}_j \mathbf{b} + u_j < ub)$ .

*a* missing (*a* ≥ .) means  $-\infty$ ; **pr(.,30)** calculates  $\Pr(-\infty < \mathbf{x}_j \mathbf{b} + u_j < 30)$ ;

**pr(*lb*,30)** calculates  $\Pr(-\infty < \mathbf{x}_j \mathbf{b} + u_j < 30)$  in observations for which *lb* ≥ .

and calculates  $\Pr(lb < \mathbf{x}_j \mathbf{b} + u_j < 30)$  elsewhere.

*b* missing (*b* ≥ .) means  $+\infty$ ; **pr(20,.)** calculates  $\Pr(+\infty > \mathbf{x}_j \mathbf{b} + u_j > 20)$ ;

**pr(20,*ub*)** calculates  $\Pr(+\infty > \mathbf{x}_j \mathbf{b} + u_j > 20)$  in observations for which *ub* ≥ .

and calculates  $\Pr(20 < \mathbf{x}_j \mathbf{b} + u_j < ub)$  elsewhere.

**e(*a*,*b*)** calculates  $E(\mathbf{x}_j \mathbf{b} + u_j \mid a < \mathbf{x}_j \mathbf{b} + u_j < b)$ , the expected value of  $y_j | \mathbf{x}_j$  conditional on  $y_j | \mathbf{x}_j$  being in the interval  $(a, b)$ , meaning that  $y_j | \mathbf{x}_j$  is censored.

*a* and *b* are specified as they are for **pr()**.

**ystar(*a*,*b*)** calculates  $E(y_j^*)$ , where  $y_j^* = a$  if  $\mathbf{x}_j \mathbf{b} + u_j \leq a$ ,  $y_j^* = b$  if  $\mathbf{x}_j \mathbf{b} + u_j \geq b$ , and  $y_j^* = \mathbf{x}_j \mathbf{b} + u_j$  otherwise, meaning that  $y_j^*$  is truncated. *a* and *b* are specified as they are for **pr()**.

**scores** calculates the scores for the model. A new score variable is created for each endogenous regressor, as well as an equation-level score that applies to all exogenous variables and constant term (if present).

## Syntax for estat endogenous

```
estat endogenous [varlist] [, lags(#) forceweights forcenonrobust]
```

## Menu

Statistics > Postestimation > Reports and statistics

## Options for estat endogenous

**lags(#)** specifies the number of lags to use for prewhitening when computing the heteroskedasticity- and autocorrelation-consistent (HAC) version of the score test of endogeneity. Specifying **lags(0)** requests no prewhitening. This option is valid only when the model was fit via 2SLS and an HAC covariance matrix was requested when the model was fit. The default is **lags(1)**.

**forceweights** requests that the tests of endogeneity be computed even though **aweight**s, **pweight**s, or **iweight**s were used in the previous 2SLS estimation. By default, these tests are conducted only after unweighted or frequency-weighted estimation. The reported critical values may be inappropriate for weighted data, so the user must determine whether the critical values are appropriate for a given application.

`forcenonrobust` requests that the Durbin and Wu–Hausman tests be performed after 2SLS estimation even though a robust VCE was used at estimation time. This option is available only if the model was fit by 2SLS.

## Syntax for estat firststage

```
estat firststage [, all forcenonrobust]
```

## Menu

Statistics > Postestimation > Reports and statistics

## Options for estat firststage

`all` requests that all first-stage goodness-of-fit statistics be reported regardless of whether the model contains one or more endogenous regressors. By default, if the model contains one endogenous regressor, then the first-stage  $R^2$ , adjusted  $R^2$ , partial  $R^2$ , and  $F$  statistics are reported, whereas if the model contains multiple endogenous regressors, then Shea's partial  $R^2$  and adjusted partial  $R^2$  are reported instead.

`forcenonrobust` requests that the minimum eigenvalue statistic and its critical values be reported even though a robust VCE was used at estimation time. The reported critical values assume that the errors are independent and identically distributed (i.i.d.) normal, so the user must determine whether the critical values are appropriate for a given application.

## Syntax for estat overid

```
estat overid [, lags(#) forceweights forcenonrobust]
```

## Menu

Statistics > Postestimation > Reports and statistics

## Options for estat overid

`lags(#)` specifies the number of lags to use for prewhitening when computing the heteroskedasticity- and autocorrelation-consistent (HAC) version of the score test of overidentifying restrictions. Specifying `lags(0)` requests no prewhitening. This option is valid only when the model was fit via 2SLS and an HAC covariance matrix was requested when the model was fit. The default is `lags(1)`.

`forceweights` requests that the tests of overidentifying restrictions be computed even though `aweight`s, `pweight`s, or `iweight`s were used in the previous estimation. By default, these tests are conducted only after unweighted or frequency-weighted estimation. The reported critical values may be inappropriate for weighted data, so the user must determine whether the critical values are appropriate for a given application.

`forcenonrobust` requests that the Sargan and Basman tests of overidentifying restrictions be performed after 2SLS or LIML estimation even though a robust VCE was used at estimation time. These tests assume that the errors are i.i.d. normal, so the user must determine whether the critical values are appropriate for a given application.

## Remarks

Remarks are presented under the following headings:

*estat endogenous*  
*estat firststage*  
*estat overid*

### estat endogenous

A natural question to ask is whether a variable presumed to be endogenous in the previously fit model could instead be treated as exogenous. If the endogenous regressors are in fact exogenous, then the OLS estimator is more efficient; and depending on the strength of the instruments and other factors, the sacrifice in efficiency by using an instrumental-variables estimator can be significant. Thus, unless an instrumental-variables estimator is really needed, OLS should be used instead. `estat endogenous` provides several tests of endogeneity after 2SLS and GMM estimation.

#### ► Example 1

In [example 1](#) of [\[R\]](#) `ivregress`, we fit a model of the average rental rate for housing in a state as a function of the percentage of the population living in urban areas and the average value of houses. We treated `hsngval` as endogenous because unanticipated shocks that affect rental rates probably affect house prices as well. We used family income and region dummies as additional instruments for `hsngval`. Here we test whether we could treat `hsngval` as exogenous.

```
. use http://www.stata-press.com/data/r11/hsng
(1980 Census housing data)
. ivregress 2sls rent pcturban (hsngval = faminc i.region)
(output omitted)
. estat endogenous

Tests of endogeneity
Ho: variables are exogenous
Durbin (score) chi2(1) = 12.8473 (p = 0.0003)
Wu-Hausman F(1,46) = 15.9067 (p = 0.0002)
```

Because we did not specify any variable names after the `estat endogenous` command, Stata by default tested all the endogenous regressors (namely, `hsngval`) in our model. The null hypothesis of the Durbin and Wu–Hausman tests is that the variable under consideration can be treated as exogenous. Here both test statistics are highly significant, so we reject the null of exogeneity; we must continue to treat `hsngval` as endogenous.

◀

The difference between the Durbin and Wu–Hausman tests of endogeneity is that the former uses an estimate of the error term’s variance based on the model assuming the variables being tested are exogenous, while the latter uses an estimate of the error variance based on the model assuming the variables being tested are endogenous. Under the null hypothesis that the variables being tested are exogenous, both estimates of the error variance are consistent. What we label the Wu–Hausman statistic is Wu’s (1974) “ $T_2$ ” statistic, which Hausman (1978) showed can be calculated very easily via linear regression. Baum, Schaffer, and Stillman (2003, 2007) provide a lucid discussion of these tests.

When you fit a model with multiple endogenous regressors, you can test the exogeneity of a subset of the regressors while continuing to treat the others as endogenous. For example, say you have three endogenous regressors, `y1`, `y2`, and `y3`, and you fit your model by typing

```
. ivregress depvar ... (y1 y2 y3 = ...)
```

Suppose you are confident that `y1` must be treated as endogenous, but you are undecided about `y2` and `y3`. To test whether `y2` and `y3` can be treated as exogenous, you would type

```
. estat endogenous y2 y3
```

The Durbin and Wu–Hausman tests assume that the error term is i.i.d. Therefore, if you requested a robust VCE at estimation time, `estat endogenous` will instead report Wooldridge’s (1995) score test and a regression-based test of exogeneity. Both these tests can tolerate heteroskedastic and autocorrelated errors, while only the regression-based test is amenable to clustering.

## ► Example 2

We refit our housing model, requesting robust standard errors, and then test the exogeneity of `hsngval`:

```
. use http://www.stata-press.com/data/r11/hsng
(1980 Census housing data)

. ivregress 2sls rent pcturban (hsngval = faminc i.region), vce(robust)
(output omitted)

. estat endogenous

Tests of endogeneity
Ho: variables are exogenous

Robust score chi2(1) = 2.10428 (p = 0.1469)
Robust regression F(1,46) = 4.31101 (p = 0.0435)
```

Wooldridge’s score test does not reject the null hypothesis that `hsngval` is exogenous at conventional significance levels ( $p = 0.1469$ ). However, the regression-based test does reject the null hypothesis at the 5% significance level ( $p = 0.0435$ ). Typically, these two tests yield the same conclusion; the fact that our dataset has only 50 observations could be contributing to the discrepancy. Here we would be inclined to continue to treat `hsngval` as endogenous. Even if `hsngval` is exogenous, the 2SLS estimates are still consistent. On the other hand, if `hsngval` is in fact endogenous, the OLS estimates would not be consistent. Moreover, as we will see in our discussion of the `estat overid` command, our additional instruments may be invalid. To test whether an endogenous variable can be treated as exogenous, we must have a valid set of instruments to use to fit the model in the first place!

◄

Unlike the Durbin and Wu–Hausman tests, Wooldridge’s score and the regression-based tests do not allow you to test a subset of the endogenous regressors in the model; you can test only whether all the endogenous regressors are in fact exogenous.

After GMM estimation, `estat endogenous` calculates what Hayashi (2000, 220) calls the  $C$  statistic, also known as the difference-in-Sargan statistic. The  $C$  statistic can be made robust to heteroskedasticity, autocorrelation, and clustering; and the version reported by `estat endogenous` is determined by the weight matrix requested via the `wmatrix()` option used when fitting the model with `ivregress`. Additionally, the test can be used to determine the exogeneity of a subset of the endogenous regressors, regardless of the type of weight matrix used.

If you fit your model using the LIML estimator, you can use the `hausman` command to carry out a traditional Hausman (1978) test between the OLS and LIML estimates.



## estat firststage

For an excluded exogenous variable to be a valid instrument, it must be sufficiently correlated with the included endogenous regressors but uncorrelated with the error term. In recent decades, researchers have paid considerable attention to the issue of instruments that are only weakly correlated with the endogenous regressors. In such cases, the usual 2SLS, GMM, and LIML estimators are biased toward the OLS estimator, and inference based on the standard errors reported by, for example, `ivregress` can be severely misleading. For more information on the theory behind instrumental-variables estimation with weak instruments, see [Nelson and Startz \(1990\)](#); [Staiger and Stock \(1997\)](#); [Hahn and Hausman \(2003\)](#); the survey article by [Stock, Wright, and Yogo \(2002\)](#); and [Angrist and Pischke \(2009, chap. 4\)](#).

When the instruments are only weakly correlated with the endogenous regressors, some Monte Carlo evidence suggests that the LIML estimator performs better than the 2SLS and GMM estimators; see, for example, [Poi \(2006\)](#) and [Stock, Wright, and Yogo \(2002\)](#) (and the papers cited therein). On the other hand, the LIML estimator often results in confidence intervals that are somewhat larger than those from the 2SLS estimator.

Moreover, using more instruments is not a solution, because the biases of instrumental-variables estimators increase with the number of instruments. See [Hahn and Hausman \(2003\)](#).

`estat firststage` produces several statistics for judging the explanatory power of the instruments and is most easily explained with examples.

### ► Example 3

Again building on the model fit in [example 1](#) of [\[R\] ivregress](#), we now explore the degree of correlation between the additional instruments `faminc`, `2.region`, `3.region`, and `4.region` and the endogenous regressor `hsngval`:

```
. use http://www.stata-press.com/data/r11/hsng
(1980 Census housing data)
. ivregress 2sls rent pcturban (hsngval = faminc i.region)
(output omitted)
. estat firststage
```

First-stage regression summary statistics

| Variable | R-sq.  | Adjusted<br>R-sq. | Partial<br>R-sq. | F(4,44) | Prob > F |
|----------|--------|-------------------|------------------|---------|----------|
| hsngval  | 0.6908 | 0.6557            | 0.5473           | 13.2978 | 0.0000   |

Minimum eigenvalue statistic = 13.2978

|                          |                             |   |
|--------------------------|-----------------------------|---|
| Critical Values          | # of endogenous regressors: | 1 |
| Ho: Instruments are weak | # of excluded instruments:  | 4 |

|                                   | 5%    | 10%   | 20%   | 30%  |
|-----------------------------------|-------|-------|-------|------|
| 2SLS relative bias                | 16.85 | 10.27 | 6.71  | 5.34 |
| 2SLS Size of nominal 5% Wald test | 10%   | 15%   | 20%   | 25%  |
| LIML Size of nominal 5% Wald test | 24.58 | 13.96 | 10.26 | 8.31 |
|                                   | 5.44  | 3.87  | 3.30  | 2.98 |

To understand these results, recall that the first-stage regression is

$$\text{hsngval}_i = \pi_0 + \pi_1 \text{pcturban}_i + \pi_2 \text{faminc} + \pi_3 2.\text{region} + \pi_4 3.\text{region} + \pi_5 4.\text{region} + v_i$$

where  $v_i$  is an error term. The column marked “R-sq.” is the simple  $R^2$  from fitting the first-stage regression by OLS, and the column marked “Adjusted R-sq.” is the adjusted  $R^2$  from that regression. Higher values purportedly indicate stronger instruments, and instrumental-variables estimators exhibit less bias when the instruments are strongly correlated with the endogenous variable.

Looking at just the  $R^2$  and adjusted  $R^2$  can be misleading, however. If `hsgval` were strongly correlated with the included exogenous variable `pcturban` but only weakly correlated with the additional instruments, then these statistics could be large even though a weak-instrument problem is present.

The partial  $R^2$  statistic measures the correlation between `hsgval` and the additional instruments after *partialling out* the effect of `pcturban`. Unlike the  $R^2$  and adjusted  $R^2$  statistics, the partial  $R^2$  statistic will not be inflated because of strong correlation between `hsgval` and `pcturban`. [Bound, Jaeger, and Baker \(1995\)](#) and others have promoted using this statistic.

The column marked “F(4, 44)” is an  $F$  statistic for the joint significance of  $\pi_2$ ,  $\pi_3$ ,  $\pi_4$ , and  $\pi_5$ , the coefficients on the additional instruments. Its  $p$ -value is listed in the column marked “Prob > F”. If the  $F$  statistic is not significant, then the additional instruments have no significant explanatory power for `hsgval` after controlling for the effect of `pcturban`. However, [Hall, Rudebusch, and Wilcox \(1996\)](#) used Monte Carlo simulation to show that simply having an  $F$  statistic that is significant at the typical 5% or 10% level is not sufficient. [Stock, Wright, and Yogo \(2002\)](#) suggest that the  $F$  statistic should exceed 10 for inference based on the 2SLS estimator to be reliable when there is one endogenous regressor.

`estat firststage` also presents the [Cragg and Donald \(1993\)](#) minimum eigenvalue statistic as a further test of weak instruments. [Stock and Yogo \(2005\)](#) discuss two characterizations of weak instruments: first, weak instruments cause instrumental-variables estimators to be biased; second, hypothesis tests of parameters estimated by instrumental-variables estimators may suffer from severe size distortions. The test statistic in our example is 13.30, which is identical to the  $F$  statistic just discussed because our model contains one endogenous regressor.

The null hypothesis of each of Stock and Yogo’s tests is that the set of instruments is weak. To perform these tests, we must first choose either the largest relative bias of the 2SLS estimator we are willing to tolerate or the largest rejection rate of a nominal 5% Wald test we are willing to tolerate. If the test statistic exceeds the critical value, we can conclude that our instruments are not weak.

The row marked “2SLS relative bias” contains critical values for the test that the instruments are weak based on the bias of the 2SLS estimator *relative to* the bias of the OLS estimator. For example, from past experience we might know that the OLS estimate of a parameter  $\beta$  may be 50% too high. Saying that we are willing to tolerate a 10% relative bias means that we are willing to tolerate a bias of the 2SLS estimator no greater than 5% (i.e., 10% of 50%). In our rental rate model, if we are willing to tolerate a 10% relative bias, then we can conclude that our instruments are not weak because the test statistic of 13.30 exceeds the critical value of 10.22. However, if we were willing to tolerate only a relative bias of 5%, we would conclude that our instruments are weak because  $13.30 < 16.85$ .

The rows marked “2SLS Size of nominal 5% Wald test” and “LIML Size of nominal 5% Wald test” contain critical values pertaining to Stock and Yogo’s (2005) second characterization of weak instruments. This characterization defines a set of instruments to be weak if a Wald test at the 5% level can have an actual rejection rate of no more than 10%, 15%, 20%, or 25%. Using the current example, suppose that we are willing to accept a rejection rate of at most 10%. Because  $13.30 < 24.58$ , we cannot reject the null hypothesis of weak instruments. On the other hand, if we use the LIML estimator instead, then we can reject the null hypothesis because  $13.30 > 5.44$ .

## □ Technical note

Stock and Yogo (2005) tabulated critical values for 2SLS relative biases of 5%, 10%, 20%, and 30% for models with 1, 2, or 3 endogenous regressors and between 3 and 30 excluded exogenous variables (instruments). They also provide critical values for worst-case rejection rates of 5%, 10%, 20%, and 25% for nominal 5% Wald tests of the endogenous regressors with 1 or 2 endogenous regressors and between 1 and 30 instruments. If the model previously fit by `ivregress` has more instruments or endogenous regressors than these limits, the critical values are not shown. Stock and Yogo did not consider GMM estimators.

□

When the model being fit contains more than one endogenous regressor, the  $R^2$  and  $F$  statistics described above can overstate the relevance of the excluded instruments. Suppose that there are two endogenous regressors,  $Y_1$  and  $Y_2$ , and that there are two additional instruments,  $z_1$  and  $z_2$ . Say that  $z_1$  is highly correlated with both  $Y_1$  and  $Y_2$  but  $z_2$  is not correlated with either  $Y_1$  or  $Y_2$ . Then the first-stage regression of  $Y_1$  on  $z_1$  and  $z_2$  (along with the included exogenous variables) will produce large  $R^2$  and  $F$  statistics, as will the regression of  $Y_2$  on  $z_1$ ,  $z_2$ , and the included exogenous variables. Nevertheless, the lack of correlation between  $z_2$  and  $Y_1$  and  $Y_2$  is problematic. Here, although the order condition indicates that the model is just identified (the number of excluded instruments equals the number of endogenous regressors), the irrelevance of  $z_2$  implies that the model is in fact not identified. Even if the model is overidentified, including irrelevant instruments can adversely affect the properties of instrumental-variables estimators, because their biases increase as the number of instruments increases.

## ▷ Example 4

`estat firststage` presents different statistics when the model contains multiple endogenous regressors. For illustration, we refit our model of rental rates, assuming that both `hsngval` and `faminc` are endogenously determined. We use `i.region` along with `popden`, a measure of population density, as additional instruments.

```
. ivregress 2sls rent pcturban (hsngval faminc = i.region popden)
(output omitted)
. estat firststage
Shea's partial R-squared
```

| Variable | Shea's<br>Partial R-sq. | Shea's<br>Adj. Partial R-sq. |
|----------|-------------------------|------------------------------|
| hsngval  | 0.3477                  | 0.2897                       |
| faminc   | 0.1893                  | 0.1173                       |

Minimum eigenvalue statistic = 2.51666

|                          |                             |   |
|--------------------------|-----------------------------|---|
| Critical Values          | # of endogenous regressors: | 2 |
| Ho: Instruments are weak | # of excluded instruments:  | 4 |

|                                   | 5%    | 10%  | 20%  | 30%  |
|-----------------------------------|-------|------|------|------|
| 2SLS relative bias                | 11.04 | 7.56 | 5.57 | 4.73 |
| 2SLS Size of nominal 5% Wald test | 16.87 | 9.93 | 7.54 | 6.28 |
| LIML Size of nominal 5% Wald test | 4.72  | 3.39 | 2.99 | 2.79 |

Consider the endogenous regressor `hsngval`. Part of its variation is attributable to its correlation with the other regressors `pcturban` and `faminc`. The other component of `hsngval`'s variation is peculiar to it and orthogonal to the variation in the other regressors. Similarly, we can think of the instruments as predicting the variation in `hsngval` in two ways, one stemming from the fact that the predicted values of `hsngval` are correlated with the predicted values of the other regressors and one from the variation in the predicted values of `hsngval` that is orthogonal to the variation in the predicted values of the other regressors.

What really matters for instrumental-variables estimation is whether the component of `hsngval` that is orthogonal to the other regressors can be explained by the component of the predicted value of `hsngval` that is orthogonal to the predicted values of the other regressors in the model. Shea's (1997) partial  $R^2$  statistic measures this correlation. Because the bias of instrumental-variables estimators increases as more instruments are used, Shea's adjusted partial  $R^2$  statistic is often used instead, as it makes a degrees-of-freedom adjustment for the number of instruments, analogous to the adjusted  $R^2$  measure used in OLS regression. Although what constitutes a "low" value for Shea's partial  $R^2$  depends on the specifics of the model being fit and the data used, these results, taken in isolation, do not strike us as being a particular cause for concern.

However, with this specification the minimum eigenvalue statistic is low. We cannot reject the null hypothesis of weak instruments for either of the characterizations we have discussed.

◀

By default, `estat firststage` determines which statistics to present based on the number of endogenous regressors in the model previously fit. However, you can specify the `all` option to obtain all the statistics.

## □ Technical note

If the previous estimation was conducted using `aweight`s, `pweight`s, or `iweight`s, then the first-stage regression summary statistics are computed using those weights. However, in these cases the minimum eigenvalue statistic and its critical values are not available.

If the previous estimation included a robust VCE, then the first-stage  $F$  statistic is based on a robust VCE as well; for example, if you fit your model with an HAC VCE using the Bartlett kernel and four lags, then the  $F$  statistic reported is based on regression results using an HAC VCE using the Bartlett kernel and four lags. By default, the minimum eigenvalue statistic and its critical values are not displayed. You can use the `forcenonrobust` option to obtain them in these cases; the minimum eigenvalue statistic is computed using the weights, though the critical values reported may not be appropriate.

□

## estat overid

In addition to the requirement that instrumental variables be correlated with the endogenous regressors, the instruments must also be uncorrelated with the structural error term. If the model is overidentified, meaning that the number of additional instruments exceeds the number of endogenous regressors, then we can test whether the instruments are uncorrelated with the error term. If the model is just identified, then we cannot perform a test of overidentifying restrictions.

The estimator you used to fit the model determines which tests of overidentifying restrictions `estat overid` reports. If you used the 2SLS estimator without a robust VCE, `estat overid` reports Sargan's (1958) and Basman's (1960)  $\chi^2$  tests. If you used the 2SLS estimator and requested a robust

VCE, Wooldridge’s robust score test of overidentifying restrictions is performed instead; without a robust VCE, Wooldridge’s test statistic is identical to Sargan’s test statistic. If you used the LIML estimator, `estat overid` reports the Anderson–Rubin (1950) likelihood-ratio test and Basmann’s (1960)  $F$  test. `estat overid` reports Hansen’s (1982)  $J$  statistic if you used the GMM estimator. Davidson and MacKinnon (1993, 235–236) give a particularly clear explanation of the intuition behind tests of overidentifying restrictions. Also see Judge et al. (1985, 614–616) for a summary of tests of overidentifying restrictions for the 2SLS and LIML estimators.

Tests of overidentifying restrictions actually test two different things simultaneously. One, as we have discussed, is whether the instruments are uncorrelated with the error term. The other is that the equation is misspecified and that one or more of the excluded exogenous variables should in fact be included in the structural equation. Thus a significant test statistic could represent either an invalid instrument or an incorrectly specified structural equation.

## ► Example 5

Here we refit the model that treated just `hsngval` as endogenous using 2SLS, and then we perform tests of overidentifying restrictions:

```
. ivregress 2sls rent pcturban (hsngval = faminc i.region)
(output omitted)
. estat overid
Tests of overidentifying restrictions:
Sargan (score) chi2(3) = 11.2877 (p = 0.0103)
Basmann chi2(3) = 12.8294 (p = 0.0050)
```

Both test statistics are significant at the 5% test level, which means that either one or more of our instruments are invalid or that our structural model is specified incorrectly.

One possibility is that the error term in our structural model is heteroskedastic. Both Sargan’s and Basmann’s tests assume that the errors are i.i.d.; if the errors are not i.i.d., then these tests are not valid. Here we refit the model by requesting heteroskedasticity-robust standard errors, and then we use `estat overid` to obtain Wooldridge’s score test of overidentifying restrictions, which is robust to heteroskedasticity.

```
. ivregress 2sls rent pcturban (hsngval = faminc i.region), vce(robust)
(output omitted)
. estat overid
Test of overidentifying restrictions:
Score chi2(3) = 6.8364 (p = 0.0773)
```

Here we no longer reject the null hypothesis that our instruments are valid at the 5% significance level, though we do reject the null at the 10% level. You can verify that the robust standard error on the coefficient for `hsngval` is more than twice as large as its nonrobust counterpart and that the robust standard error for `pcturban` is nearly 50% larger.

◀

## □ Technical note

The test statistic for the test of overidentifying restrictions performed after GMM estimation is simply the sample size times the value of the objective function  $Q(\beta_1, \beta_2)$  defined in (5) of [R] `ivregress`, evaluated at the GMM parameter estimates. If the weighting matrix  $\mathbf{W}$  is optimal, meaning that  $\mathbf{W} = \text{Var}(\mathbf{z}_i u_i)$ , then  $Q(\beta_1, \beta_2) \overset{A}{\sim} \chi^2(q)$ , where  $q$  is the number of overidentifying restrictions. However, if the estimated  $\mathbf{W}$  is not optimal, then the test statistic will not have an asymptotic  $\chi^2$  distribution.

Like the Sargan and Basman tests of overidentifying restrictions for the 2SLS estimator, the Anderson–Rubin and Basman tests after LIML estimation are predicated on the errors’ being i.i.d. If the previous LIML results were reported with robust standard errors, then `estat overid` by default issues an error message and refuses to report the Anderson–Rubin and Basman test statistics. You can use the `forcenonrobust` option to override this behavior. You can also use `forcenonrobust` to obtain the Sargan and Basman test statistics after 2SLS estimation with robust standard errors. □

By default, `estat overid` issues an error message if the previous estimation was conducted using `aweight`s, `pweight`s, or `iweight`s. You can use the `forceweights` option to override this behavior, though the test statistics may no longer have the expected  $\chi^2$  distributions.

## Saved results

After 2SLS estimation, `estat endogenous` saves the following in `r()`:

|                             |                                                     |
|-----------------------------|-----------------------------------------------------|
| Scalars                     |                                                     |
| <code>r(durbin)</code>      | Durbin $\chi^2$ statistic                           |
| <code>r(p_durbin)</code>    | $p$ -value for Durbin $\chi^2$ statistic            |
| <code>r(wu)</code>          | Wu–Hausman $F$ statistic                            |
| <code>r(p_wu)</code>        | $p$ -value for Wu–Hausman $F$ statistic             |
| <code>r(df)</code>          | degrees of freedom                                  |
| <code>r(wudf_r)</code>      | denominator degrees of freedom for Wu–Hausman $F$   |
| <code>r(r_score)</code>     | robust score statistic                              |
| <code>r(p_r_score)</code>   | $p$ -value for robust score statistic               |
| <code>r(hac_score)</code>   | HAC score statistic                                 |
| <code>r(p_hac_score)</code> | $p$ -value for HAC score statistic                  |
| <code>r(lags)</code>        | lags used in prewhitening                           |
| <code>r(regF)</code>        | regression-based $F$ statistic                      |
| <code>r(p_regF)</code>      | $p$ -value for regression-based $F$ statistic       |
| <code>r(regFdf_n)</code>    | regression-based $F$ numerator degrees of freedom   |
| <code>r(regFdf_r)</code>    | regression-based $F$ denominator degrees of freedom |

After GMM estimation, `estat endogenous` saves the following in `r()`:

|                     |                                       |
|---------------------|---------------------------------------|
| Scalars             |                                       |
| <code>r(C)</code>   | $C$ $\chi^2$ statistic                |
| <code>r(p_C)</code> | $p$ -value for $C$ $\chi^2$ statistic |
| <code>r(df)</code>  | degrees of freedom                    |

`estat firststage` saves the following in `r()`:

|                               |                                                  |
|-------------------------------|--------------------------------------------------|
| Scalars                       |                                                  |
| <code>r(mineig)</code>        | minimum eigenvalue statistic                     |
| Matrices                      |                                                  |
| <code>r(mineigcv)</code>      | critical values for minimum eigenvalue statistic |
| <code>r(multiresults)</code>  | Shea’s partial $R^2$ statistics                  |
| <code>r(singleresults)</code> | first-stage $R^2$ and $F$ statistics             |

After 2SLS estimation, `estat overid` saves the following in `r()`:

Scalars

|                           |                                           |
|---------------------------|-------------------------------------------|
| <code>r(lags)</code>      | lags used in prewhitening                 |
| <code>r(df)</code>        | $\chi^2$ degrees of freedom               |
| <code>r(score)</code>     | score $\chi^2$ statistic                  |
| <code>r(p_score)</code>   | $p$ -value for score $\chi^2$ statistic   |
| <code>r(basmann)</code>   | Basmann $\chi^2$ statistic                |
| <code>r(p_basmann)</code> | $p$ -value for Basmann $\chi^2$ statistic |
| <code>r(sargan)</code>    | Sargan $\chi^2$ statistic                 |
| <code>r(p_sargan)</code>  | $p$ -value for Sargan $\chi^2$ statistic  |

After LIML estimation, `estat overid` saves the following in `r()`:

Scalars

|                              |                                                  |
|------------------------------|--------------------------------------------------|
| <code>r(ar)</code>           | Anderson–Rubin $\chi^2$ statistic                |
| <code>r(p_ar)</code>         | $p$ -value for Anderson–Rubin $\chi^2$ statistic |
| <code>r(ar_df)</code>        | $\chi^2$ degrees of freedom                      |
| <code>r(basmann)</code>      | Basmann $F$ statistic                            |
| <code>r(p_basmann)</code>    | $p$ -value for Basmann $F$ statistic             |
| <code>r(basmann_df_n)</code> | $F$ numerator degrees of freedom                 |
| <code>r(basmann_df_d)</code> | $F$ denominator degrees of freedom               |

After GMM estimation, `estat overid` saves the following in `r()`:

Scalars

|                           |                                                |
|---------------------------|------------------------------------------------|
| <code>r(HansenJ)</code>   | Hansen’s $J$ $\chi^2$ statistic                |
| <code>r(p_HansenJ)</code> | $p$ -value for Hansen’s $J$ $\chi^2$ statistic |
| <code>r(J_df)</code>      | $\chi^2$ degrees of freedom                    |

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

Methods and formulas are presented under the following headings:

*Notation*  
*estat endogenous*  
*estat firststage*  
*estat overid*

## Notation

Recall from [R] [ivregress](#) that the model is

$$\mathbf{y} = \mathbf{Y}\beta_1 + \mathbf{X}_1\beta_2 + \mathbf{u} = \mathbf{X}\beta + \mathbf{u}$$

$$\mathbf{Y} = \mathbf{X}_1\boldsymbol{\Pi}_1 + \mathbf{X}_2\boldsymbol{\Pi}_2 + \mathbf{V} = \mathbf{Z}\boldsymbol{\Pi} + \mathbf{V}$$

where  $\mathbf{y}$  is an  $N \times 1$  vector of the left-hand-side variable,  $N$  is the sample size,  $\mathbf{Y}$  is an  $N \times p$  matrix of  $p$  endogenous regressors,  $\mathbf{X}_1$  is an  $N \times k_1$  matrix of  $k_1$  included exogenous regressors,  $\mathbf{X}_2$  is an  $N \times k_2$  matrix of  $k_2$  excluded exogenous variables,  $\mathbf{X} = [\mathbf{Y} \ \mathbf{X}_1]$ ,  $\mathbf{Z} = [\mathbf{X}_1 \ \mathbf{X}_2]$ ,  $\mathbf{u}$  is an  $N \times 1$  vector of errors,  $\mathbf{V}$  is an  $N \times p$  matrix of errors,  $\beta = [\beta_1 \ \beta_2]$  is a  $k = (p + k_1) \times 1$  vector of parameters, and  $\boldsymbol{\Pi}$  is a  $(k_1 + k_2) \times p$  vector of parameters. If a constant term is included in the model, then one column of  $\mathbf{X}_1$  contains all ones.

## estat endogenous

Partition  $\mathbf{Y}$  as  $\mathbf{Y} = [\mathbf{Y}_1 \ \mathbf{Y}_2]$ , where  $\mathbf{Y}_1$  represents the  $p_1$  endogenous regressors whose endogeneity is being tested and  $\mathbf{Y}_2$  represents the  $p_2$  endogenous regressors whose endogeneity is not being tested. If the endogeneity of all endogenous regressors is being tested,  $\mathbf{Y} = \mathbf{Y}_1$  and  $p_2 = 0$ . After GMM estimation, `estat endogenous` refits the model treating  $\mathbf{Y}_1$  as exogenous using the same type of weight matrix as requested at estimation time with the `wmatrix()` option; denote the Sargan statistic from this model by  $J_e$  and the estimated weight matrix by  $\mathbf{W}_e$ . Let  $\mathbf{S}_e = \mathbf{W}_e^{-1}$ . `estat endogenous` removes from  $\mathbf{S}_e$  the rows and columns corresponding to the variables represented by  $\mathbf{Y}_1$ ; denote the inverse of the resulting matrix by  $\mathbf{W}'_e$ . Next `estat endogenous` fits the model treating both  $\mathbf{Y}_1$  and  $\mathbf{Y}_2$  as endogenous, using the weight matrix  $\mathbf{W}'_e$ ; denote the Sargan statistic from this model by  $J_c$ . Then  $C = (J_e - J_c) \sim \chi^2(p_1)$ . If one simply used the  $J$  statistic from the original model fit by `ivregress` in place of  $J_c$ , then in finite samples  $J_e - J$  might be negative. The procedure used by `estat endogenous` is guaranteed to yield  $C \geq 0$ ; see Hayashi (2000, 220).

Let  $\hat{\mathbf{u}}_e$  denote the residuals from the model treating both  $\mathbf{Y}_1$  and  $\mathbf{Y}_2$  as endogenous, and let  $\hat{\mathbf{u}}_c$  denote the residuals from the model treating only  $\mathbf{Y}_2$  as endogenous. Then Durbin's (1954) statistic is

$$D = \frac{\hat{\mathbf{u}}'_e \mathbf{P}_{ZY_1} \hat{\mathbf{u}}_e - \hat{\mathbf{u}}'_c \mathbf{P}_Z \hat{\mathbf{u}}_c}{\hat{\mathbf{u}}'_e \hat{\mathbf{u}}_e / N}$$

where  $\mathbf{P}_Z = \mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'$  and  $\mathbf{P}_{ZY_1} = [\mathbf{Z} \ \mathbf{Y}_1][(\mathbf{Z} \ \mathbf{Y}_1)'(\mathbf{Z} \ \mathbf{Y}_1)]^{-1}[\mathbf{Z} \ \mathbf{Y}_1]'$ .  $D \sim \chi^2(p_1)$ . The Wu–Hausman (Wu 1974; Hausman 1978) statistic is

$$WH = \frac{(\hat{\mathbf{u}}'_e \mathbf{P}_{ZY_1} \hat{\mathbf{u}}_e - \hat{\mathbf{u}}'_c \mathbf{P}_Z \hat{\mathbf{u}}_c) / p_1}{\{\hat{\mathbf{u}}'_e \hat{\mathbf{u}}_e - (\hat{\mathbf{u}}'_c \mathbf{P}_{ZY_1} \hat{\mathbf{u}}_e - \hat{\mathbf{u}}'_c \mathbf{P}_Z \hat{\mathbf{u}}_c)\} / (N - k_1 - p - p_1)}$$

$WH \sim F(p_1, N - k_1 - p - p_1)$ . Baum, Schaffer, and Stillman (2003, 2007) discuss these tests in more detail.

Next we describe Wooldridge's (1995) score test. The nonrobust version of Wooldridge's test is identical to Durbin's test. Suppose a robust covariance matrix was used at estimation time. Let  $\hat{\mathbf{e}}$  denote the sample residuals obtained by fitting the model via OLS, treating  $\mathbf{Y}$  as exogenous. We then regress each variable represented in  $\mathbf{Y}$  on  $\mathbf{Z}$ ; call the residuals for the  $j$ th regression  $\hat{\mathbf{r}}_j$ ,  $j = 1, \dots, p$ . Define  $\hat{k}_{ij} = \hat{e}_i \hat{r}_{ij}$ ,  $i = 1, \dots, N$ . We then run the regression

$$\mathbf{1} = \theta_1 \hat{\mathbf{k}}_1 + \dots + \theta_p \hat{\mathbf{k}}_p + \epsilon$$

where  $\mathbf{1}$  is an  $N \times 1$  vector of ones and  $\epsilon$  is a regression error term.  $N - \text{RSS} \sim \chi^2(p)$ , where RSS is the residual sum of squares from the regression just described. If instead an HAC VCE was used at estimation time, then before running the final regression we prewhiten the  $\hat{\mathbf{k}}_j$  series by using a VAR( $q$ ) model, where  $q$  is the number of lags specified with the `lags()` option.

The regression-based test proceeds as follows. Following Hausman (1978, 1259), we regress  $\mathbf{Y}$  on  $\mathbf{Z}$  and obtain the residuals  $\hat{\mathbf{V}}$ . Next we fit the augmented regression

$$\mathbf{y} = \mathbf{Y}\beta_1 + \mathbf{X}_1\beta_2 + \hat{\mathbf{V}}\gamma + \epsilon$$

by OLS regression, where  $\epsilon$  is a regression error term. A test of the exogeneity of  $\mathbf{Y}$  is equivalent to a test of  $\gamma = \mathbf{0}$ . As Cameron and Trivedi (2005, 276) suggest, this test can be made robust to heteroskedasticity, autocorrelation, or clustering by using the appropriate robust VCE when testing  $\gamma = \mathbf{0}$ . When a nonrobust VCE is used, this test is equivalent to the Wu–Hausman test described earlier. One cannot simply fit this augmented regression via 2SLS to test the endogeneity of a subset of the endogenous regressors; Davidson and MacKinnon (1993, 229–231) discuss a test of  $\gamma = \mathbf{0}$  for the homoskedastic version of the augmented regression fit by 2SLS, but an appropriate robust test is not apparent.



## estat firststage

When the structural equation includes one endogenous regressor, `estat firststage` fits the regression

$$\mathbf{Y} = \mathbf{X}_1\boldsymbol{\pi}_1 + \mathbf{X}_2\boldsymbol{\pi}_2 + \mathbf{v}$$

via OLS. The  $R^2$  and adjusted  $R^2$  from that regression are reported in the output, as well as the  $F$  statistic from the Wald test of  $H_0: \boldsymbol{\pi}_2 = \mathbf{0}$ . To obtain the partial  $R^2$  statistic, `estat firststage` fits the regression

$$\mathbf{M}_{\mathbf{X}_1}\mathbf{y} = \mathbf{M}_{\mathbf{X}_1}\mathbf{X}_2\boldsymbol{\xi} + \boldsymbol{\epsilon}$$

by OLS, where  $\boldsymbol{\epsilon}$  is a regression error term,  $\boldsymbol{\xi}$  is a  $k_2 \times 1$  parameter vector, and  $\mathbf{M}_{\mathbf{X}_1} = \mathbf{I} - \mathbf{X}_1(\mathbf{X}_1'\mathbf{X}_1)^{-1}\mathbf{X}_1'$ ; i.e., the partial  $R^2$  is the  $R^2$  between  $\mathbf{y}$  and  $\mathbf{X}_2$  after eliminating the effects of  $\mathbf{X}_1$ . If the model contains multiple endogenous regressors and the `all` option is specified, these statistics are calculated for each endogenous regressor in turn.

To calculate Shea's partial  $R^2$ , let  $\mathbf{y}_1$  denote the endogenous regressor whose statistic is being calculated and  $\mathbf{Y}_0$  denote the other endogenous regressors. Define  $\tilde{\mathbf{y}}_1$  as the residuals obtained from regressing  $\mathbf{y}_1$  on  $\mathbf{Y}_0$  and  $\mathbf{X}_1$ . Let  $\hat{\mathbf{y}}_1$  denote the fitted values obtained from regressing  $\mathbf{y}_1$  on  $\mathbf{X}_1$  and  $\mathbf{X}_2$ ; i.e.,  $\hat{\mathbf{y}}_1$  are the fitted values from the first-stage regression for  $\mathbf{y}_1$ , and define the columns of  $\hat{\mathbf{Y}}_0$  analogously. Finally, let  $\tilde{\tilde{\mathbf{y}}}_1$  denote the residuals from regressing  $\hat{\mathbf{y}}_1$  on  $\hat{\mathbf{Y}}_0$  and  $\mathbf{X}_1$ . Shea's partial  $R^2$  is the simple  $R^2$  from the regression of  $\tilde{\tilde{\mathbf{y}}}_1$  on  $\tilde{\mathbf{y}}_1$ ; denote this as  $R_S^2$ . Shea's adjusted partial  $R^2$  is equal to  $1 - (1 - R_S^2)(N - 1)/(N - k_Z + 1)$  if a constant term is included and  $1 - (1 - R_S^2)(N - 1)/(N - k_Z)$  if there is no constant term included in the model, where  $k_Z = k_1 + k_2$ . For one endogenous regressor, one instrument, no exogenous regressors, and a constant term,  $R_S^2$  equals the adjusted  $R_S^2$ .

The Stock and Yogo minimum eigenvalue statistic, first proposed by [Cragg and Donald \(1993\)](#) as a test for underidentification, is the minimum eigenvalue of the matrix

$$\mathbf{G} = \frac{1}{k_Z} \hat{\boldsymbol{\Sigma}}_{\mathbf{V}\mathbf{V}}^{-1/2} \mathbf{Y}'\mathbf{M}'_{\mathbf{X}_1}\mathbf{X}_2(\mathbf{X}_2'\mathbf{M}_{\mathbf{X}_1}\mathbf{X}_2)^{-1}\mathbf{X}_2'\mathbf{M}_{\mathbf{X}_1}\mathbf{Y}\hat{\boldsymbol{\Sigma}}_{\mathbf{V}\mathbf{V}}^{-1/2}$$

where

$$\hat{\boldsymbol{\Sigma}}_{\mathbf{V}\mathbf{V}} = \frac{1}{N - k_Z} \mathbf{Y}'\mathbf{M}_Z\mathbf{Y}$$

$\mathbf{M}_Z = \mathbf{I} - \mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'$ , and  $\mathbf{Z} = [\mathbf{X}_1 \ \mathbf{X}_2]$ . Critical values are obtained from the tables in [Stock and Yogo \(2005\)](#).

## estat overid

The [Sargan \(1958\)](#) and [Basmann \(1960\)](#)  $\chi^2$  statistics are calculated by running the auxiliary regression

$$\hat{\mathbf{u}} = \mathbf{Z}\boldsymbol{\delta} + \mathbf{e}$$

where  $\hat{\mathbf{u}}$  are the sample residuals from the model and  $\mathbf{e}$  is an error term. Then Sargan's statistic is

$$S = N \left( 1 - \frac{\hat{\mathbf{e}}'\hat{\mathbf{e}}}{\hat{\mathbf{u}}'\hat{\mathbf{u}}} \right)$$

where  $\hat{\mathbf{e}}$  are the residuals from that auxiliary regression. Basmann's statistic is calculated as

$$B = S \frac{N - k_Z}{N - S}$$

Both  $S$  and  $B$  are distributed  $\chi^2(m)$ , where  $m$ , the number of overidentifying restrictions, is equal to  $k_Z - k$ , where  $k$  is the number of endogenous regressors.

Wooldridge's (1995) score test of overidentifying restrictions is identical to Sargan's (1958) statistic under the assumption of i.i.d. and therefore is not recomputed unless a robust VCE was used at estimation time. If a heteroskedasticity-robust VCE was used, Wooldridge's test proceeds as follows. Let  $\hat{\mathbf{Y}}$  denote the  $N \times k$  matrix of fitted values obtained by regressing the endogenous regressors on  $\mathbf{X}_1$  and  $\mathbf{X}_2$ . Let  $\mathbf{Q}$  denote an  $N \times m$  matrix of excluded exogenous variables; the test statistic to be calculated is invariant to whichever  $m$  of the  $k_2$  excluded exogenous variables is chosen. Define the  $i$ th element of  $\hat{\mathbf{k}}_j$ ,  $i = 1, \dots, N$ ,  $j = 1, \dots, m$ , as

$$k_{ij} = \hat{q}_{ij}u_i$$

where  $\hat{q}_{ij}$  is the  $i$ th element of  $\hat{\mathbf{q}}_j$ , the fitted values from regressing the  $j$ th column of  $\mathbf{Q}$  on  $\hat{\mathbf{Y}}$  and  $\mathbf{X}_1$ . Finally, fit the regression

$$\mathbf{1} = \theta_1 \hat{\mathbf{k}}_1 + \dots + \theta_m \hat{\mathbf{k}}_m + \epsilon$$

where  $\mathbf{1}$  is an  $N \times 1$  vector of ones and  $\epsilon$  is a regression error term, and calculate the residual sum of squares, RSS. Then the test statistic is  $W = N - \text{RSS}$ .  $W \sim \chi^2(m)$ . If an HAC VCE was used at estimation, then the  $\hat{\mathbf{k}}_j$  are prewhitened using a VAR( $p$ ) model, where  $p$  is specified using the `lags()` option.

The Anderson–Rubin (1950), AR, test of overidentifying restrictions for use after the LIML estimator is calculated as  $\text{AR} = N(\kappa - 1)$ , where  $\kappa$  is the minimal eigenvalue of a certain matrix defined in [Methods and formulas](#) of [\[R\] ivregress](#).  $\text{AR} \sim \chi^2(m)$ . (Some texts define this statistic as  $N \ln(\kappa)$  because  $\ln(x) \approx (x - 1)$  for  $x$  near one.) Basman's  $F$  statistic for use after the LIML estimator is calculated as  $B_F = (\kappa - 1)(N - k_Z)/m$ .  $B_F \sim F(m, N - k_Z)$ .

Hansen's  $J$  statistic is simply the value of the GMM objective function defined in (5) of [\[R\] ivregress](#), evaluated at the estimated parameter values. Under the null hypothesis that the overidentifying restrictions are valid,  $J \sim \chi^2(m)$ .

## References

- Anderson, T. W., and H. Rubin. 1950. The asymptotic properties of estimates of the parameters of a single equation in a complete system of stochastic equations. *Annals of Mathematical Statistics* 21: 570–582.
- Angrist, J. D., and J.-S. Pischke. 2009. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton, NJ: Princeton University Press.
- Basman, R. L. 1960. On finite sample distributions of generalized classical linear identifiability test statistics. *Journal of the American Statistical Association* 55: 650–659.
- Baum, C. F., M. E. Schaffer, and S. Stillman. 2003. Instrumental variables and GMM: Estimation and testing. *Stata Journal* 3: 1–31.
- . 2007. Enhanced routines for instrumental variables/generalized method of moments estimation and testing. *Stata Journal* 7: 465–506.
- Bound, J., D. A. Jaeger, and R. M. Baker. 1995. Problems with instrumental variables estimation when the correlation between the instruments and the endogenous explanatory variable is weak. *Journal of the American Statistical Association* 90: 443–450.
- Cameron, A. C., and P. K. Trivedi. 2005. *Microeconometrics: Methods and Applications*. New York: Cambridge University Press.
- Cragg, J. G., and S. G. Donald. 1993. Testing identifiability and specification in instrumental variable models. *Econometric Theory* 9: 222–240.
- Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.

- Durbin, J. 1954. Errors in variables. *Review of the International Statistical Institute* 22: 23–32.
- Hahn, J., and J. A. Hausman. 2003. Weak instruments: Diagnosis and cures in empirical econometrics. *American Economic Review Papers and Proceedings* 93: 118–125.
- Hall, A. R., G. D. Rudebusch, and D. W. Wilcox. 1996. Judging instrument relevance in instrumental variables estimation. *International Economic Review* 37: 283–298.
- Hansen, L. P. 1982. Large sample properties of generalized method of moments estimators. *Econometrica* 50: 1029–1054.
- Hausman, J. A. 1978. Specification tests in econometrics. *Econometrica* 46: 1251–1271.
- Hayashi, F. 2000. *Econometrics*. Princeton, NJ: Princeton University Press.
- Judge, G. G., W. E. Griffiths, R. C. Hill, H. Lütkepohl, and T.-C. Lee. 1985. *The Theory and Practice of Econometrics*. 2nd ed. New York: Wiley.
- Nelson, C. R., and R. Startz. 1990. The distribution of the instrumental variable estimator and its t ratio when the instrument is a poor one. *Journal of Business* 63: S125–S140.
- Poi, B. P. 2006. Jackknife instrumental variables estimation in Stata. *Stata Journal* 6: 364–376.
- Sargan, J. D. 1958. The estimation of economic relationships using instrumental variables. *Econometrica* 26: 393–415.
- Shea, J. 1997. Instrument relevance in multivariate linear models: A simple measure. *Review of Economics and Statistics* 79: 348–352.
- Staiger, D., and J. H. Stock. 1997. Instrumental variables regression with weak instruments. *Econometrica* 65: 557–586.
- Stock, J. H., J. H. Wright, and M. Yogo. 2002. A survey of weak instruments and weak identification in generalized method of moments. *Journal of Business and Economic Statistics* 20: 518–529.
- Stock, J. H., and M. Yogo. 2005. Testing for weak instruments in linear IV regression. In *Identification and Inference for Econometric Models: Essays in Honor of Thomas Rothenberg*, ed. D. W. K. Andrews and J. H. Stock, 80–108. New York: Cambridge University Press.
- Wooldridge, J. M. 1995. Score diagnostics for linear models estimated by two stage least squares. In *Advances in Econometrics and Quantitative Economics: Essays in Honor of Professor C. R. Rao*, ed. G. S. Maddala, P. C. B. Phillips, and T. N. Srinivasan, 66–87. Oxford: Blackwell.
- Wu, D.-M. 1974. Alternative tests of independence between stochastic regressors and disturbances: Finite sample results. *Econometrica* 42: 529–546.

## Also see

- [R] **ivregress** — Single-equation instrumental-variables regression
- [U] **20 Estimation and postestimation commands**

Syntax

Maximum likelihood estimator

```
ivtobit depvar [varlist1] (varlist2 = varlistiv) [if] [in] [weight],
 ll[(#)] ul[(#)] [mle_options]
```

Two-step estimator

```
ivtobit depvar [varlist1] (varlist2 = varlistiv) [if] [in] [weight], twostep
 ll[(#)] ul[(#)] [tse_options]
```

| mle_options                      | description                                                                                             |
|----------------------------------|---------------------------------------------------------------------------------------------------------|
| Model                            |                                                                                                         |
| * ll[(#)]                        | lower limit for left censoring                                                                          |
| * ul[(#)]                        | upper limit for right censoring                                                                         |
| <u>mle</u>                       | use conditional maximum-likelihood estimator; the default                                               |
| <u>constraints</u> (constraints) | apply specified linear constraints                                                                      |
| SE/Robust                        |                                                                                                         |
| vce(vctype)                      | vctype may be oim, <u>robust</u> , <u>cluster</u> clustvar, opg, <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting                        |                                                                                                         |
| <u>level</u> (#)                 | set confidence level; default is level(95)                                                              |
| <u>first</u>                     | report first-stage regression                                                                           |
| <u>nocnsreport</u>               | do not display constraints                                                                              |
| <u>display_options</u>           | control spacing and display of omitted variables and base and empty cells                               |
| Maximization                     |                                                                                                         |
| <u>maximize_options</u>          | control the maximization process                                                                        |
| <sup>†</sup> <u>coeflegend</u>   | display coefficients' legend instead of coefficient table                                               |

\*You must specify at least one of ll[(#)] and ul[(#)].

<sup>†</sup>coeflegend does not appear in the dialog box.

| <i>tse_options</i>             | description                                                                                     |
|--------------------------------|-------------------------------------------------------------------------------------------------|
| Model                          |                                                                                                 |
| * <b>twostep</b>               | use Newey’s two-step estimator; the default is <code>mle</code>                                 |
| * <b>ll</b> [(#)]              | lower limit for left censoring                                                                  |
| * <b>ul</b> [(#)]              | upper limit for right censoring                                                                 |
| SE                             |                                                                                                 |
| <b>vce</b> ( <i>vcetype</i> )  | <i>vcetype</i> may be <code>twostep</code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                      |                                                                                                 |
| <b>level</b> (#)               | set confidence level; default is <code>level(95)</code>                                         |
| <b>first</b>                   | report first-stage regression                                                                   |
| <i>display_options</i>         | control spacing and display of omitted variables and base and empty cells                       |
| <sup>†</sup> <b>coeflegend</b> | display coefficients’ legend instead of coefficient table                                       |

\***twostep** is required. You must specify at least one of `ll[(#)]` and `ul[(#)]`.

<sup>†</sup>**coeflegend** does not appear in the dialog box.

`varlist1` and `varlist1v` may contain factor variables; see [U] 11.4.3 Factor variables.

`depvar`, `varlist1`, `varlist2`, and `varlist1v` may contain time-series operators; see [U] 11.4.4 Time-series varlists.

`bootstrap`, `by`, `jackknife`, `rolling`, `statsby`, and `svy` are allowed; see [U] 11.1.10 Prefix commands.

Weights are not allowed with the `bootstrap` prefix.

`vce()`, `first`, `twostep`, and `weights` are not allowed with the `svy` prefix.

`fweights`, `iweights`, and `pweights` are allowed with the maximum likelihood estimator. `fweights` are allowed with Newey’s two-step estimator. See [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Endogenous covariates > Tobit model with endogenous covariates

Description

`ivtobit` fits tobit models where one or more of the regressors is endogenously determined. By default, `ivtobit` uses maximum likelihood estimation. Alternatively, Newey’s (1987) minimum chi-squared estimator can be invoked with the `twostep` option. Both estimators assume that the endogenous regressors are continuous and so are not appropriate for use with discrete endogenous regressors. See [R] `ivprobit` for probit estimation with endogenous regressors and [R] `tobit` for tobit estimation when the model contains no endogenous regressors.

Options for ML estimator

Model

`ll(#)` and `ul(#)` indicate the lower and upper limits for censoring, respectively. You may specify one or both. Observations with `depvar ≤ ll()` are left-censored; observations with `depvar ≥ ul()` are right-censored; and remaining observations are not censored. You do not have to specify the censoring values at all. It is enough to type `ll`, `ul`, or both. When you do not specify a censoring value, `ivtobit` assumes that the lower limit is the minimum observed in the data (if `ll` is specified) and that the upper limit is the maximum (if `ul` is specified).

`mle` requests that the conditional maximum-likelihood estimator be used. This is the default.

`constraints(constraints)`; see [R] [estimation options](#).

#### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

#### Reporting

`level(#)`; see [R] [estimation options](#).

`first` requests that the parameters for the reduced-form equations showing the relationships between the endogenous variables and instruments be displayed. For the two-step estimator, `first` shows the first-stage regressions. For the maximum likelihood estimator, these parameters are estimated jointly with the parameters of the tobit equation. The default is not to show these parameter estimates.

`nocnsreport`; see [R] [estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

#### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [R] [maximize](#). This model's likelihood function can be difficult to maximize, especially with multiple endogenous variables. The `difficult` and `technique(bfgs)` options may be helpful in achieving convergence.

Setting the optimization type to `technique(bhhh)` resets the default `vcetype` to `vce(opg)`.

The following option is available with `ivtobit` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Options for two-step estimator

#### Model

`twostep` is required and requests that Newey's (1987) efficient two-step estimator be used to obtain the coefficient estimates.

`ll(#)` and `ul(#)` indicate the lower and upper limits for censoring, respectively. You may specify one or both. Observations with `depvar ≤ ll()` are left-censored; observations with `depvar ≥ ul()` are right-censored; and remaining observations are not censored. You do not have to specify the censoring values at all. It is enough to type `ll`, `ul`, or both. When you do not specify a censoring value, `ivtobit` assumes that the lower limit is the minimum observed in the data (if `ll` is specified) and that the upper limit is the maximum (if `ul` is specified).

#### SE

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

## Reporting

`level(#)`; see [R] [estimation options](#).

`first` requests that the parameters for the reduced-form equations showing the relationships between the endogenous variables and instruments be displayed. For the two-step estimator, `first` shows the first-stage regressions. For the maximum likelihood estimator, these parameters are estimated jointly with the parameters of the tobit equation. The default is not to show these parameter estimates.

*display\_options*: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

The following option is available with `ivtobit` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

`ivtobit` fits models with censored dependent variables and endogenous regressors. You can use it to fit a tobit model when you suspect that one or more of the regressors is correlated with the error term. `ivtobit` is to tobit what `ivregress` is to linear regression analysis; see [R] [ivregress](#) for more information.

Formally, the model is

$$\begin{aligned}y_{1i}^* &= \mathbf{y}_{2i}\boldsymbol{\beta} + \mathbf{x}_{1i}\boldsymbol{\gamma} + u_i \\ \mathbf{y}_{2i} &= \mathbf{x}_{1i}\boldsymbol{\Pi}_1 + \mathbf{x}_{2i}\boldsymbol{\Pi}_2 + \mathbf{v}_i\end{aligned}$$

where  $i = 1, \dots, N$ ;  $\mathbf{y}_{2i}$  is a  $1 \times p$  vector of endogenous variables;  $\mathbf{x}_{1i}$  is a  $1 \times k_1$  vector of exogenous variables;  $\mathbf{x}_{2i}$  is a  $1 \times k_2$  vector of additional instruments; and the equation for  $\mathbf{y}_{2i}$  is written in reduced form. By assumption  $(u_i, \mathbf{v}_i) \sim N(\mathbf{0})$ .  $\boldsymbol{\beta}$  and  $\boldsymbol{\gamma}$  are vectors of structural parameters, and  $\boldsymbol{\Pi}_1$  and  $\boldsymbol{\Pi}_2$  are matrices of reduced-form parameters. We do not observe  $y_{1i}^*$ ; instead, we observe

$$y_{1i} = \begin{cases} a & y_{1i}^* < a \\ y_{1i}^* & a \leq y_{1i}^* \leq b \\ b & y_{1i}^* > b \end{cases}$$

The order condition for identification of the structural parameters is that  $k_2 \geq p$ . Presumably,  $\boldsymbol{\Sigma}$  is not block diagonal between  $u_i$  and  $\mathbf{v}_i$ ; otherwise,  $\mathbf{y}_{2i}$  would not be endogenous.

## □ Technical note

This model is derived under the assumption that  $(u_i, \mathbf{v}_i)$  is independent and identically distributed multivariate normal for all  $i$ . The `vce(cluster clustvar)` option can be used to control for a lack of independence. As with the standard tobit model without endogeneity, if  $u_i$  is heteroskedastic, point estimates will be inconsistent.

□

➤ Example 1

Using the same dataset as in [R] [ivprobit](#), we now want to estimate women’s incomes. In our hypothetical world, all women who choose not to work receive \$10,000 in welfare and child-support payments. Therefore, we never observe incomes under \$10,000: a woman offered a job with an annual wage less than that would not accept and instead would collect the welfare payment. We model income as a function of the number of years of schooling completed, the number of children at home, and other household income. We again believe that `other_inc` is endogenous, so we use `male_educ` as an instrument.

```
. use http://www.stata-press.com/data/r11/laborsup
. ivtobit fem_inc fem_educ kids (other_inc = male_educ), ll
Fitting exogenous tobit model
Fitting full model
Iteration 0: log likelihood = -3228.4224
Iteration 1: log likelihood = -3226.2882
Iteration 2: log likelihood = -3226.085
Iteration 3: log likelihood = -3226.0845
Iteration 4: log likelihood = -3226.0845

Tobit model with endogenous regressors Number of obs = 500
 Wald chi2(3) = 117.42
Log likelihood = -3226.0845 Prob > chi2 = 0.0000
```

|           | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-----------|-----------|-----------|-------|-------|----------------------|-----------|
| other_inc | -.9045399 | .1329762  | -6.80 | 0.000 | -1.165168            | -.6439114 |
| fem_educ  | 3.272391  | .3968708  | 8.25  | 0.000 | 2.494538             | 4.050243  |
| kids      | -3.312357 | .7218628  | -4.59 | 0.000 | -4.727182            | -1.897532 |
| _cons     | 19.24735  | 7.372391  | 2.61  | 0.009 | 4.797725             | 33.69697  |
| /alpha    | .2907654  | .1379965  | 2.11  | 0.035 | .0202972             | .5612336  |
| /lns      | 2.874031  | .0506672  | 56.72 | 0.000 | 2.774725             | 2.973337  |
| /lnv      | 2.813383  | .0316228  | 88.97 | 0.000 | 2.751404             | 2.875363  |
| s         | 17.70826  | .897228   |       |       | 16.03422             | 19.55707  |
| v         | 16.66621  | .5270318  |       |       | 15.66461             | 17.73186  |

```
Instrumented: other_inc
Instruments: fem_educ kids male_educ

Wald test of exogeneity (/alpha = 0): chi2(1) = 4.44 Prob > chi2 = 0.0351
Obs. summary: 272 left-censored observations at fem_inc<=10
 228 uncensored observations
 0 right-censored observations
```

Because we did not specify `mle` or `twostep`, `ivtobit` used the maximum likelihood estimator by default. `ivtobit` fits a tobit model, ignoring endogeneity, to get starting values for the full model. The header of the output contains the maximized log likelihood, the number of observations, and a Wald statistic and *p*-value for the test of the hypothesis that all the slope coefficients are jointly zero. At the end of the output, we see a count of the censored and uncensored observations.

Near the bottom of the output is a Wald test of the exogeneity of the instrumented variables. If the test statistic is not significant, there is not sufficient information in the sample to reject the null hypothesis of no endogeneity. Then the point estimates from `ivtobit` are consistent, although those from `tobit` are likely to have smaller standard errors.



Various two-step estimators have also been proposed for the endogenous tobit model, and Newey's (1987) minimum chi-squared estimator is available with the `twostep` option.

## ► Example 2

Refitting our labor-supply model with the two-step estimator yields

```
. ivtobit fem_inc fem_educ kids (other_inc = male_educ), ll twostep
Two-step tobit with endogenous regressors Number of obs = 500
 Wald chi2(3) = 117.38
 Prob > chi2 = 0.0000
```

|           | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-----------|-----------|-----------|-------|-------|----------------------|-----------|
| other_inc | -.9045397 | .1330015  | -6.80 | 0.000 | -1.165218            | -.6438616 |
| fem_educ  | 3.27239   | .3969399  | 8.24  | 0.000 | 2.494402             | 4.050378  |
| kids      | -3.312356 | .7220066  | -4.59 | 0.000 | -4.727463            | -1.897249 |
| _cons     | 19.24735  | 7.37392   | 2.61  | 0.009 | 4.794728             | 33.69997  |

```
Instrumented: other_inc
Instruments: fem_educ kids male_educ
```

---

```
Wald test of exogeneity: chi2(1) = 4.64 Prob > chi2 = 0.0312
Obs. summary: 272 left-censored observations at fem_inc<=10
 228 uncensored observations
 0 right-censored observations
```

All the coefficients have the same signs as their counterparts in the maximum likelihood model. The Wald test at the bottom of the output confirms our earlier finding of endogeneity.

◀

## □ Technical note

In the tobit model with endogenous regressors, we assume that  $(u_i, v_i)$  is multivariate normal with covariance matrix

$$\text{Var}(u_i, v_i) = \Sigma = \begin{bmatrix} \sigma_u^2 & \Sigma'_{21} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$$

Using the properties of the multivariate normal distribution,  $\text{Var}(u_i | v_i) \equiv \sigma_{u|v}^2 = \sigma_u^2 - \Sigma'_{21} \Sigma_{22}^{-1} \Sigma_{21}$ . Calculating the marginal effects on the conditional expected values of the observed and latent dependent variables and on the probability of censoring requires an estimate of  $\sigma_{u|v}^2$ . The two-step estimator identifies only  $\sigma_{u|v}^2$ , not  $\sigma_u^2$ , so only the linear prediction and its standard error are available after you have used the `twostep` option. However, unlike the two-step probit estimator described in [R] [ivprobit](#), the two-step tobit estimator does identify  $\beta$  and  $\gamma$ . See [Wooldridge \(2002, 532\)](#) for more information.

□

(Continued on next page)

## Saved results

`ivtobit`, `mle` saves the following in `e()`:

### Scalars

|                            |                                                                            |
|----------------------------|----------------------------------------------------------------------------|
| <code>e(N)</code>          | number of observations                                                     |
| <code>e(N_unc)</code>      | number of uncensored observations                                          |
| <code>e(N_ltc)</code>      | number of left-censored observations                                       |
| <code>e(N_rc)</code>       | number of right-censored observations                                      |
| <code>e(llopt)</code>      | contents of <code>ll()</code>                                              |
| <code>e(ulopt)</code>      | contents of <code>ul()</code>                                              |
| <code>e(k)</code>          | number of parameters                                                       |
| <code>e(k_eq)</code>       | number of equations                                                        |
| <code>e(k_eq_model)</code> | number of equations in model Wald test                                     |
| <code>e(k_aux)</code>      | number of auxiliary parameters                                             |
| <code>e(k_dv)</code>       | number of dependent variables                                              |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints                             |
| <code>e(df_m)</code>       | model degrees of freedom                                                   |
| <code>e(ll)</code>         | log likelihood                                                             |
| <code>e(N_clust)</code>    | number of clusters                                                         |
| <code>e(k_eq_skip)</code>  | identifies which equations should not be reported in the coefficient table |
| <code>e(endog_ct)</code>   | number of endogenous regressors                                            |
| <code>e(p)</code>          | model Wald $p$ -value                                                      |
| <code>e(p_exog)</code>     | exogeneity test Wald $p$ -value                                            |
| <code>e(chi2)</code>       | model Wald $\chi^2$                                                        |
| <code>e(chi2_exog)</code>  | Wald $\chi^2$ test of exogeneity                                           |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                                                  |
| <code>e(ic)</code>         | number of iterations                                                       |
| <code>e(rc)</code>         | return code                                                                |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                                                |

## Macros

|                    |                                                                          |
|--------------------|--------------------------------------------------------------------------|
| e(cmd)             | ivtobit                                                                  |
| e(cmdline)         | command as typed                                                         |
| e(depvar)          | name of dependent variable                                               |
| e(instd)           | instrumented variables                                                   |
| e(insts)           | instruments                                                              |
| e(wtype)           | weight type                                                              |
| e(wexp)            | weight expression                                                        |
| e(title)           | title in estimation output                                               |
| e(clustvar)        | name of cluster variable                                                 |
| e(chi2type)        | Wald; type of model $\chi^2$ test                                        |
| e(vce)             | vcetype specified in vce()                                               |
| e(vcetype)         | title used to label Std. Err.                                            |
| e(diparm#)         | display transformed parameter #                                          |
| e(method)          | ml                                                                       |
| e(opt)             | type of optimization                                                     |
| e(which)           | max or min; whether optimizer is to perform maximization or minimization |
| e(ml_method)       | type of ml method                                                        |
| e(user)            | name of likelihood-evaluator program                                     |
| e(technique)       | maximization technique                                                   |
| e(singularHmethod) | m-marquardt or hybrid; method used when Hessian is singular              |
| e(crittype)        | optimization criterion                                                   |
| e(properties)      | b V                                                                      |
| e(predict)         | program used to implement predict                                        |
| e(footnote)        | program used to implement the footnote display                           |
| e(marginsok)       | predictions allowed by margins                                           |
| e(asbalanced)      | factor variables fvset as asbalanced                                     |
| e(asobserved)      | factor variables fvset as asobserved                                     |

## Matrices

|                 |                                              |
|-----------------|----------------------------------------------|
| e(b)            | coefficient vector                           |
| e(Cns)          | constraints matrix                           |
| e(ilog)         | iteration log (up to 20 iterations)          |
| e(gradient)     | gradient vector                              |
| e(Sigma)        | $\Sigma$                                     |
| e(V)            | variance–covariance matrix of the estimators |
| e(V_modelbased) | model-based variance                         |
| e(ml_h)         | derivative tolerance, (abs(b)+1e-3)*1e-3     |
| e(ml_scale)     | derivative scale factor                      |

## Functions

|           |                         |
|-----------|-------------------------|
| e(sample) | marks estimation sample |
|-----------|-------------------------|

(Continued on next page)

`ivtobit`, `twostep` saves the following in `e()`:

#### Scalars

|                           |                                                    |
|---------------------------|----------------------------------------------------|
| <code>e(N)</code>         | number of observations                             |
| <code>e(N_unc)</code>     | number of uncensored observations                  |
| <code>e(N_ltc)</code>     | number of left-censored observations               |
| <code>e(N_rc)</code>      | number of right-censored observations              |
| <code>e(llopt)</code>     | contents of <code>ll()</code>                      |
| <code>e(uopt)</code>      | contents of <code>ul()</code>                      |
| <code>e(df_m)</code>      | model degrees of freedom                           |
| <code>e(df_exog)</code>   | degrees of freedom for $\chi^2$ test of exogeneity |
| <code>e(p)</code>         | model Wald $p$ -value                              |
| <code>e(p_exog)</code>    | exogeneity test Wald $p$ -value                    |
| <code>e(chi2)</code>      | model Wald $\chi^2$                                |
| <code>e(chi2_exog)</code> | Wald $\chi^2$ test of exogeneity                   |
| <code>e(rank)</code>      | rank of <code>e(V)</code>                          |

#### Macros

|                            |                                                                |
|----------------------------|----------------------------------------------------------------|
| <code>e(cmd)</code>        | <code>ivtobit</code>                                           |
| <code>e(cmdline)</code>    | command as typed                                               |
| <code>e(depvar)</code>     | name of dependent variable                                     |
| <code>e(instd)</code>      | instrumented variables                                         |
| <code>e(insts)</code>      | instruments                                                    |
| <code>e(wtype)</code>      | weight type                                                    |
| <code>e(wexp)</code>       | weight expression                                              |
| <code>e(chi2type)</code>   | Wald; type of model $\chi^2$ test                              |
| <code>e(vce)</code>        | <i>vcetype</i> specified in <code>vce()</code>                 |
| <code>e(vcetype)</code>    | title used to label Std. Err.                                  |
| <code>e(method)</code>     | <code>twostep</code>                                           |
| <code>e(properties)</code> | <code>b V</code>                                               |
| <code>e(predict)</code>    | program used to implement <code>predict</code>                 |
| <code>e(footnote)</code>   | program used to implement the footnote display                 |
| <code>e(marginsok)</code>  | predictions allowed by <code>margins</code>                    |
| <code>e(asbalanced)</code> | factor variables <code>fvset</code> as <code>asbalanced</code> |
| <code>e(asobserved)</code> | factor variables <code>fvset</code> as <code>asobserved</code> |

#### Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

#### Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`ivtobit` is implemented as an ado-file.

The estimation procedure used by `ivtobit` is similar to that used by `ivprobit`. For compactness, we write the model as

$$y_{1i}^* = z_i \delta + u_i \quad (1a)$$

$$y_{2i} = x_i \Pi + v_i \quad (1b)$$

where  $\mathbf{z}_i = (\mathbf{y}_{2i}, \mathbf{x}_{1i})$ ,  $\mathbf{x}_i = (\mathbf{x}_{1i}, \mathbf{x}_{2i})$ ,  $\boldsymbol{\delta} = (\boldsymbol{\beta}', \boldsymbol{\gamma}')'$ , and  $\boldsymbol{\Pi} = (\boldsymbol{\Pi}'_1, \boldsymbol{\Pi}'_2)'$ . We do not observe  $y_{1i}^*$ ; instead, we observe

$$y_{1i} = \begin{cases} a & y_{1i}^* < a \\ y_{1i}^* & a \leq y_{1i}^* \leq b \\ b & y_{1i}^* > b \end{cases}$$

$(u_i, v_i)$  is distributed multivariate normal with mean zero and covariance matrix

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_u^2 & \boldsymbol{\Sigma}'_{21} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}$$

Using the properties of the multivariate normal distribution, we can write  $u_i = \mathbf{v}'_i \boldsymbol{\alpha} + \epsilon_i$ , where  $\boldsymbol{\alpha} = \boldsymbol{\Sigma}_{22}^{-1} \boldsymbol{\Sigma}_{21}$ ;  $\epsilon_i \sim N(0; \sigma_{u|v}^2)$ , where  $\sigma_{u|v}^2 = \sigma_u^2 - \boldsymbol{\Sigma}'_{21} \boldsymbol{\Sigma}_{22}^{-1} \boldsymbol{\Sigma}_{21}$ ; and  $\epsilon_i$  is independent of  $\mathbf{v}_i$ ,  $\mathbf{z}_i$ , and  $\mathbf{x}_i$ .

The likelihood function is straightforward to derive because we can write the joint density  $f(y_{1i}, \mathbf{y}_{2i} | \mathbf{x}_i)$  as  $f(y_{1i} | \mathbf{y}_{2i}, \mathbf{x}_i) f(\mathbf{y}_{2i} | \mathbf{x}_i)$ . With one endogenous regressor,

$$\ln f(y_{2i} | \mathbf{x}_i) = -\frac{1}{2} \left\{ \ln 2\pi + \ln \sigma_v^2 + \frac{(y_{2i} - \mathbf{x}_i \boldsymbol{\Pi})^2}{\sigma_v^2} \right\}$$

and

$$\ln f(y_{1i} | y_{2i}, \mathbf{x}_i) = \begin{cases} \ln \left\{ 1 - \Phi \left( \frac{m_i - a}{\sigma_{u|v}} \right) \right\} & y_{1i} = a \\ -\frac{1}{2} \left\{ \ln 2\pi + \ln \sigma_{u|v}^2 + \frac{(y_{1i} - m_i)^2}{\sigma_{u|v}^2} \right\} & a < y_{1i} < b \\ \ln \Phi \left( \frac{m_i - b}{\sigma_{u|v}} \right) & y_{1i} = b \end{cases}$$

where

$$m_i = \mathbf{z}_i \boldsymbol{\delta} + \alpha (y_{2i} - \mathbf{x}_i \boldsymbol{\Pi})$$

and  $\Phi(\cdot)$  is the normal distribution function so that the log likelihood for observation  $i$  is

$$\ln L_i = w_i \{ \ln f(y_{1i} | y_{2i}, \mathbf{x}_i) + \ln f(y_{2i} | \mathbf{x}_i) \}$$

where  $w_i$  is the weight for observation  $i$  or one if no weights were specified. Instead of estimating  $\sigma_{u|v}$  and  $\sigma_v$  directly, we estimate  $\ln \sigma_{u|v}$  and  $\ln \sigma_v$ .

For multiple endogenous regressors, we have

$$\ln f(\mathbf{y}_{2i} | \mathbf{x}_i) = -\frac{1}{2} \left( \ln 2\pi + \ln |\boldsymbol{\Sigma}_{22}| + \mathbf{v}'_i \boldsymbol{\Sigma}_{22}^{-1} \mathbf{v}_i \right)$$

and  $\ln f(y_{1i} | \mathbf{y}_{2i}, \mathbf{x}_i)$  is the same as before, except that now

$$m_i = \mathbf{z}_i \boldsymbol{\delta} + (\mathbf{y}_{2i} - \mathbf{x}_i \boldsymbol{\Pi}) \boldsymbol{\Sigma}_{22}^{-1} \boldsymbol{\Sigma}_{21}$$

Instead of maximizing the log-likelihood function with respect to  $\boldsymbol{\Sigma}$ , we maximize with respect to the Cholesky decomposition  $\mathbf{S}$  of  $\boldsymbol{\Sigma}$ ; that is, there exists a lower triangular matrix  $\mathbf{S}$  such that  $\mathbf{S} \mathbf{S}' = \boldsymbol{\Sigma}$ . This maximization ensures that  $\boldsymbol{\Sigma}$  is positive definite, as a covariance matrix must be. Let

$$S = \begin{bmatrix} s_{11} & 0 & 0 & \dots & 0 \\ s_{21} & s_{22} & 0 & \dots & 0 \\ s_{31} & s_{32} & s_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{p+1,1} & s_{p+1,2} & s_{p+1,3} & \dots & s_{p+1,p+1} \end{bmatrix}$$

With maximum likelihood estimation, this command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

The maximum likelihood version of `ivtobit` also supports estimation with survey data. For details on VCEs with survey data, see [\[SVY\] variance estimation](#).

The two-step estimates are obtained using Newey's (1987) minimum chi-squared estimator. The procedure is identical to the one described in [\[R\] ivprobit](#), except that `tobit` is used instead of `probit`.

## Acknowledgments

The two-step estimator is based on the `tobitiv` command written by Jonah Gelbach, College of Law, Florida State University, and the `ivtobit` command written by Joe Harkness, Institute of Policy Studies, Johns Hopkins University.

## References

- Miranda, A., and S. Rabe-Hesketh. 2006. [Maximum likelihood estimation of endogenous switching and sample selection models for binary, ordinal, and count variables](#). *Stata Journal* 6: 285–308.
- Newey, W. K. 1987. Efficient estimation of limited dependent variable models with endogenous explanatory variables. *Journal of Econometrics* 36: 231–250.
- Wooldridge, J. M. 2002. *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA: MIT Press.

## Also see

- [\[R\] ivtobit postestimation](#) — Postestimation tools for `ivtobit`
- [\[R\] gmm](#) — Generalized method of moments estimation
- [\[R\] ivprobit](#) — Probit model with continuous endogenous regressors
- [\[R\] ivregress](#) — Single-equation instrumental-variables regression
- [\[R\] regress](#) — Linear regression
- [\[R\] tobit](#) — Tobit regression
- [\[XT\] xtintreg](#) — Random-effects interval-data regression models
- [\[XT\] xttobit](#) — Random-effects tobit models
- [\[U\] 20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `ivtobit`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code> <sup>1</sup>  | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>hausman</code>             | Hausman's specification test                                                                        |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code> <sup>2</sup> | likelihood-ratio test; not available with two-step estimator                                        |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code> <sup>1</sup>  | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `estat ic` and `suest` are not appropriate after `ivtobit`, `twostep`.

<sup>2</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\]](#) `estat` for details about `estat (svy)`.

Syntax for predict

After *ML* or *twostep*

```
predict [type] newvar [if] [in] [, statistic]
```

After *ML*

```
predict [type] { stub* | newvarlist } [if] [in] , scores
```

| <i>statistic</i> | <i>description</i> |
|------------------|--------------------|
|------------------|--------------------|

Main

|                                 |                                                                                      |
|---------------------------------|--------------------------------------------------------------------------------------|
| <b>xb</b>                       | linear prediction; the default                                                       |
| <b>stdp</b>                     | standard error of the linear prediction                                              |
| <b>stdf</b>                     | standard error of the forecast; not available with two-step estimator                |
| <b>pr(<i>a</i>,<i>b</i>)</b>    | $\Pr(a < y_j < b)$ ; not available with two-step estimator                           |
| <b>e(<i>a</i>,<i>b</i>)</b>     | $E(y_j   a < y_j < b)$ ; not available with two-step estimator                       |
| <b>ystar(<i>a</i>,<i>b</i>)</b> | $E(y_j^*)$ , $y_j = \max\{a, \min(y_j, b)\}$ ; not available with two-step estimator |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

**stdf** is not allowed with **svy** estimation results.

where *a* and *b* may be numbers or variables; *a* missing ( $a \geq .$ ) means  $-\infty$ , and *b* missing ( $b \geq .$ ) means  $+\infty$ ; see [U] **12.2.1 Missing values**.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

**xb**, the default, calculates the linear prediction.

**stdp** calculates the standard error of the linear prediction. It can be thought of as the standard error of the predicted expected value or mean for the observation's covariate pattern. The standard error of the prediction is also referred to as the standard error of the fitted value.

**stdf** calculates the standard error of the forecast, which is the standard error of the point prediction for 1 observation. It is commonly referred to as the standard error of the future or forecast value. By construction, the standard errors produced by **stdf** are always larger than those produced by **stdp**; see [Methods and formulas](#) in [R] **regress**. **stdf** is not available with the two-step estimator.

**pr(*a*,*b*)** calculates  $\Pr(a < \mathbf{x}_j \mathbf{b} + u_j < b)$ , the probability that  $y_j | \mathbf{x}_j$  would be observed in the interval (*a*, *b*).

*a* and *b* may be specified as numbers or variable names; *lb* and *ub* are variable names;

**pr(20,30)** calculates  $\Pr(20 < \mathbf{x}_j \mathbf{b} + u_j < 30)$ ;

**pr(*lb*,*ub*)** calculates  $\Pr(lb < \mathbf{x}_j \mathbf{b} + u_j < ub)$ ; and

**pr(20,*ub*)** calculates  $\Pr(20 < \mathbf{x}_j \mathbf{b} + u_j < ub)$ .

*a* missing ( $a \geq .$ ) means  $-\infty$ ; **pr(.,30)** calculates  $\Pr(-\infty < \mathbf{x}_j \mathbf{b} + u_j < 30)$ ;

**pr(*lb*,30)** calculates  $\Pr(-\infty < \mathbf{x}_j \mathbf{b} + u_j < 30)$  in observations for which *lb*  $\geq .$  and calculates  $\Pr(lb < \mathbf{x}_j \mathbf{b} + u_j < 30)$  elsewhere.

*b* missing ( $b \geq .$ ) means  $+\infty$ ; **pr(20,.)** calculates  $\Pr(+\infty > \mathbf{x}_j \mathbf{b} + u_j > 20)$ ;

**pr(20,*ub*)** calculates  $\Pr(+\infty > \mathbf{x}_j \mathbf{b} + u_j > 20)$  in observations for which *ub*  $\geq .$  and calculates  $\Pr(20 < \mathbf{x}_j \mathbf{b} + u_j < ub)$  elsewhere.

**pr(*a*,*b*)** is not available with the two-step estimator.

**e(*a*,*b*)** calculates  $E(\mathbf{x}_j \mathbf{b} + u_j \mid a < \mathbf{x}_j \mathbf{b} + u_j < b)$ , the expected value of  $y_j | \mathbf{x}_j$  conditional on  $y_j | \mathbf{x}_j$  being in the interval (*a*, *b*), meaning that  $y_j | \mathbf{x}_j$  is censored. *a* and *b* are specified as they are for **pr()**. **e(*a*,*b*)** is not available with the two-step estimator.



`ystar(a,b)` calculates  $E(y_j^*)$ , where  $y_j^* = a$  if  $\mathbf{x}_j\mathbf{b} + u_j \leq a$ ,  $y_j^* = b$  if  $\mathbf{x}_j\mathbf{b} + u_j \geq b$ , and  $y_j^* = \mathbf{x}_j\mathbf{b} + u_j$  otherwise, meaning that  $y_j^*$  is truncated.  $a$  and  $b$  are specified as they are for `pr()`. `ystar(a,b)` is not available with the two-step estimator.

`scores`, not available with `twostep`, calculates equation-level score variables.

For models with one endogenous regressor, five new variables are created.

The first new variable will contain  $\partial \ln L / \partial (\mathbf{z}_i \boldsymbol{\delta})$ .

The second new variable will contain  $\partial \ln L / \partial (\mathbf{x}_i \boldsymbol{\Pi})$ .

The third new variable will contain  $\partial \ln L / \partial \alpha$ .

The fourth new variable will contain  $\partial \ln L / \partial \ln \sigma_{u|v}$ .

The fifth new variable will contain  $\partial \ln L / \partial \ln \sigma_v$ .

For models with  $p$  endogenous regressors,  $p + \{(p+1)(p+2)\}/2 + 1$  new variables are created.

The first new variable will contain  $\partial \ln L / \partial (\mathbf{z}_i \boldsymbol{\delta})$ .

The second through  $(p+1)$ th new score variables will contain  $\partial \ln L / \partial (\mathbf{x}_i \boldsymbol{\Pi}_k)$ ,  $k = 1, \dots, p$ , where  $\boldsymbol{\Pi}_k$  is the  $k$ th column of  $\boldsymbol{\Pi}$ .

The remaining score variables will contain the partial derivatives of  $\ln L$  with respect to  $s_{11}$ ,  $s_{21}$ ,  $\dots$ ,  $s_{p+1,1}$ ,  $s_{22}$ ,  $\dots$ ,  $s_{p+1,2}$ ,  $\dots$ ,  $s_{p+1,p+1}$ , where  $s_{m,n}$  denotes the  $(m,n)$  element of the Cholesky decomposition of the error covariance matrix.

## Remarks

Remarks are presented under the following headings:

*Marginal effects*

*Obtaining predicted values*

## Marginal effects

### ► Example 1

We can obtain average marginal effects by using the `margins` command after `ivtobit`. For the labor-supply model of [example 1](#) in [\[R\] ivtobit](#), suppose that we wanted to know the average marginal effects on the woman's expected income, conditional on her income being greater than \$10,000.

```
. use http://www.stata-press.com/data/r11/laborsup
. ivtobit fem_inc fem_educ kids (other_inc = male_educ), ll
 (output omitted)
. margins, dydx(*) predict(e(10, .))

Average marginal effects Number of obs = 500
Model VCE : OIM
Expression : E(fem_inc|fem_inc>10), predict(e(10, .))
dy/dx w.r.t. : other_inc fem_educ kids male_educ
```

|           | Delta-method |           | z     | P> z  | [95% Conf. Interval] |           |
|-----------|--------------|-----------|-------|-------|----------------------|-----------|
|           | dy/dx        | Std. Err. |       |       |                      |           |
| other_inc | -.3420189    | .0553591  | -6.18 | 0.000 | -.4505208            | -.233517  |
| fem_educ  | 1.237336     | .1534025  | 8.07  | 0.000 | .9366723             | 1.537999  |
| kids      | -1.252447    | .2725166  | -4.60 | 0.000 | -1.78657             | -.7183246 |
| male_educ | (omitted)    |           |       |       |                      |           |

In our sample, increasing the number of children in the family by one decreases the expected wage by \$1,252 on average (wages in our dataset are measured in thousands of dollars). `male_educ` has no effect because it appears only as an instrument.



## Obtaining predicted values

After fitting your model using `ivtobit`, you can obtain the linear prediction and its standard error for both the estimation sample and other samples using the `predict` command. If you used the maximum likelihood estimator, you can also obtain conditional expected values of the observed and latent dependent variables, the standard error of the forecast, and the probability of observing the dependent variable in a specified interval. See [U] 20 Estimation and postestimation commands and [R] `predict`.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

The linear prediction is calculated as  $z_i \hat{\delta}$ , where  $\hat{\delta}$  is the estimated value of  $\delta$ , and  $z_i$  and  $\delta$  are defined in (1a) of [R] `ivtobit`. Expected values and probabilities are calculated using the same formulas as those used by the standard exogenous tobit model.

## Also see

[R] `ivtobit` — Tobit model with continuous endogenous regressors

[U] 20 Estimation and postestimation commands

Syntax

```
jackknife exp_list [, options eform_option] : command
```

| options                                | description                                                                                                                    |
|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| Main                                   |                                                                                                                                |
| <code>eclass</code>                    | number of observations used is stored in <code>e(N)</code>                                                                     |
| <code>rclass</code>                    | number of observations used is stored in <code>r(N)</code>                                                                     |
| <code>n(exp)</code>                    | specify <i>exp</i> that evaluates to the number of observations used                                                           |
| Options                                |                                                                                                                                |
| <code>cluster(varlist)</code>          | variables identifying sample clusters                                                                                          |
| <code>idcluster(newvar)</code>         | create new cluster ID variable                                                                                                 |
| <code>saving(filename, ...)</code>     | save results to <i>filename</i> ; save statistics in double precision;<br>save results to <i>filename</i> every # replications |
| <code>keep</code>                      | keep pseudovalues                                                                                                              |
| <code>mse</code>                       | use MSE formula for variance estimation                                                                                        |
| Reporting                              |                                                                                                                                |
| <code>level(#)</code>                  | set confidence level; default is <code>level(95)</code>                                                                        |
| <code>notable</code>                   | suppress table of results                                                                                                      |
| <code>noheader</code>                  | suppress table header                                                                                                          |
| <code>nolegend</code>                  | suppress table legend                                                                                                          |
| <code>verbose</code>                   | display the full table legend                                                                                                  |
| <code>nodots</code>                    | suppress the replication dots                                                                                                  |
| <code>noisily</code>                   | display any output from <i>command</i>                                                                                         |
| <code>trace</code>                     | trace the <i>command</i>                                                                                                       |
| <code>title(text)</code>               | use <i>text</i> as title for jackknife results                                                                                 |
| <code>display_options</code>           | control spacing and display of omitted variables and base and empty cells                                                      |
| Advanced                               |                                                                                                                                |
| <code>nodrop</code>                    | do not drop observations                                                                                                       |
| <code>reject(exp)</code>               | identify invalid results                                                                                                       |
| <sup>†</sup> <code>eform_option</code> | display coefficient table in exponentiated form                                                                                |
| <sup>†</sup> <code>coeflegend</code>   | display coefficients' legend instead of coefficient table                                                                      |

<sup>†</sup>`eform_option` and `coeflegend` do not appear in the dialog.  
svy is allowed; see [SVY] [svy jackknife](#).  
All weight types supported by *command* are allowed except `aweight`s; see [U] [11.1.6 weight](#).  
See [U] [20 Estimation and postestimation commands](#) for more capabilities of estimation commands.

```

exp_list contains (name: elist)
 elist
 eexp
elist contains newvar = (exp)
 (exp)
eexp is specname
 [eqno]specname
specname is _b
 _b[]
 _se
 _se[]
eqno is ##
 name

```

*exp* is a standard Stata expression; see [U] 13 Functions and expressions.

Distinguish between [], which are to be typed, and [], which indicate optional arguments.

## Menu

Statistics > Resampling > Jackknife estimation

## Description

jackknife performs jackknife estimation. Typing

```
. jackknife exp_list: command
```

executes *command* once for each observation in the dataset, leaving the associated observation out of the calculations that make up *exp\_list*.

*command* defines the statistical command to be executed. Most Stata commands and user-written programs can be used with **jackknife**, as long as they follow standard Stata syntax and allow the **if** qualifier; see [U] 11 Language syntax. The **by** prefix may not be part of *command*.

*exp\_list* specifies the statistics to be collected from the execution of *command*. If *command* changes the contents in **e(b)**, *exp\_list* is optional and defaults to **\_b**.

Many estimation commands allow the **vce(jackknife)** option. For those commands, we recommend using **vce(jackknife)** over **jackknife** because the estimation command already handles clustering and other model-specific details for you. The **jackknife** prefix command is intended for use with nonestimation commands, such as **summarize**, user-written commands, or functions of coefficients.

**jknife** is a synonym for **jackknife**.

## Options

Main

**eclass**, **rclass**, and **n(exp)** specify where *command* saves the number of observations on which it based the calculated results. We strongly advise you to specify one of these options.

`eclass` specifies that *command* save the number of observations in `e(N)`.

`rclass` specifies that *command* save the number of observations in `r(N)`.

`n(exp)` specifies an expression that evaluates to the number of observations used. Specifying `n(r(N))` is equivalent to specifying the `rclass` option. Specifying `n(e(N))` is equivalent to specifying the `eclass` option. If *command* saves the number of observations in `r(N1)`, specify `n(r(N1))`.

If you specify no options, `jackknife` will assume `eclass` or `rclass`, depending on which of `e(N)` and `r(N)` is not missing (in that order). If both `e(N)` and `r(N)` are missing, `jackknife` assumes that all observations in the dataset contribute to the calculated result. If that assumption is incorrect, the reported standard errors will be incorrect. For instance, say that you specify

```
. jackknife coef=_b[x2]: myreg y x1 x2 x3
```

where `myreg` uses `e(n)` instead of `e(N)` to identify the number of observations used in calculations. Further assume that observation 42 in the dataset has `x3` equal to missing. The 42nd observation plays no role in obtaining the estimates, but `jackknife` has no way of knowing that and will use the wrong *N*. If, on the other hand, you specify

```
. jackknife coef=_b[x2], n(e(n)): myreg y x1 x2 x3
```

`jackknife` will notice that observation 42 plays no role. The `n(e(n))` option is specified because `myreg` is an estimation command but it saves the number of observations used in `e(n)` (instead of the standard `e(N)`). When `jackknife` runs the regression omitting the 42nd observation, `jackknife` will observe that `e(n)` has the same value as when `jackknife` previously ran the regression using all the observations. Thus `jackknife` will know that `myreg` did not use the observation.

### Options

`cluster(varlist)` specifies the variables identifying sample clusters. If `cluster()` is specified, one cluster is left out of each call to *command*, instead of 1 observation.

`idcluster(newvar)` creates a new variable containing a unique integer identifier for each resampled cluster, starting at 1 and leading up to the number of clusters. This option may be specified only when the `cluster()` option is specified. `idcluster()` helps identify the cluster to which a pseudovalue belongs.

`saving(filename[, suboptions])` creates a Stata data file (`.dta` file) consisting of (for each statistic in *exp\_list*) a variable containing the jackknife replicates.

`double` specifies that the results for each replication be stored as doubles, meaning 8-byte reals.

By default, they are stored as floats, meaning 4-byte reals. This option may be used without the `saving()` option to compute the variance estimates by using double precision.

`every(#)` specifies that results be written to disk every *#*th replication. `every()` should be specified only in conjunction with `saving()` when *command* takes a long time for each replication. This will allow recovery of partial results should some other software crash your computer. See [\[P\] postfile](#).

`replace` specifies that *filename* be overwritten, if it exists. This option does not appear in the dialog.

`keep` specifies that new variables be added to the dataset containing the pseudovalues of the requested statistics. For instance, if you typed

```
. jackknife coef=_b[x2], eclass keep: regress y x1 x2 x3
```

new variable `coef` would be added to the dataset containing the pseudovalues for `_b[x2]`. Let  $b$  be the value of `_b[x2]` when all observations are used to fit the model, and let  $b(j)$  be the value when the  $j$ th observation is omitted. The pseudovalues are defined as

$$\text{pseudovalue}_j = N \{b - b(j)\} + b(j)$$

where  $N$  is the number of observations used to produce  $b$ .

`mse` specifies that `jackknife` compute the variance by using deviations of the replicates from the observed value of the statistics based on the entire dataset. By default, `jackknife` computes the variance by using deviations of the pseudovalues from their mean.

#### Reporting

`level(#)`; see [R] [estimation options](#).

`notable` suppresses the display of the table of results.

`noheader` suppresses display of the table header. This option implies `nolegend`.

`nolegend` suppresses display of the table legend. The table legend identifies the rows of the table with the expressions they represent.

`verbose` specifies that the full table legend be displayed. By default, coefficients and standard errors are not displayed.

`nodots` suppresses display of the replication dots. By default, one dot character is displayed for each successful replication. A red 'x' is displayed if `command` returns an error or if one of the values in `exp_list` is missing.

`noisily` specifies that any output from `command` be displayed. This option implies the `nodots` option.

`trace` causes a trace of the execution of `command` to be displayed. This option implies the `noisily` option.

`title(text)` specifies a title to be displayed above the table of jackknife results; the default title is Jackknife results or what is produced in `e(title)` by an estimation command.

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

#### Advanced

`nodrop` prevents observations outside `e(sample)` and the `if` and `in` qualifiers from being dropped before the data are resampled.

`reject(exp)` identifies an expression that indicates when results should be rejected. When `exp` is true, the resulting values are reset to missing values.

The following options are available with `jackknife` but are not shown in the dialog:

`eform_option` causes the coefficient table to be displayed in exponentiated form: for each coefficient,  $\exp(b)$  rather than  $b$  is displayed. Standard errors and confidence intervals are also transformed. Display of the intercept, if any, is suppressed.

`command` determines which of the following are allowed (`eform(string)` and `eform` are always allowed):

| <i>eform_option</i>        | description                                          |
|----------------------------|------------------------------------------------------|
| <code>eform(string)</code> | use <i>string</i> for the column title               |
| <code>eform</code>         | exponentiated coefficient; <i>string</i> is “exp(b)” |
| <code>hr</code>            | hazard ratio; <i>string</i> is “Haz. Ratio”          |
| <code>irr</code>           | incidence-rate ratio; <i>string</i> is “IRR”         |
| <code>or</code>            | odds ratio; <i>string</i> is “Odds ratio”            |
| <code>rrr</code>           | relative-risk ratio; <i>string</i> is “RRR”          |

`coeflegend`; see [R] [estimation options](#).

## Remarks

Remarks are presented under the following headings:

[Introduction](#)  
[Jackknifed standard deviation](#)  
[Collecting multiple statistics](#)  
[Collecting coefficients](#)

## Introduction

Although the jackknife—developed in the late 1940s and early 1950s—is of largely historical interest today, it is still useful in searching for overly influential observations. This feature is often forgotten. In any case, the jackknife is

- an alternative, first-order unbiased estimator for a statistic;
- a data-dependent way to calculate the standard error of the statistic and to obtain significance levels and confidence intervals; and
- a way of producing measures called pseudovalues for each observation, reflecting the observation’s influence on the overall statistic.

The idea behind the simplest form of the jackknife—the one implemented here—is to repeatedly calculate the statistic in question, each time omitting just one of the dataset’s observations. Assume that our statistic of interest is the sample mean. Let  $y_j$  be the  $j$ th observation of our data on some measurement  $y$ , where  $j = 1, \dots, N$  and  $N$  is the sample size. If  $\bar{y}$  is the sample mean of  $y$  using the entire dataset and  $\bar{y}_{(j)}$  is the mean when the  $j$ th observation is omitted, then

$$\bar{y} = \frac{(N-1)\bar{y}_{(j)} + y_j}{N}$$

Solving for  $y_j$ , we obtain

$$y_j = N\bar{y} - (N-1)\bar{y}_{(j)}$$

These are the pseudovalues that `jackknife` calculates. To move this discussion beyond the sample mean, let  $\hat{\theta}$  be the value of our statistic (not necessarily the sample mean) using the entire dataset, and let  $\hat{\theta}_{(j)}$  be the computed value of our statistic with the  $j$ th observation omitted. The pseudovalue for the  $j$ th observation is

$$\hat{\theta}_j^* = N\hat{\theta} - (N-1)\hat{\theta}_{(j)}$$





```

. clear
. input x
 x
1. 0.1
2. 0.1
3. 0.1
4. 0.4
5. 0.5
6. 1.0
7. 1.1
8. 1.3
9. 1.9
10. 1.9
11. 4.7
12. end

. jackknife sd=r(sd), rclass keep: summarize x
(running summarize on estimation sample)

Jackknife replications (11)
-----|----- 1 -----|----- 2 -----|----- 3 -----|----- 4 -----|----- 5
.....

Jackknife results
 Number of obs = 11
 Replications = 11

 command: summarize x
 sd: r(sd)
 n(): r(N)

```

|    | Coef.    | Jackknife<br>Std. Err. | t    | P> t  | [95% Conf. Interval] |         |
|----|----------|------------------------|------|-------|----------------------|---------|
| sd | 1.343469 | .624405                | 2.15 | 0.057 | -.047792             | 2.73473 |

Interpreting the output, the standard deviation reported by `summarize mpg` is 1.34. The jackknife standard error is 0.62. The 95% confidence interval for the standard deviation is  $-0.048$  to  $2.73$ .

By specifying `keep`, `jackknife` creates in our dataset a new variable, `sd`, for the pseudovalues.

```
. list, sep(4)
```

|     | x   | sd       |
|-----|-----|----------|
| 1.  | .1  | 1.139977 |
| 2.  | .1  | 1.139977 |
| 3.  | .1  | 1.139977 |
| 4.  | .4  | .8893147 |
| 5.  | .5  | .824267  |
| 6.  | 1   | .632489  |
| 7.  | 1.1 | .6203189 |
| 8.  | 1.3 | .6218889 |
| 9.  | 1.9 | .835419  |
| 10. | 1.9 | .835419  |
| 11. | 4.7 | 7.703949 |

The jackknife estimate is the average of the `sd` variable, so `sd` contains the individual values of our statistic. We can see that the last observation is substantially larger than the others. The last observation is certainly an outlier, but whether that reflects the considerable information it contains or indicates that it should be excluded from analysis depends on the context of the problem. Here Mosteller

and Tukey created the dataset by sampling from an exponential distribution, so the observation is informative.

➤ Example 3

Let’s repeat the example above using the automobile dataset, obtaining the standard error of the standard deviation of mpg.

```
. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)

. jackknife sd=r(sd), rclass keep: summarize mpg
(running summarize on estimation sample)

Jackknife replications (74)
-----|----- 1 -----|----- 2 -----|----- 3 -----|----- 4 -----|----- 5
..... 50
.....

Jackknife results Number of obs = 74
 Replications = 74

 command: summarize mpg
 sd: r(sd)
 n(): r(N)
```

|    | Coef.    | Jackknife<br>Std. Err. | t    | P> t  | [95% Conf. Interval] |          |
|----|----------|------------------------|------|-------|----------------------|----------|
| sd | 5.785503 | .6072509               | 9.53 | 0.000 | 4.575254             | 6.995753 |

Let’s look at sd more carefully:

```
. summarize sd, detail

 pseudovalues: r(sd)

Percentiles Smallest
1% 2.870471 2.870471
5% 2.870471 2.870471
10% 2.906255 2.870471 Obs 74
25% 3.328489 2.870471 Sum of Wgt. 74
50% 3.948335
75% 6.844418 Largest
90% 9.597018 17.34316 Mean 5.817374
95% 17.34316 19.7617 Std. Dev. 5.22377
99% 38.60905 38.60905 Variance 27.28777
 Skewness 4.07202
 Kurtosis 23.37823

. list make mpg sd if sd > 30
```

|     | make      | mpg | sd       |
|-----|-----------|-----|----------|
| 71. | VW Diesel | 41  | 38.60905 |

Here the VW Diesel is the only diesel car in our dataset.

## Collecting multiple statistics

### ► Example 4

jackknife is not limited to collecting just one statistic. For instance, we can use `summarize`, `detail` and then obtain the jackknife estimate of the standard deviation and skewness. `summarize`, `detail` saves the standard deviation in `r(sd)` and the skewness in `r(skewness)`, so we might type

```
. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)

. jackknife sd=r(sd) skew=r(skewness), rclass: summarize mpg, detail
(running summarize on estimation sample)

Jackknife replications (74)
-----|----- 1 -----|----- 2 -----|----- 3 -----|----- 4 -----|----- 5
..... 50
.....

Jackknife results Number of obs = 74
 Replications = 74

 command: summarize mpg, detail
 sd: r(sd)
 skew: r(skewness)
 n(): r(N)
```

|      | Coef.    | Jackknife<br>Std. Err. | t    | P> t  | [95% Conf. Interval] |          |
|------|----------|------------------------|------|-------|----------------------|----------|
| sd   | 5.785503 | .6072509               | 9.53 | 0.000 | 4.575254             | 6.995753 |
| skew | .9487176 | .3367242               | 2.82 | 0.006 | .2776272             | 1.619808 |

◀

## Collecting coefficients

### ► Example 5

jackknife can also collect coefficients from estimation commands. For instance, using `auto.dta`, we might wish to obtain the jackknife standard errors of the coefficients from a regression in which we model the mileage of a car by its weight and trunk space. To do this, we could refer to the coefficients as `_b[weight]`, `_b[trunk]`, `_se[weight]`, and `_se[trunk]` in the `exp_list`, or we could simply use the extended expressions `_b`. In fact, jackknife assumes `_b` by default when used with estimation commands.

(Continued on next page)

|               |   |        |
|---------------|---|--------|
| Number of obs | = | 74     |
| Replications  | = | 74     |
| F( 2, 73)     | = | 78.10  |
| Prob > F      | = | 0.0000 |
| R-squared     | = | 0.6543 |
| Adj R-squared | = | 0.6446 |
| Root MSE      | = | 3.4492 |

| mpg    | Coef.     | Jackknife<br>Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|--------|-----------|------------------------|-------|-------|----------------------|-----------|
| weight | -.0056527 | .0010216               | -5.53 | 0.000 | -.0076887            | -.0036167 |
| trunk  | -.096229  | .1486236               | -0.65 | 0.519 | -.3924354            | .1999773  |
| _cons  | 39.68913  | 1.873324               | 21.19 | 0.000 | 35.9556              | 43.42266  |

```
. regress mpg weight trunk, vce(jackknife, nodots)
```

|               |   |        |
|---------------|---|--------|
| Number of obs | = | 74     |
| Replications  | = | 74     |
| F( 2, 73)     | = | 78.10  |
| Prob > F      | = | 0.0000 |
| R-squared     | = | 0.6543 |
| Adj R-squared | = | 0.6446 |
| Root MSE      | = | 3.4492 |

|        | Coef.     | Jackknife<br>Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|--------|-----------|------------------------|-------|-------|----------------------|-----------|
| weight | -.0056527 | .0010216               | -5.53 | 0.000 | -.0076887            | -.0036167 |
| trunk  | -.096229  | .1486236               | -0.65 | 0.519 | -.3924354            | .1999773  |
| _cons  | 39.68913  | 1.873324               | 21.19 | 0.000 | 35.9556              | 43.42266  |

◀

John Wilder Tukey (1915–2000) was born in Massachusetts. He studied chemistry at Brown and mathematics at Princeton and afterward worked at both Princeton and Bell Labs, as well as being involved in a great many government projects, consultancies, and committees. He made outstanding contributions to several areas of statistics, including time series, multiple comparisons, robust statistics, and exploratory data analysis. Tukey was extraordinarily energetic and inventive, not least in his use of terminology: he is credited with inventing the terms bit and software, in addition to ANOVA, boxplot, data analysis, hat matrix, jackknife, stem-and-leaf plot, trimming, and winsorizing, among many others. Tukey's direct and indirect impacts mark him as one of the greatest statisticians of all time.

## Saved results

jkknife saves the following in `e()`:

### Scalars

|                           |                                   |
|---------------------------|-----------------------------------|
| <code>e(N)</code>         | sample size                       |
| <code>e(N_reps)</code>    | number of complete replications   |
| <code>e(N_misreps)</code> | number of incomplete replications |
| <code>e(N_clust)</code>   | number of clusters                |
| <code>e(df_r)</code>      | degrees of freedom                |

### Macros

|                            |                                                                           |
|----------------------------|---------------------------------------------------------------------------|
| <code>e(cmdname)</code>    | command name from <i>command</i>                                          |
| <code>e(cmd)</code>        | same as <code>e(cmdname)</code> or <code>jackknife</code>                 |
| <code>e(command)</code>    | <i>command</i>                                                            |
| <code>e(cmdline)</code>    | command as typed                                                          |
| <code>e(prefix)</code>     | jackknife                                                                 |
| <code>e(wtype)</code>      | weight type                                                               |
| <code>e(wexp)</code>       | weight expression                                                         |
| <code>e(title)</code>      | title in estimation output                                                |
| <code>e(cluster)</code>    | cluster variables                                                         |
| <code>e(pseudo)</code>     | new variables containing pseudovalues                                     |
| <code>e(nfunction)</code>  | <code>e(N)</code> , <code>r(N)</code> , <code>n()</code> option, or empty |
| <code>e(exp#)</code>       | expression for the <i>#</i> th statistic                                  |
| <code>e(mse)</code>        | from <code>mse</code> option                                              |
| <code>e(vce)</code>        | jackknife                                                                 |
| <code>e(vcetype)</code>    | title used to label Std. Err.                                             |
| <code>e(properties)</code> | b V                                                                       |

### Matrices

|                              |                                      |
|------------------------------|--------------------------------------|
| <code>e(b)</code>            | observed statistics                  |
| <code>e(b_jk)</code>         | jackknife estimates                  |
| <code>e(V)</code>            | jackknife variance–covariance matrix |
| <code>e(V_modelbased)</code> | model-based variance                 |

When *exp\_list* is `_b`, `jackknife` will also carry forward most of the results already in `e()` from *command*.

## Methods and formulas

`jackknife` is implemented as an ado-file.

Let  $\hat{\theta}$  be the observed value of the statistic, that is, the value of the statistic calculated using the original dataset. Let  $\hat{\theta}_{(j)}$  be the value of the statistic computed by leaving out the *j*th observation (or cluster); thus  $j = 1, 2, \dots, N$  identifies an individual observation (or cluster), and *N* is the total number of observations (or clusters). The *j*th pseudovalue is given by

$$\hat{\theta}_j^* = \hat{\theta}_{(j)} + N\{\hat{\theta} - \hat{\theta}_{(j)}\}$$

When the `mse` option is specified, the standard error is estimated as

$$\widehat{\text{se}} = \left\{ \frac{N-1}{N} \sum_{j=1}^N (\hat{\theta}_{(j)} - \hat{\theta})^2 \right\}^{1/2}$$

and the jackknife estimate is

$$\bar{\theta}_{(.)} = \frac{1}{N} \sum_{j=1}^N \hat{\theta}_{(j)}$$

Otherwise, the standard error is estimated as

$$\widehat{\text{se}} = \left\{ \frac{1}{N(N-1)} \sum_{j=1}^N (\hat{\theta}_j^* - \bar{\theta}^*)^2 \right\}^{1/2} \quad \bar{\theta}^* = \frac{1}{N} \sum_{j=1}^N \hat{\theta}_j^*$$

where  $\bar{\theta}^*$  is the jackknife estimate. The variance–covariance matrix is similarly computed.

## References

- Brillinger, D. R. 2002. John W. Tukey: His life and professional contributions. *Annals of Statistics* 30: 1535–1575.
- Gould, W. W. 1995. [sg34: Jackknife estimation](#). *Stata Technical Bulletin* 24: 25–29. Reprinted in *Stata Technical Bulletin Reprints*, vol. 4, pp. 165–170. College Station, TX: Stata Press.
- Mooney, C. Z., and R. D. Duval. 1993. *Bootstrapping: A Nonparametric Approach to Statistical Inference*. Newbury Park, CA: Sage.
- Mosteller, F., and J. W. Tukey. 1977. *Data Analysis and Regression: A Second Course in Statistics*. Reading, MA: Addison–Wesley.
- Tukey, J. W. 1958. Bias and confidence in not-quite large samples. Abstract in *Annals of Mathematical Statistics* 29: 614.

## Also see

- [\[R\] jackknife postestimation](#) — Postestimation tools for jackknife
- [\[R\] bootstrap](#) — Bootstrap sampling and estimation
- [\[R\] permute](#) — Monte Carlo permutation tests
- [\[R\] simulate](#) — Monte Carlo simulations
- [\[SVY\] svy jackknife](#) — Jackknife estimation for survey data
- [\[U\] 13.5 Accessing coefficients and standard errors](#)
- [\[U\] 13.6 Accessing results from Stata commands](#)
- [\[U\] 20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `jackknife`:

| command                  | description                                                                                         |
|--------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>       | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estimates</code>   | cataloging estimation results                                                                       |
| <code>lincom</code>      | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| * <code>margins</code>   | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>       | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| * <code>predict</code>   | predictions, residuals, influence statistics, and other diagnostic measures                         |
| * <code>predictnl</code> | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>test</code>        | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>      | Wald tests of nonlinear hypotheses                                                                  |

\*This postestimation command is allowed only if it may be used after *command*.  
See the corresponding entries in the *Base Reference Manual* for details.

Syntax for predict

The syntax of `predict` (and even if `predict` is allowed) following `jackknife` depends on the *command* used with `jackknife`.

Methods and formulas

All postestimation commands listed above are implemented as ado-files.

Also see

- [R] [jackknife](#) — Jackknife estimation
- [U] [20 Estimation and postestimation commands](#)

Syntax

*Interrater agreement, two unique raters*

```
kap varname1 varname2 [if] [in] [weight] [, options]
```

*Weights for weighting disagreements*

```
kapwgt wgtid [1 \ # 1 [\ # # 1 ...]]
```

*Interrater agreement, nonunique raters, variables record ratings for each rater*

```
kap varname1 varname2 varname3 [...] [if] [in] [weight]
```

*Interrater agreement, nonunique raters, variables record frequency of ratings*

```
kappa varlist [if] [in]
```

| options                                              | description                                                              |
|------------------------------------------------------|--------------------------------------------------------------------------|
| Main                                                 |                                                                          |
| <u>tab</u>                                           | display table of assessments                                             |
| <u>wgt</u> (wgtid)                                   | specify how to weight disagreements; see <i>Options</i> for alternatives |
| <u>absolute</u>                                      | treat rating categories as absolute                                      |
| fweights are allowed; see [U] 11.1.6 <b>weight</b> . |                                                                          |

Menu

**kap: two unique raters**

Statistics > Epidemiology and related > Other > Interrater agreement, two unique raters

**kapwgt**

Statistics > Epidemiology and related > Other > Define weights for the above (kap)

**kap: nonunique raters**

Statistics > Epidemiology and related > Other > Interrater agreement, nonunique raters

**kappa**

Statistics > Epidemiology and related > Other > Interrater agreement, nonunique raters with frequencies



## Description

`kap` (first syntax) calculates the kappa-statistic measure of interrater agreement when there are two unique raters and two or more ratings.

`kapwgt` defines weights for use by `kap` in measuring the importance of disagreements.

`kap` (second syntax) and `kappa` calculate the kappa-statistic measure when there are two or more (nonunique) raters and two outcomes, more than two outcomes when the number of raters is fixed, and more than two outcomes when the number of raters varies. `kap` (second syntax) and `kappa` produce the same results; they merely differ in how they expect the data to be organized.

`kap` assumes that each observation is a subject. `varname1` contains the ratings by the first rater, `varname2` by the second rater, and so on.

`kappa` also assumes that each observation is a subject. The variables, however, record the frequencies with which ratings were assigned. The first variable records the number of times the first rating was assigned, the second variable records the number of times the second rating was assigned, and so on.

## Options

### Main

`tab` displays a tabulation of the assessments by the two raters.

`wgt(wgtid)` specifies that `wgtid` be used to weight disagreements. You can define your own weights by using `kapwgt`; `wgt()` then specifies the name of the user-defined matrix. For instance, you might define

```
. kapwgt mine 1 \ .8 1 \ 0 .8 1 \ 0 0 .8 1
```

and then

```
. kap rata ratb, wgt(mine)
```

Also, two prerecorded weights are available.

`wgt(w)` specifies weights  $1 - |i - j|/(k - 1)$ , where  $i$  and  $j$  index the rows and columns of the ratings by the two raters and  $k$  is the maximum number of possible ratings.

`wgt(w2)` specifies weights  $1 - \{(i - j)/(k - 1)\}^2$ .

`absolute` is relevant only if `wgt()` is also specified; see `wgt()` above. The `absolute` option modifies how  $i$ ,  $j$ , and  $k$  are defined and how corresponding entries are found in a user-defined weighting matrix. When `absolute` is not specified,  $i$  and  $j$  refer to the row and column index, not to the ratings themselves. Say that the ratings are recorded as  $\{0, 1, 1.5, 2\}$ . There are four ratings;  $k = 4$ , and  $i$  and  $j$  are still 1, 2, 3, and 4 in the formulas above. Index 3, for instance, corresponds to rating = 1.5. This system is convenient but can, with some data, lead to difficulties.

When `absolute` is specified, all ratings must be integers, and they must be coded from the set  $\{1, 2, 3, \dots\}$ . Not all values need be used; integer values that do not occur are simply assumed to be unobserved.

## Remarks

Remarks are presented under the following headings:

*Two raters*  
*More than two raters*

The kappa-statistic measure of agreement is scaled to be 0 when the amount of agreement is what would be expected to be observed by chance and 1 when there is perfect agreement. For intermediate values, [Landis and Koch \(1977a, 165\)](#) suggest the following interpretations:

|           |                |
|-----------|----------------|
| below 0.0 | Poor           |
| 0.00–0.20 | Slight         |
| 0.21–0.40 | Fair           |
| 0.41–0.60 | Moderate       |
| 0.61–0.80 | Substantial    |
| 0.81–1.00 | Almost perfect |

## Two raters

### ► Example 1

Consider the classification by two radiologists of 85 xeromammograms as normal, benign disease, suspicion of cancer, or cancer (a subset of the data from [Boyd et al. \[1982\]](#) and discussed in the context of kappa in [Altman \[1991, 403–405\]](#)).

```
. use http://www.stata-press.com/data/r11/rate2
(Altman p. 403)
. tabulate rada radb
```

| Radiologist<br>A's<br>assessment | Radiologist B's assessment |        |         |        | Total |
|----------------------------------|----------------------------|--------|---------|--------|-------|
|                                  | normal                     | benign | suspect | cancer |       |
| normal                           | 21                         | 12     | 0       | 0      | 33    |
| benign                           | 4                          | 17     | 1       | 0      | 22    |
| suspect                          | 3                          | 9      | 15      | 2      | 29    |
| cancer                           | 0                          | 0      | 0       | 1      | 1     |
| Total                            | 28                         | 38     | 16      | 3      | 85    |

Our dataset contains two variables: `rada`, radiologist A's assessment, and `radb`, radiologist B's assessment. Each observation is a patient.

We can obtain the kappa measure of interrater agreement by typing

```
. kap rada radb
```

| Agreement | Expected Agreement | Kappa  | Std. Err. | Z    | Prob>Z |
|-----------|--------------------|--------|-----------|------|--------|
| 63.53%    | 30.82%             | 0.4728 | 0.0694    | 6.81 | 0.0000 |

If each radiologist had made his determination randomly (but with probabilities equal to the overall proportions), we would expect the two radiologists to agree on 30.8% of the patients. In fact, they agreed on 63.5% of the patients, or 47.3% of the way between random agreement and perfect agreement. The amount of agreement indicates that we can reject the hypothesis that they are making their determinations randomly.

### ► Example 2: Weighted kappa, prerecorded weight w

There is a difference between two radiologists disagreeing about whether a xeromammogram indicates cancer or the suspicion of cancer and disagreeing about whether it indicates cancer or is normal. The weighted kappa attempts to deal with this. `kap` provides two “prerecorded” weights, `w` and `w2`:

```
. kap rada radb, wgt(w)
```

Ratings weighted by:

|        |        |        |        |
|--------|--------|--------|--------|
| 1.0000 | 0.6667 | 0.3333 | 0.0000 |
| 0.6667 | 1.0000 | 0.6667 | 0.3333 |
| 0.3333 | 0.6667 | 1.0000 | 0.6667 |
| 0.0000 | 0.3333 | 0.6667 | 1.0000 |

| Agreement | Expected Agreement | Kappa  | Std. Err. | Z    | Prob>Z |
|-----------|--------------------|--------|-----------|------|--------|
| 86.67%    | 69.11%             | 0.5684 | 0.0788    | 7.22 | 0.0000 |

The `w` weights are given by  $1 - |i - j|/(k - 1)$ , where  $i$  and  $j$  index the rows of columns of the ratings by the two raters and  $k$  is the maximum number of possible ratings. The weighting matrix is printed above the table. Here the rows and columns of the  $4 \times 4$  matrix correspond to the ratings normal, benign, suspicious, and cancerous.

A weight of 1 indicates that an observation should count as perfect agreement. The matrix has 1s down the diagonals—when both radiologists make the same assessment, they are in agreement. A weight of, say, 0.6667 means that they are in two-thirds agreement. In our matrix, they get that score if they are “one apart”—one radiologist assesses cancer and the other is merely suspicious, or one is suspicious and the other says benign, and so on. An entry of 0.3333 means that they are in one-third agreement, or, if you prefer, two-thirds disagreement. That is the score attached when they are “two apart”. Finally, they are in complete disagreement when the weight is zero, which happens only when they are three apart—one says cancer and the other says normal.

◀

### ► Example 3: Weighted kappa, prerecorded weight w2

The other prerecorded weight is `w2`, where the weights are given by  $1 - \{(i - j)/(k - 1)\}^2$ :

```
. kap rada radb, wgt(w2)
```

Ratings weighted by:

|        |        |        |        |
|--------|--------|--------|--------|
| 1.0000 | 0.8889 | 0.5556 | 0.0000 |
| 0.8889 | 1.0000 | 0.8889 | 0.5556 |
| 0.5556 | 0.8889 | 1.0000 | 0.8889 |
| 0.0000 | 0.5556 | 0.8889 | 1.0000 |

| Agreement | Expected Agreement | Kappa  | Std. Err. | Z    | Prob>Z |
|-----------|--------------------|--------|-----------|------|--------|
| 94.77%    | 84.09%             | 0.6714 | 0.1079    | 6.22 | 0.0000 |

The `w2` weight makes the categories even more alike and is probably inappropriate here.

◀

➤ Example 4: Weighted kappa, user-defined weights

In addition to using prerecorded weights, we can define our own weights with the `kapwgt` command. For instance, we might feel that suspicious and cancerous are reasonably similar, that benign and normal are reasonably similar, but that the suspicious/cancerous group is nothing like the benign/normal group:

```
. kapwgt xm 1 \ .8 1 \ 0 0 1 \ 0 0 .8 1
. kapwgt xm
1.0000
0.8000 1.0000
0.0000 0.0000 1.0000
0.0000 0.0000 0.8000 1.0000
```

We name the weights `xm`, and after the weight name, we enter the lower triangle of the weighting matrix, using `\` to separate rows. We have four outcomes, so we continued entering numbers until we had defined the fourth row of the weighting matrix. If we type `kapwgt` followed by a name and nothing else, it shows us the weights recorded under that name. Satisfied that we have entered them correctly, we now use the weights to recalculate kappa:

```
. kap rada radb, wgt(xm)
Ratings weighted by:
 1.0000 0.8000 0.0000 0.0000
 0.8000 1.0000 0.0000 0.0000
 0.0000 0.0000 1.0000 0.8000
 0.0000 0.0000 0.8000 1.0000
```

| Agreement | Expected Agreement | Kappa  | Std. Err. | Z    | Prob>Z |
|-----------|--------------------|--------|-----------|------|--------|
| 80.47%    | 52.67%             | 0.5874 | 0.0865    | 6.79 | 0.0000 |

❏ Technical note

In addition to using weights for weighting the differences in categories, you can specify Stata’s traditional weights for weighting the data. In the examples above, we have 85 observations in our dataset—one for each patient. If we only knew the table of outcomes—that there were 21 patients rated normal by both radiologists, etc.—it would be easier to enter the table into Stata and work from it. The easiest way to enter the data is with `tabi`; see [\[R\] tabulate twoway](#).

```
. tabi 21 12 0 0 \ 4 17 1 0 \ 3 9 15 2 \ 0 0 0 1, replace
```

| row   | col |    |    |   | Total |
|-------|-----|----|----|---|-------|
|       | 1   | 2  | 3  | 4 |       |
| 1     | 21  | 12 | 0  | 0 | 33    |
| 2     | 4   | 17 | 1  | 0 | 22    |
| 3     | 3   | 9  | 15 | 2 | 29    |
| 4     | 0   | 0  | 0  | 1 | 1     |
| Total | 28  | 38 | 16 | 3 | 85    |

Pearson chi2(9) = 77.8111 Pr = 0.000

`tabi` reported the Pearson  $\chi^2$  for this table, but we do not care about it. The important thing is that, with the `replace` option, `tabi` left the table in memory:

```
. list in 1/5
```

|    | row | col | pop |
|----|-----|-----|-----|
| 1. | 1   | 1   | 21  |
| 2. | 1   | 2   | 12  |
| 3. | 1   | 3   | 0   |
| 4. | 1   | 4   | 0   |
| 5. | 2   | 1   | 4   |

The variable `row` is radiologist A's assessment, `col` is radiologist B's assessment, and `pop` is the number so assessed by both. Thus

```
. kap row col [freq=pop]
```

| Agreement | Expected Agreement | Kappa  | Std. Err. | Z    | Prob>Z |
|-----------|--------------------|--------|-----------|------|--------|
| 63.53%    | 30.82%             | 0.4728 | 0.0694    | 6.81 | 0.0000 |

If we are going to keep these data, the names `row` and `col` are not indicative of what the data reflect. We could try (see [U] 12.6 Dataset, variable, and value labels)

```
. rename row rada
. rename col radb
. label var rada "Radiologist A's assessment"
. label var radb "Radiologist B's assessment"
. label define assess 1 normal 2 benign 3 suspect 4 cancer
. label values rada assess
. label values radb assess
. label data "Altman p. 403"
```

`kap`'s `tab` option, which can be used with or without weighted data, shows the table of assessments:

```
. kap rada radb [freq=pop], tab
```

| Radiologist<br>A's<br>assessment | Radiologist B's assessment |        |         |        | Total |
|----------------------------------|----------------------------|--------|---------|--------|-------|
|                                  | normal                     | benign | suspect | cancer |       |
| normal                           | 21                         | 12     | 0       | 0      | 33    |
| benign                           | 4                          | 17     | 1       | 0      | 22    |
| suspect                          | 3                          | 9      | 15      | 2      | 29    |
| cancer                           | 0                          | 0      | 0       | 1      | 1     |
| Total                            | 28                         | 38     | 16      | 3      | 85    |

| Agreement | Expected Agreement | Kappa  | Std. Err. | Z    | Prob>Z |
|-----------|--------------------|--------|-----------|------|--------|
| 63.53%    | 30.82%             | 0.4728 | 0.0694    | 6.81 | 0.0000 |



□ Technical note

You have data on individual patients. There are two raters, and the possible ratings are 1, 2, 3, and 4, but neither rater ever used rating 3:

```
. use http://www.stata-press.com/data/r11/rate2no3, clear
. tabulate ratera raterb
```

| ratera | raterb |   |    | Total |
|--------|--------|---|----|-------|
|        | 1      | 2 | 4  |       |
| 1      | 6      | 4 | 3  | 13    |
| 2      | 5      | 3 | 3  | 11    |
| 4      | 1      | 1 | 26 | 28    |
| Total  | 12     | 8 | 32 | 52    |

Here `kap` would determine that the ratings are from the set  $\{1, 2, 4\}$  because those were the only values observed. `kap` would expect a user-defined weighting matrix to be  $3 \times 3$ , and if it were not, `kap` would issue an error message. In the formula-based weights, the calculation would be based on  $i, j = 1, 2, 3$  corresponding to the three observed ratings  $\{1, 2, 4\}$ .

Specifying the `absolute` option would clarify that the ratings are 1, 2, 3, and 4; it just so happens that rating 3 was never assigned. If a user-defined weighting matrix were also specified, `kap` would expect it to be  $4 \times 4$  or larger (larger because we can think of the ratings being 1, 2, 3, 4, 5, ... and it just so happens that ratings 5, 6, ... were never observed, just as rating 3 was not observed). In the formula-based weights, the calculation would be based on  $i, j = 1, 2, 4$ .

```
. kap ratera raterb, wgt(w)
```

Ratings weighted by:

|        |        |        |
|--------|--------|--------|
| 1.0000 | 0.5000 | 0.0000 |
| 0.5000 | 1.0000 | 0.5000 |
| 0.0000 | 0.5000 | 1.0000 |

| Agreement | Expected Agreement | Kappa  | Std. Err. | Z    | Prob>Z |
|-----------|--------------------|--------|-----------|------|--------|
| 79.81%    | 57.17%             | 0.5285 | 0.1169    | 4.52 | 0.0000 |

```
. kap ratera raterb, wgt(w) absolute
```

Ratings weighted by:

|        |        |        |
|--------|--------|--------|
| 1.0000 | 0.6667 | 0.0000 |
| 0.6667 | 1.0000 | 0.3333 |
| 0.0000 | 0.3333 | 1.0000 |

| Agreement | Expected Agreement | Kappa  | Std. Err. | Z    | Prob>Z |
|-----------|--------------------|--------|-----------|------|--------|
| 81.41%    | 55.08%             | 0.5862 | 0.1209    | 4.85 | 0.0000 |

If all conceivable ratings are observed in the data, specifying `absolute` makes no difference. For instance, if rater A assigns ratings  $\{1, 2, 4\}$  and rater B assigns  $\{1, 2, 3, 4\}$ , the complete set of assigned ratings is  $\{1, 2, 3, 4\}$ , the same that `absolute` would specify. Without `absolute`, it makes no difference whether the ratings are coded  $\{1, 2, 3, 4\}$ ,  $\{0, 1, 2, 3\}$ ,  $\{1, 7, 9, 100\}$ ,  $\{0, 1, 1.5, 2.0\}$ , or otherwise.



## More than two raters

For more than two raters, the mathematics are such that the two raters are not considered unique. For instance, if there are three raters, there is no assumption that the three raters who rate the first subject are the same as the three raters who rate the second. Although we call this the “more than two raters” case, it can be used with two raters when the raters’ identities vary.

The nonunique rater case can be usefully broken down into three subcases: 1) there are two possible ratings, which we will call positive and negative; 2) there are more than two possible ratings, but the number of raters per subject is the same for all subjects; and 3) there are more than two possible ratings, and the number of raters per subject varies. `kappa` handles all these cases. To emphasize that there is no assumption of constant identity of raters across subjects, the variables specified contain counts of the number of raters rating the subject into a particular category.

Jacob Cohen (1923–1998) was born in New York City. After studying psychology at City College of New York and New York University, he worked as a medical psychologist until 1959 when he became a full professor in the Department of Psychology at New York University. He made many contributions to research methods, including the kappa measure. He persistently emphasized the value of multiple regression and the importance of power and of measuring effects rather than testing significance.

### ► Example 5: Two ratings

Fleiss, Levin, and Paik (2003, 612) offers the following hypothetical ratings by different sets of raters on 25 subjects:

| Subject | No. of<br>raters | No. of<br>pos. ratings | Subject | No. of<br>raters | No. of<br>pos. ratings |
|---------|------------------|------------------------|---------|------------------|------------------------|
| 1       | 2                | 2                      | 14      | 4                | 3                      |
| 2       | 2                | 0                      | 15      | 2                | 0                      |
| 3       | 3                | 2                      | 16      | 2                | 2                      |
| 4       | 4                | 3                      | 17      | 3                | 1                      |
| 5       | 3                | 3                      | 18      | 2                | 1                      |
| 6       | 4                | 1                      | 19      | 4                | 1                      |
| 7       | 3                | 0                      | 20      | 5                | 4                      |
| 8       | 5                | 0                      | 21      | 3                | 2                      |
| 9       | 2                | 0                      | 22      | 4                | 0                      |
| 10      | 4                | 4                      | 23      | 3                | 0                      |
| 11      | 5                | 5                      | 24      | 3                | 3                      |
| 12      | 3                | 3                      | 25      | 2                | 2                      |
| 13      | 4                | 4                      |         |                  |                        |

We have entered these data into Stata, and the variables are called `subject`, `raters`, and `pos`. `kappa`, however, requires that we specify variables containing the number of positive ratings and negative ratings, that is, `pos` and `raters-pos`:

```
. use http://www.stata-press.com/data/r11/p612
. gen neg = raters-pos
. kappa pos neg
```

Two-outcomes, multiple raters:

| Kappa  | Z    | Prob>Z |
|--------|------|--------|
| 0.5415 | 5.28 | 0.0000 |

We would have obtained the same results if we had typed `kappa neg pos`.



➤ Example 6: More than two ratings, constant number of raters, kappa

Each of 10 subjects is rated into one of three categories by five raters (Fleiss, Levin, and Paik 2003, 615):

```
. use http://www.stata-press.com/data/r11/p615, clear
. list
```

|     | subject | cat1 | cat2 | cat3 |
|-----|---------|------|------|------|
| 1.  | 1       | 1    | 4    | 0    |
| 2.  | 2       | 2    | 0    | 3    |
| 3.  | 3       | 0    | 0    | 5    |
| 4.  | 4       | 4    | 0    | 1    |
| 5.  | 5       | 3    | 0    | 2    |
| 6.  | 6       | 1    | 4    | 0    |
| 7.  | 7       | 5    | 0    | 0    |
| 8.  | 8       | 0    | 4    | 1    |
| 9.  | 9       | 1    | 0    | 4    |
| 10. | 10      | 3    | 0    | 2    |

We obtain the kappa statistic:

```
. kappa cat1-cat3
```

| Outcome  | Kappa  | Z    | Prob>Z |
|----------|--------|------|--------|
| cat1     | 0.2917 | 2.92 | 0.0018 |
| cat2     | 0.6711 | 6.71 | 0.0000 |
| cat3     | 0.3490 | 3.49 | 0.0002 |
| combined | 0.4179 | 5.83 | 0.0000 |

The first part of the output shows the results of calculating kappa for each of the categories separately against an amalgam of the remaining categories. For instance, the `cat1` line is the two-rating kappa, where positive is `cat1` and negative is `cat2` or `cat3`. The test statistic, however, is calculated differently (see [Methods and formulas](#)). The combined kappa is the appropriately weighted average of the individual kappas. There is considerably less agreement about the rating of subjects into the first category than there is for the second.



➤ Example 7: More than two ratings, constant number of raters, kap

Now suppose that we have the same data as in the previous example but that the data are organized differently:



```
. use http://www.stata-press.com/data/r11/p615b
. list
```

|     | subject | rater1 | rater2 | rater3 | rater4 | rater5 |
|-----|---------|--------|--------|--------|--------|--------|
| 1.  | 1       | 1      | 2      | 2      | 2      | 2      |
| 2.  | 2       | 1      | 1      | 3      | 3      | 3      |
| 3.  | 3       | 3      | 3      | 3      | 3      | 3      |
| 4.  | 4       | 1      | 1      | 1      | 1      | 3      |
| 5.  | 5       | 1      | 1      | 1      | 3      | 3      |
| 6.  | 6       | 1      | 2      | 2      | 2      | 2      |
| 7.  | 7       | 1      | 1      | 1      | 1      | 1      |
| 8.  | 8       | 2      | 2      | 2      | 2      | 3      |
| 9.  | 9       | 1      | 3      | 3      | 3      | 3      |
| 10. | 10      | 1      | 1      | 1      | 3      | 3      |

Here we would use `kap` rather than `kappa` because the variables record ratings for each rater.

```
. kap rater1 rater2 rater3 rater4 rater5
```

There are 5 raters per subject:

| Outcome  | Kappa  | Z    | Prob>Z |
|----------|--------|------|--------|
| 1        | 0.2917 | 2.92 | 0.0018 |
| 2        | 0.6711 | 6.71 | 0.0000 |
| 3        | 0.3490 | 3.49 | 0.0002 |
| combined | 0.4179 | 5.83 | 0.0000 |

It does not matter which rater is which when there are more than two raters.



## ► Example 8: More than two ratings, varying number of raters, kappa

In this unfortunate case, kappa can be calculated, but there is no test statistic for testing against  $\kappa > 0$ . We do nothing differently—`kappa` calculates the total number of raters for each subject, and, if it is not a constant, `kappa` suppresses the calculation of test statistics.

```
. use http://www.stata-press.com/data/r11/rvary
. list
```

|     | subject | cat1 | cat2 | cat3 |
|-----|---------|------|------|------|
| 1.  | 1       | 1    | 3    | 0    |
| 2.  | 2       | 2    | 0    | 3    |
| 3.  | 3       | 0    | 0    | 5    |
| 4.  | 4       | 4    | 0    | 1    |
| 5.  | 5       | 3    | 0    | 2    |
| 6.  | 6       | 1    | 4    | 0    |
| 7.  | 7       | 5    | 0    | 0    |
| 8.  | 8       | 0    | 4    | 1    |
| 9.  | 9       | 1    | 0    | 2    |
| 10. | 10      | 3    | 0    | 2    |

```
. kappa cat1-cat3
```

| Outcome  | Kappa  | Z | Prob>Z |
|----------|--------|---|--------|
| cat1     | 0.2685 | . | .      |
| cat2     | 0.6457 | . | .      |
| cat3     | 0.2938 | . | .      |
| combined | 0.3816 | . | .      |

Note: number of ratings per subject vary; cannot calculate test statistics.

➤ Example 9: More than two ratings, varying number of raters, kap

This case is similar to the previous example, but the data are organized differently:

```
. use http://www.stata-press.com/data/r11/rvary2
. list
```

|     | subject | rater1 | rater2 | rater3 | rater4 | rater5 |
|-----|---------|--------|--------|--------|--------|--------|
| 1.  | 1       | 1      | 2      | 2      | .      | 2      |
| 2.  | 2       | 1      | 1      | 3      | 3      | 3      |
| 3.  | 3       | 3      | 3      | 3      | 3      | 3      |
| 4.  | 4       | 1      | 1      | 1      | 1      | 3      |
| 5.  | 5       | 1      | 1      | 1      | 3      | 3      |
| 6.  | 6       | 1      | 2      | 2      | 2      | 2      |
| 7.  | 7       | 1      | 1      | 1      | 1      | 1      |
| 8.  | 8       | 2      | 2      | 2      | 2      | 3      |
| 9.  | 9       | 1      | 3      | .      | .      | 3      |
| 10. | 10      | 1      | 1      | 1      | 3      | 3      |

Here we specify kap instead of kappa because the variables record ratings for each rater.

```
. kap rater1-rater5
```

There are between 3 and 5 (median = 5.00) raters per subject:

| Outcome  | Kappa  | Z | Prob>Z |
|----------|--------|---|--------|
| 1        | 0.2685 | . | .      |
| 2        | 0.6457 | . | .      |
| 3        | 0.2938 | . | .      |
| combined | 0.3816 | . | .      |

Note: number of ratings per subject vary; cannot calculate test statistics.

## Saved results

kap and kappa save the following in r():

Scalars

|           |                                             |          |                                    |
|-----------|---------------------------------------------|----------|------------------------------------|
| r(N)      | number of subjects (kap only)               | r(kappa) | kappa                              |
| r(prop_o) | observed proportion of agreement (kap only) | r(z)     | z statistic                        |
| r(prop_e) | expected proportion of agreement (kap only) | r(se)    | standard error for kappa statistic |

## Methods and formulas

kap, kapwgt, and kappa are implemented as ado-files.

The kappa statistic was first proposed by [Cohen \(1960\)](#). The generalization for weights reflecting the relative seriousness of each possible disagreement is due to [Cohen \(1968\)](#). The analysis-of-variance approach for  $k = 2$  and  $m \geq 2$  is due to [Landis and Koch \(1977b\)](#). See [Altman \(1991, 403–409\)](#) or [Dunn \(2000, chap. 2\)](#) for an introductory treatment and [Fleiss, Levin, and Paik \(2003, chap. 18\)](#) for a more detailed treatment. All formulas below are as presented in [Fleiss, Levin, and Paik \(2003\)](#). Let  $m$  be the number of raters, and let  $k$  be the number of rating outcomes.

Methods and formulas are presented under the following headings:

kap:  $m = 2$   
 kappa:  $m > 2, k = 2$   
 kappa:  $m > 2, k > 2$

### kap: m = 2

Define  $w_{ij}$  ( $i = 1, \dots, k$  and  $j = 1, \dots, k$ ) as the weights for agreement and disagreement ( $\text{wgt}()$ ), or, if the data are not weighted, define  $w_{ii} = 1$  and  $w_{ij} = 0$  for  $i \neq j$ . If  $\text{wgt}(\mathbf{w})$  is specified,  $w_{ij} = 1 - |i - j|/(k - 1)$ . If  $\text{wgt}(\mathbf{w2})$  is specified,  $w_{ij} = 1 - \{(i - j)/(k - 1)\}^2$ .

The observed proportion of agreement is

$$p_o = \sum_{i=1}^k \sum_{j=1}^k w_{ij} p_{ij}$$

where  $p_{ij}$  is the fraction of ratings  $i$  by the first rater and  $j$  by the second. The expected proportion of agreement is

$$p_e = \sum_{i=1}^k \sum_{j=1}^k w_{ij} p_{i\cdot} p_{\cdot j}$$

where  $p_{i\cdot} = \sum_j p_{ij}$  and  $p_{\cdot j} = \sum_i p_{ij}$ .

Kappa is given by  $\hat{\kappa} = (p_o - p_e)/(1 - p_e)$ .

The standard error of  $\hat{\kappa}$  for testing against 0 is

$$\hat{s}_0 = \frac{1}{(1 - p_e)\sqrt{n}} \left( \left[ \sum_i \sum_j p_{i\cdot} p_{\cdot j} \{w_{ij} - (\bar{w}_{i\cdot} + \bar{w}_{\cdot j})\}^2 \right] - p_e^2 \right)^{1/2}$$

where  $n$  is the number of subjects being rated,  $\bar{w}_{i\cdot} = \sum_j p_{\cdot j} w_{ij}$ , and  $\bar{w}_{\cdot j} = \sum_i p_{i\cdot} w_{ij}$ . The test statistic  $Z = \hat{\kappa}/\hat{s}_0$  is assumed to be distributed  $N(0, 1)$ .

### kappa: m > 2, k = 2

Each subject  $i$ ,  $i = 1, \dots, n$ , is found by  $x_i$  of  $m_i$  raters to be positive (the choice as to what is labeled positive is arbitrary).

The overall proportion of positive ratings is  $\bar{p} = \sum_i x_i / (n\bar{m})$ , where  $\bar{m} = \sum_i m_i / n$ . The between-subjects mean square is (approximately)

$$B = \frac{1}{n} \sum_i \frac{(x_i - m_i \bar{p})^2}{m_i}$$

and the within-subject mean square is

$$W = \frac{1}{n(\bar{m} - 1)} \sum_i \frac{x_i(m_i - x_i)}{m_i}$$

Kappa is then defined as

$$\hat{\kappa} = \frac{B - W}{B + (\bar{m} - 1)W}$$

The standard error for testing against 0 (Fleiss and Cuzick 1979) is approximately equal to and is calculated as

$$\hat{s}_0 = \frac{1}{(\bar{m} - 1)\sqrt{n\bar{m}_H}} \left\{ 2(\bar{m}_H - 1) + \frac{(\bar{m} - \bar{m}_H)(1 - 4\bar{p}\bar{q})}{\bar{m}\bar{p}\bar{q}} \right\}^{1/2}$$

where  $\bar{m}_H$  is the harmonic mean of  $m_i$  and  $\bar{q} = 1 - \bar{p}$ .

The test statistic  $Z = \hat{\kappa}/\hat{s}_0$  is assumed to be distributed  $N(0, 1)$ .

### kappa: m > 2, k > 2

Let  $x_{ij}$  be the number of ratings on subject  $i$ ,  $i = 1, \dots, n$ , into category  $j$ ,  $j = 1, \dots, k$ . Define  $\bar{p}_j$  as the overall proportion of ratings in category  $j$ ,  $\bar{q}_j = 1 - \bar{p}_j$ , and let  $\hat{\kappa}_j$  be the kappa statistic given above for  $k = 2$  when category  $j$  is compared with the amalgam of all other categories. Kappa is

$$\bar{\kappa} = \frac{\sum_j \bar{p}_j \bar{q}_j \hat{\kappa}_j}{\sum_j \bar{p}_j \bar{q}_j}$$

(Landis and Koch 1977b). In the case where the number of raters per subject,  $\sum_j x_{ij}$ , is a constant  $m$  for all  $i$ , Fleiss, Nee, and Landis (1979) derived the following formulas for the approximate standard errors. The standard error for testing  $\hat{\kappa}_j$  against 0 is

$$\hat{s}_j = \left\{ \frac{2}{nm(m-1)} \right\}^{1/2}$$

and the standard error for testing  $\bar{\kappa}$  is

$$\bar{s} = \frac{\sqrt{2}}{\sum_j \bar{p}_j \bar{q}_j \sqrt{nm(m-1)}} \left\{ \left( \sum_j \bar{p}_j \bar{q}_j \right)^2 - \sum_j \bar{p}_j \bar{q}_j (\bar{q}_j - \bar{p}_j) \right\}^{1/2}$$

## References

- Abramson, J. H., and Z. H. Abramson. 2001. *Making Sense of Data: A Self-Instruction Manual on the Interpretation of Epidemiological Data*. 3rd ed. New York: Oxford University Press.
- Altman, D. G. 1991. *Practical Statistics for Medical Research*. London: Chapman & Hall/CRC.
- Boyd, N. F., C. Wolfson, M. Moskowitz, T. Carlile, M. Petitclerc, H. A. Ferri, E. Fishell, A. Gregoire, M. Kiernan, J. D. Longley, I. S. Simor, and A. B. Miller. 1982. Observer variation in the interpretation of xeromammograms. *Journal of the National Cancer Institute* 68: 357–363.
- Campbell, M. J., D. Machin, and S. J. Walters. 2007. *Medical Statistics: A Textbook for the Health Sciences*. 4th ed. Chichester, UK: Wiley.
- Cohen, J. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20: 37–46.
- . 1968. Weighted kappa: Nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological Bulletin* 70: 213–220.
- Cox, N. J. 2006. Assessing agreement of measurements and predictions in geomorphology. *Geomorphology* 76: 332–346.
- Dunn, G. 2000. *Statistics in Psychiatry*. London: Arnold.
- Fleiss, J. L., and J. Cuzick. 1979. The reliability of dichotomous judgments: Unequal numbers of judges per subject. *Applied Psychological Measurement* 3: 537–542.
- Fleiss, J. L., B. Levin, and M. C. Paik. 2003. *Statistical Methods for Rates and Proportions*. 3rd ed. New York: Wiley.
- Fleiss, J. L., J. C. M. Nee, and J. R. Landis. 1979. Large sample variance of kappa in the case of different sets of raters. *Psychological Bulletin* 86: 974–977.
- Gould, W. W. 1997. [stata49: Interrater agreement](#). *Stata Technical Bulletin* 40: 2–8. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 20–28. College Station, TX: Stata Press.
- Landis, J. R., and G. G. Koch. 1977a. The measurement of observer agreement for categorical data. *Biometrics* 33: 159–174.
- . 1977b. A one-way components of variance model for categorical data. *Biometrics* 33: 671–679.
- Reichenheim, M. E. 2000. [sxd3: Sample size for the kappa-statistic of interrater agreement](#). *Stata Technical Bulletin* 58: 41–45. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 382–387. College Station, TX: Stata Press.
- . 2004. [Confidence intervals for the kappa statistic](#). *Stata Journal* 4: 421–428.
- Shrout, P. E. 2001. Jacob Cohen (1923–1998). *American Psychologist* 56: 166.
- Steichen, T. J., and N. J. Cox. 1998a. [sg84: Concordance correlation coefficient](#). *Stata Technical Bulletin* 43: 35–39. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 137–143. College Station, TX: Stata Press.
- . 1998b. [sg84.1: Concordance correlation coefficient, revisited](#). *Stata Technical Bulletin* 45: 21–23. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 143–145. College Station, TX: Stata Press.
- . 2000a. [sg84.3: Concordance correlation coefficient: Minor corrections](#). *Stata Technical Bulletin* 58: 9. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, p. 137. College Station, TX: Stata Press.
- . 2000b. [sg84.2: Concordance correlation coefficient: Update for Stata 6](#). *Stata Technical Bulletin* 54: 25–26. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 169–170. College Station, TX: Stata Press.
- . 2002. [A note on the concordance correlation coefficient](#). *Stata Journal* 2: 183–189.

Syntax

```
kdensity varname [if] [in] [weight] [, options]
```

| <i>options</i>                                                                          | description                                                                                                                    |
|-----------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| Main                                                                                    |                                                                                                                                |
| <u>k</u> ernel( <i>kernel</i> )                                                         | specify kernel function; default is <code>kernel(epanechnikov)</code>                                                          |
| <u>b</u> width( <i>#</i> )                                                              | half-width of kernel                                                                                                           |
| <u>g</u> enerate( <i>newvar</i> <sub><i>x</i></sub> <i>newvar</i> <sub><i>d</i></sub> ) | store the estimation points in <i>newvar</i> <sub><i>x</i></sub> and the density estimate in <i>newvar</i> <sub><i>d</i></sub> |
| <u>n</u> ( <i>#</i> )                                                                   | estimate density using <i>#</i> points; default is <code>min(<i>N</i>, 50)</code>                                              |
| <u>a</u> t( <i>var</i> <sub><i>x</i></sub> )                                            | estimate density using the values specified by <i>var</i> <sub><i>x</i></sub>                                                  |
| <u>n</u> ograph                                                                         | suppress graph                                                                                                                 |
| Kernel plot                                                                             |                                                                                                                                |
| <i>cline_options</i>                                                                    | affect rendition of the plotted kernel density estimate                                                                        |
| Density plots                                                                           |                                                                                                                                |
| <u>n</u> ormal                                                                          | add normal density to the graph                                                                                                |
| <u>norm</u> opts( <i>cline_options</i> )                                                | affect rendition of normal density                                                                                             |
| <u>s</u> tudent( <i>#</i> )                                                             | add Student's <i>t</i> density with <i>#</i> degrees of freedom to the graph                                                   |
| <u>st</u> opts( <i>cline_options</i> )                                                  | affect rendition of the Student's <i>t</i> density                                                                             |
| Add plots                                                                               |                                                                                                                                |
| <u>a</u> ddplot( <i>plot</i> )                                                          | add other plots to the generated graph                                                                                         |
| Y axis, X axis, Titles, Legend, Overall                                                 |                                                                                                                                |
| <i>twoway_options</i>                                                                   | any options other than <code>by()</code> documented in [G] <a href="#">twoway_options</a>                                      |
| <hr/>                                                                                   |                                                                                                                                |
| <i>kernel</i>                                                                           | description                                                                                                                    |
| <u>e</u> panechnikov                                                                    | Epanechnikov kernel function; the default                                                                                      |
| <u>e</u> pan2                                                                           | alternative Epanechnikov kernel function                                                                                       |
| <u>b</u> iweight                                                                        | biweight kernel function                                                                                                       |
| <u>c</u> osine                                                                          | cosine trace kernel function                                                                                                   |
| <u>g</u> aussian                                                                        | Gaussian kernel function                                                                                                       |
| <u>p</u> arzen                                                                          | Parzen kernel function                                                                                                         |
| <u>r</u> ectangle                                                                       | rectangle kernel function                                                                                                      |
| <u>t</u> riangle                                                                        | triangle kernel function                                                                                                       |

fweights, aweights, and iweights are allowed; see [U] [11.1.6 weight](#).

## Menu

Statistics > Nonparametric analysis > Kernel density estimation

## Description

**kdensity** produces kernel density estimates and graphs the result.

## Options

### Main

**kernel**(*kernel*) specifies the kernel function for use in calculating the kernel density estimate. The default kernel is the Epanechnikov kernel (**epanechnikov**).

**bwidth**(*#*) specifies the half-width of the kernel, the width of the density window around each point. If **bwidth**() is not specified, the “optimal” width is calculated and used. The optimal width is the width that would minimize the mean integrated squared error if the data were Gaussian and a Gaussian kernel were used, so it is not optimal in any global sense. In fact, for multimodal and highly skewed densities, this width is usually too wide and oversmooths the density ([Silverman 1992](#)).

**generate**(*newvar<sub>x</sub>* *newvar<sub>d</sub>*) stores the results of the estimation. *newvar<sub>x</sub>* will contain the points at which the density is estimated. *newvar<sub>d</sub>* will contain the density estimate.

**n**(*#*) specifies the number of points at which the density estimate is to be evaluated. The default is  $\min(N, 50)$ , where  $N$  is the number of observations in memory.

**at**(*var<sub>x</sub>*) specifies a variable that contains the values at which the density should be estimated. This option allows you to more easily obtain density estimates for different variables or different subsamples of a variable and then overlay the estimated densities for comparison.

**nograph** suppresses the graph. This option is often used with the **generate**() option.

### Kernel plot

*cline\_options* affect the rendition of the plotted kernel density estimate. See [\[G\] \*cline\\_options\*](#).

### Density plots

**normal** requests that a normal density be overlaid on the density estimate for comparison.

**normopts**(*cline\_options*) specifies details about the rendition of the normal curve, such as the color and style of line used. See [\[G\] \*cline\\_options\*](#).

**student**(*#*) specifies that a Student’s  $t$  density with  $\#$  degrees of freedom be overlaid on the density estimate for comparison.

**stopts**(*cline\_options*) affects the rendition of the Student’s  $t$  density. See [\[G\] \*cline\\_options\*](#).

### Add plots

**addplot**(*plot*) provides a way to add other plots to the generated graph. See [\[G\] \*addplot\\_option\*](#).

---

Y axis, X axis, Titles, Legend, Overall

---

`twoway_options` are any of the options documented in [G] [twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

## Remarks

Kernel density estimators approximate the density  $f(x)$  from observations on  $x$ . Histograms do this, too, and the histogram itself is a kind of kernel density estimate. The data are divided into nonoverlapping intervals, and counts are made of the number of data points within each interval. Histograms are bar graphs that depict these frequency counts—the bar is centered at the midpoint of each interval—and its height reflects the average number of data points in the interval.

In more general kernel density estimates, the range is still divided into intervals, and estimates of the density at the center of intervals are produced. One difference is that the intervals are allowed to overlap. We can think of sliding the interval—called a window—along the range of the data and collecting the center-point density estimates. The second difference is that, rather than merely counting the number of observations in a window, a kernel density estimator assigns a weight between 0 and 1—based on the distance from the center of the window—and sums the weighted values. The function that determines these weights is called the kernel.

Kernel density estimates have the advantages of being smooth and of being independent of the choice of origin (corresponding to the location of the bins in a histogram).

See [Salgado-Ugarte, Shimizu, and Taniuchi \(1993\)](#) and [Fox \(1990\)](#) for discussions of kernel density estimators that stress their use as exploratory data-analysis tools.

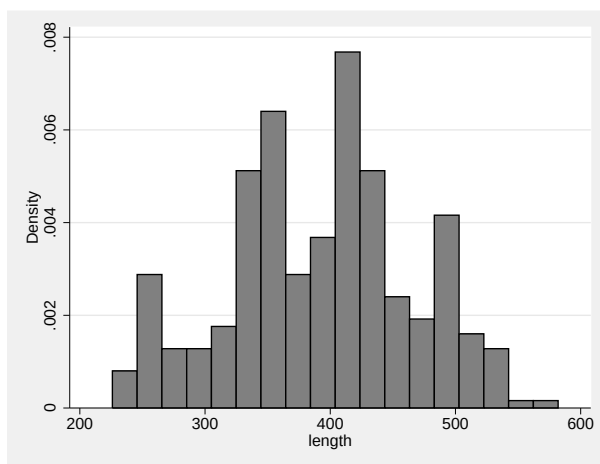
[Cox \(2007\)](#) gives a lucid introductory tutorial on kernel density estimation with several Stata produced examples. He provides tips and tricks for working with skewed or bounded distributions and applying the same techniques to estimate the intensity function of a point process.

### ► Example 1: Histogram and kernel density estimate

[Goeden \(1978\)](#) reports data consisting of 316 length observations of coral trout. We wish to investigate the underlying density of the lengths. To begin on familiar ground, we might draw a histogram. In [R] [histogram](#), we suggest setting the bins to  $\min(\sqrt{n}, 10 \cdot \log_{10} n)$ , which for  $n = 316$  is roughly 18:

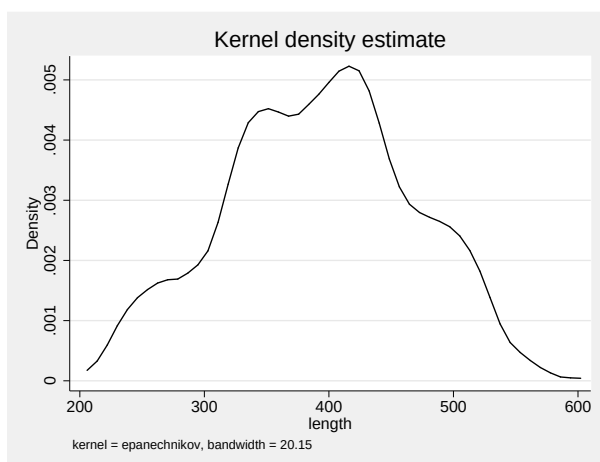


```
. use http://www.stata-press.com/data/r11/trocolen
. histogram length, bin(18)
(bin=18, start=226, width=19.777778)
```



The kernel density estimate, on the other hand, is smooth.

```
. kdensity length
```



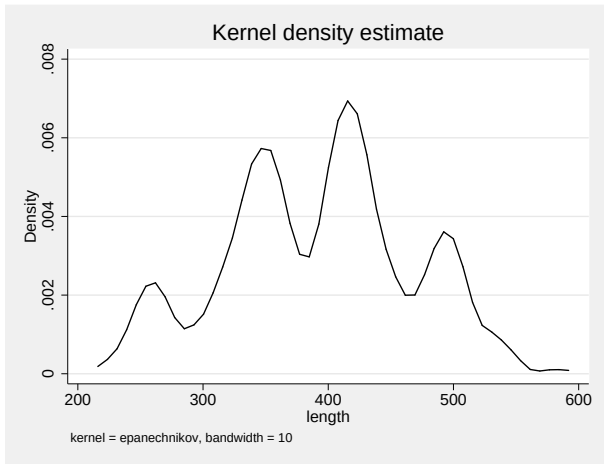
Kernel density estimators are, however, sensitive to an assumption, just as are histograms. In histograms, we specify a number of bins. For kernel density estimators, we specify a width. In the graph above, we used the default width. `kdensity` is smarter than `twoway histogram` in that its default width is not a fixed constant. Even so, the default width is not necessarily best.

`kdensity` saves the width in the returned scalar `bwidth`, so typing `display r(bwidth)` reveals it. Doing this, we discover that the width is approximately 20.

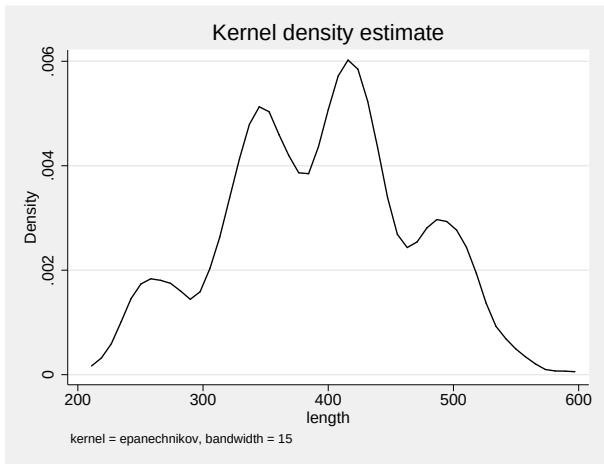
Widths are similar to the inverse of the number of bins in a histogram in that smaller widths provide more detail. The units of the width are the units of  $x$ , the variable being analyzed. The width is specified as a half-width, meaning that the kernel density estimator with half-width 20 corresponds to sliding a window of size 40 across the data.

We can specify half-widths for ourselves by using the `bwidth()` option. Smaller widths do not smooth the density as much:

```
. kdensity length, bwidth(10)
```



```
. kdensity length, bwidth(15)
```



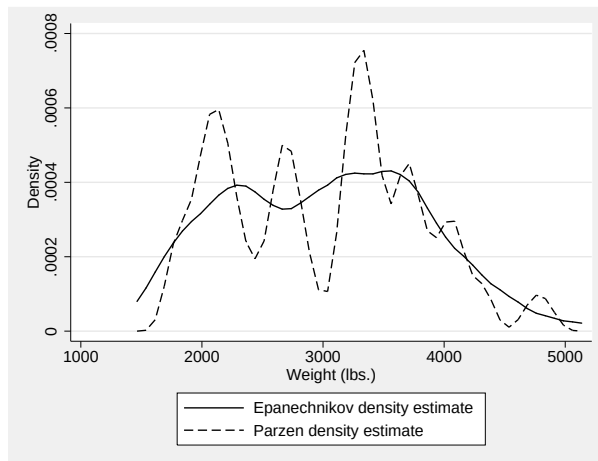
### ➤ Example 2: Different kernels can produce different results

When widths are held constant, different kernels can produce surprisingly different results. This is really an attribute of the kernel and width combination; for a given width, some kernels are more sensitive than others at identifying peaks in the density estimate.

We can see this when using a dataset with lots of peaks. In the automobile dataset, we characterize the density of `weight`, the weight of the vehicles. Below we compare the Epanechnikov and Parzen kernels.

```
. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)

. kdensity weight, kernel(epanechnikov) nograph generate(x epan)
. kdensity weight, kernel(parzen) nograph generate(x2 parzen)
. label var epan "Epanechnikov density estimate"
. label var parzen "Parzen density estimate"
. line epan parzen x, sort ytitle(Density) legend(cols(1))
```



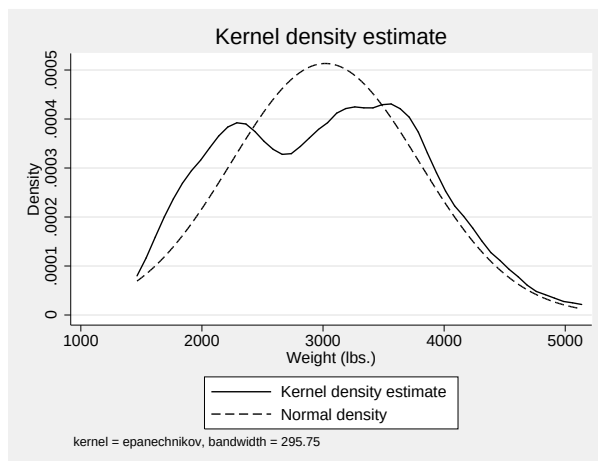
We did not specify a width, so we obtained the default width. That width is not a function of the selected kernel, but of the data. See [Methods and formulas](#) for the calculation of the optimal width.



### ► Example 3: Density with overlaid normal density

In examining the density estimates, we may wish to overlay a normal density or a Student's  $t$  density for comparison. Using automobile weights, we can get an idea of the distance from normality by using the normal option.

```
. kdensity weight, kernel(epanechnikov) normal
```



### ► Example 4: Compare two densities

We also may want to compare two or more densities. In this example, we will compare the density estimates of the weights for the foreign and domestic cars.

```
. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)

. kdensity weight, nograph generate(x fx)

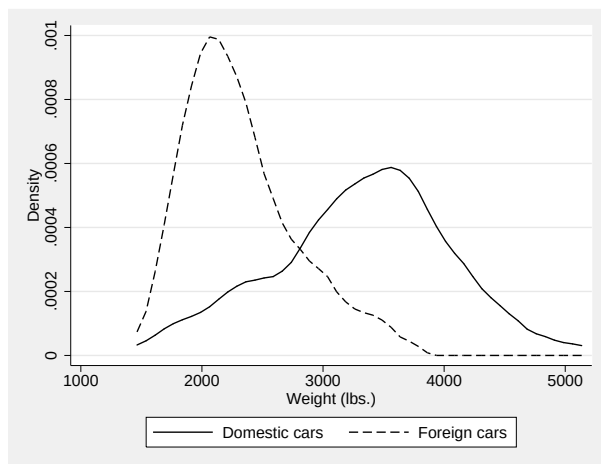
. kdensity weight if foreign==0, nograph generate(fx0) at(x)

. kdensity weight if foreign==1, nograph generate(fx1) at(x)

. label var fx0 "Domestic cars"

. label var fx1 "Foreign cars"

. line fx0 fx1 x, sort ytitle(Density)
```



◀

### □ Technical note

Although all the examples we included had densities of less than 1, the density may exceed 1.

The probability density  $f(x)$  of a continuous variable,  $x$ , has the units and dimensions of the reciprocal of  $x$ . If  $x$  is measured in meters,  $f(x)$  has units 1/meter. Thus the density is not measured on a probability scale, so it is possible for  $f(x)$  to exceed 1.

To see this, think of a uniform density on the interval 0 to 1. The area under the density curve is 1: this is the product of the density, which is constant at 1, and the range, which is 1. If the variable is then transformed by doubling, the area under the curve remains 1 and is the product of the density, constant at 0.5, and the range, which is 2. Conversely, if the variable is transformed by halving, the area under the curve also remains at 1 and is the product of the density, constant at 2, and the range, which is 0.5. (Strictly, the range is measured in certain units, and the density is measured in the reciprocal of those units, so the units cancel on multiplication.)

□

## Saved results

`kdensity` saves the following in `r()`:

|                        |                                                      |
|------------------------|------------------------------------------------------|
| Scalars                |                                                      |
| <code>r(bwidth)</code> | kernel bandwidth                                     |
| <code>r(n)</code>      | number of points at which the estimate was evaluated |
| <code>r(scale)</code>  | density bin width                                    |
| Macros                 |                                                      |
| <code>r(kernel)</code> | name of kernel                                       |

## Methods and formulas

`kdensity` is implemented as an ado-file.

A kernel density estimate is formed by summing the weighted values calculated with the kernel function  $K$ , as in

$$\hat{f}_K = \frac{1}{qh} \sum_{i=1}^n w_i K\left(\frac{x - X_i}{h}\right)$$

where  $q = \sum_i w_i$  if weights are frequency weights (`fweight`) or analytic weights (`aweight`), and  $q = 1$  if weights are importance weights (`iweights`). Analytic weights are rescaled so that  $\sum_i w_i = n$  (see [\[U\] 11 Language syntax](#)). If weights are not used, then  $w_i = 1$ , for  $i = 1, \dots, n$ . `kdensity` includes seven different kernel functions. The Epanechnikov is the default function if no other kernel is specified and is the most efficient in minimizing the mean integrated squared error.

| Kernel       | Formula                                                                               |                                                         |
|--------------|---------------------------------------------------------------------------------------|---------------------------------------------------------|
| Biweight     | $K[z] = \begin{cases} \frac{15}{16}(1 - z^2)^2 \\ 0 \end{cases}$                      | if $ z  < 1$<br>otherwise                               |
| Cosine       | $K[z] = \begin{cases} 1 + \cos(2\pi z) \\ 0 \end{cases}$                              | if $ z  < 1/2$<br>otherwise                             |
| Epanechnikov | $K[z] = \begin{cases} \frac{3}{4}(1 - \frac{1}{5}z^2)/\sqrt{5} \\ 0 \end{cases}$      | if $ z  < \sqrt{5}$<br>otherwise                        |
| Epan2        | $K[z] = \begin{cases} \frac{3}{4}(1 - z^2) \\ 0 \end{cases}$                          | if $ z  < 1$<br>otherwise                               |
| Gaussian     | $K[z] = \frac{1}{\sqrt{2\pi}}e^{-z^2/2}$                                              |                                                         |
| Parzen       | $K[z] = \begin{cases} \frac{4}{3} - 8z^2 + 8 z ^3 \\ 8(1 -  z )^3/3 \\ 0 \end{cases}$ | if $ z  \leq 1/2$<br>if $1/2 <  z  \leq 1$<br>otherwise |
| Rectangular  | $K[z] = \begin{cases} 1/2 \\ 0 \end{cases}$                                           | if $ z  < 1$<br>otherwise                               |
| Triangular   | $K[z] = \begin{cases} 1 -  z  \\ 0 \end{cases}$                                       | if $ z  < 1$<br>otherwise                               |

From the definitions given in the table, we can see that the choice of  $h$  will drive how many values are included in estimating the density at each point. This value is called the *window width* or *bandwidth*. If the window width is not specified, it is determined as

$$m = \min \left( \sqrt{\text{variance}_x}, \frac{\text{interquartile range}_x}{1.349} \right)$$

$$h = \frac{0.9m}{n^{1/5}}$$

where  $x$  is the variable for which we wish to estimate the kernel and  $n$  is the number of observations.

Most researchers agree that the choice of kernel is not as important as the choice of bandwidth. There is a great deal of literature on choosing bandwidths under various conditions; see, for example, [Parzen \(1962\)](#) or [Tapia and Thompson \(1978\)](#). Also see [Newton \(1988\)](#) for a comparison with sample spectral density estimation in time-series applications.

## Acknowledgments

We gratefully acknowledge the previous work by Isafías H. Salgado-Ugarte of Universidad Nacional Autónoma de México, and Makoto Shimizu and Toru Taniuchi of the University of Tokyo; see [Salgado-Ugarte, Shimizu, and Taniuchi \(1993\)](#). Their article provides a good overview of the subject of univariate kernel density estimation and presents arguments for its use in exploratory data analysis.

## References

- Cox, N. J. 2005. [Speaking Stata: Density probability plots](#). *Stata Journal* 5: 259–273.
- . 2007. Kernel estimation as a basic tool for geomorphological data analysis. *Earth Surface Processes and Landforms* 32: 1902–1912.
- Fiorio, C. V. 2004. [Confidence intervals for kernel density estimation](#). *Stata Journal* 4: 168–179.
- Fox, J. 1990. Describing univariate distributions. In *Modern Methods of Data Analysis*, ed. J. Fox and J. S. Long, 58–125. Newbury Park, CA: Sage.
- Goeden, G. B. 1978. A monograph of the coral trout, *Plectropomus leopardus* (Lacépède). *Queensland Fisheries Services Research Bulletin* 1: 1–42.
- Kohler, U., and F. Kreuter. 2009. [Data Analysis Using Stata](#). 2nd ed. College Station, TX: Stata Press.
- Newton, H. J. 1988. *TIMESLAB: A Time Series Analysis Laboratory*. Belmont, CA: Wadsworth.
- Parzen, E. 1962. On estimation of a probability density function and mode. *Annals of Mathematical Statistics* 33: 1065–1076.
- Royston, P., and N. J. Cox. 2005. [A multivariable scatterplot smoother](#). *Stata Journal* 5: 405–412.
- Salgado-Ugarte, I. H., and M. A. Pérez-Hernández. 2003. [Exploring the use of variable bandwidth kernel density estimators](#). *Stata Journal* 3: 133–147.
- Salgado-Ugarte, I. H., M. Shimizu, and T. Taniuchi. 1993. [snp6: Exploring the shape of univariate data using kernel density estimators](#). *Stata Technical Bulletin* 16: 8–19. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 155–173. College Station, TX: Stata Press.
- . 1995a. [snp6.1: ASH, WARPing, and kernel density estimation for univariate data](#). *Stata Technical Bulletin* 26: 23–31. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 161–172. College Station, TX: Stata Press.
- . 1995b. [snp6.2: Practical rules for bandwidth selection in univariate density estimation](#). *Stata Technical Bulletin* 27: 5–19. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 172–190. College Station, TX: Stata Press.
- . 1997. [snp13: Nonparametric assessment of multimodality for univariate data](#). *Stata Technical Bulletin* 38: 27–35. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 232–243. College Station, TX: Stata Press.
- Scott, D. W. 1992. *Multivariate Density Estimation: Theory, Practice, and Visualization*. New York: Wiley.

- Silverman, B. W. 1992. *Density Estimation for Statistics and Data Analysis*. London: Chapman & Hall.
- Simonoff, J. S. 1996. *Smoothing Methods in Statistics*. New York: Springer.
- Steichen, T. J. 1998. [gr33: Violin plots](#). *Stata Technical Bulletin* 46: 13–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 57–65. College Station, TX: Stata Press.
- Tapia, R. A., and J. R. Thompson. 1978. *Nonparametric Probability Density Estimation*. Baltimore: Johns Hopkins University Press.
- Van Kerm, P. 2003. [Adaptive kernel density estimation](#). *Stata Journal* 3: 148–156.
- Wand, M. P., and M. C. Jones. 1995. *Kernel Smoothing*. London: Chapman & Hall.

## Also see

[\[R\] histogram](#) — Histograms for continuous and categorical variables

# Title

ksmirnov — Kolmogorov–Smirnov equality-of-distributions test

## Syntax

*One-sample Kolmogorov–Smirnov test*

```
ksmirnov varname = exp [if] [in]
```

*Two-sample Kolmogorov–Smirnov test*

```
ksmirnov varname [if] [in] , by(groupvar) [exact]
```

## Menu

### one-sample

Statistics > Nonparametric analysis > Tests of hypotheses > One-sample Kolmogorov-Smirnov test

### two-sample

Statistics > Nonparametric analysis > Tests of hypotheses > Two-sample Kolmogorov-Smirnov test

## Description

`ksmirnov` performs one- and two-sample Kolmogorov–Smirnov tests of the equality of distributions. In the first syntax, *varname* is the variable whose distribution is being tested, and *exp* must evaluate to the corresponding (theoretical) cumulative. In the second syntax, *groupvar* must take on two distinct values. The distribution of *varname* for the first value of *groupvar* is compared with that of the second value.

When testing for normality, please see [\[R\] sktest](#) and [\[R\] swilk](#).

## Options for two-sample test

### Main

`by(groupvar)` is required. It specifies a binary variable that identifies the two groups.

`exact` specifies that the exact *p*-value be computed. This may take a long time if  $n > 50$ .

## Remarks

### ► Example 1: Two-sample test

Say that we have data on `x` that resulted from two different experiments, labeled as `group==1` and `group==2`. Our data contain



```
. use http://www.stata-press.com/data/r11/ksxmpl
. list
```

|    | group | x  |
|----|-------|----|
| 1. | 2     | 2  |
| 2. | 1     | 0  |
| 3. | 2     | 3  |
| 4. | 1     | 4  |
| 5. | 1     | 5  |
| 6. | 2     | 8  |
| 7. | 2     | 10 |

We wish to use the two-sample Kolmogorov–Smirnov test to determine if there are any differences in the distribution of  $x$  for these two groups:

```
. ksmirnov x, by(group)
Two-sample Kolmogorov-Smirnov test for equality of distribution functions
```

| Smaller group | D       | P-value | Corrected |
|---------------|---------|---------|-----------|
| 1:            | 0.5000  | 0.424   |           |
| 2:            | -0.1667 | 0.909   |           |
| Combined K-S: | 0.5000  | 0.785   | 0.735     |

The first line tests the hypothesis that  $x$  for group 1 contains *smaller* values than for group 2. The largest difference between the distribution functions is 0.5. The approximate  $p$ -value for this is 0.424, which is not significant.

The second line tests the hypothesis that  $x$  for group 1 contains *larger* values than for group 2. The largest difference between the distribution functions in this direction is 0.1667. The approximate  $p$ -value for this small difference is 0.909.

Finally, the approximate  $p$ -value for the combined test is 0.785, corrected to 0.735. The  $p$ -values `ksmirnov` calculates are based on the asymptotic distributions derived by [Smirnov \(1933\)](#). These approximations are not good for small samples ( $n < 50$ ). They are too conservative—real  $p$ -values tend to be substantially smaller. We have also included a less conservative approximation for the nondirectional hypothesis based on an empirical continuity correction—the 0.735 reported in the third column.

That number, too, is only an approximation. An exact value can be calculated using the `exact` option:

```
. ksmirnov x, by(group) exact
Two-sample Kolmogorov-Smirnov test for equality of distribution functions
```

| Smaller group | D       | P-value | Exact |
|---------------|---------|---------|-------|
| 1:            | 0.5000  | 0.424   |       |
| 2:            | -0.1667 | 0.909   |       |
| Combined K-S: | 0.5000  | 0.785   | 0.657 |

◀

## ► Example 2: One-sample test

Let's now test whether  $x$  in the example above is distributed normally. Kolmogorov–Smirnov is not a particularly powerful test in testing for normality, and we do not endorse such use of it; see [\[R\] sktest](#) and [\[R\] swilk](#) for better tests.

In any case, we will test against a normal distribution with the same mean and standard deviation:

```
. summarize x
+-----+-----+-----+-----+-----+
| Variable | Obs | Mean | Std. Dev. | Min | Max |
+-----+-----+-----+-----+-----+
| x | 7 | 4.571429 | 3.457222 | 0 | 10 |
+-----+-----+-----+-----+-----+
. ksmirnov x = normal((x-4.571429)/3.457222)
One-sample Kolmogorov-Smirnov test against theoretical distribution
normal((x-4.571429)/3.457222)
+-----+-----+-----+
| Smaller group | D | P-value | Corrected |
+-----+-----+-----+
x:	0.1650	0.683	
Cumulative:	-0.1250	0.803	
Combined K-S:	0.1650	0.991	0.978
+-----+-----+-----+
```

Because Stata has no way of knowing that we based this calculation on the calculated mean and standard deviation of `x`, the test statistics will be slightly conservative in addition to being approximations. Nevertheless, they clearly indicate that the data cannot be distinguished from normally distributed data.



## Saved results

`ksmirnov` saves the following in `r()`:

|                        |                                   |                         |                                          |
|------------------------|-----------------------------------|-------------------------|------------------------------------------|
| Scalars                |                                   |                         |                                          |
| <code>r(D_1)</code>    | <code>D</code> from line 1        | <code>r(D)</code>       | combined <code>D</code>                  |
| <code>r(p_1)</code>    | <code>p</code> -value from line 1 | <code>r(p)</code>       | combined <code>p</code> -value           |
| <code>r(D_2)</code>    | <code>D</code> from line 2        | <code>r(p_cor)</code>   | corrected combined <code>p</code> -value |
| <code>r(p_2)</code>    | <code>p</code> -value from line 2 | <code>r(p_exact)</code> | exact combined <code>p</code> -value     |
| Macros                 |                                   |                         |                                          |
| <code>r(group1)</code> | name of group from line 1         | <code>r(group2)</code>  | name of group from line 2                |

## Methods and formulas

`ksmirnov` is implemented as an ado-file.

In general, the Kolmogorov–Smirnov test (Kolmogorov 1933; Smirnov 1933; also see Conover [1999], 428–465) is not very powerful against differences in the tails of distributions. In return for this, it is fairly powerful for alternative hypotheses that involve lumpiness or clustering in the data.

The directional hypotheses are evaluated with the statistics

$$D^+ = \max_x \left\{ F(x) - G(x) \right\}$$
$$D^- = \min_x \left\{ F(x) - G(x) \right\}$$

where  $F(x)$  and  $G(x)$  are the empirical distribution functions for the sample being compared. The combined statistic is

$$D = \max \left( |D^+|, |D^-| \right)$$

The  $p$ -value for this statistic may be obtained by evaluating the asymptotic limiting distribution. Let  $m$  be the sample size for the first sample, and let  $n$  be the sample size for the second sample. [Smirnov \(1933\)](#) shows that

$$\lim_{m,n \rightarrow \infty} \Pr \left\{ \sqrt{mn/(m+n)} D_{m,n} \leq z \right\} = 1 - 2 \sum_{i=1}^{\infty} (-1)^{i-1} \exp(-2i^2 z^2)$$

The first five terms form the approximation  $P_a$  used by Stata. The exact  $p$ -value is calculated by a counting algorithm; see [Gibbons \(1971, 127–131\)](#). A corrected  $p$ -value was obtained by modifying the asymptotic  $p$ -value by using a numerical approximation technique:

$$Z = \Phi^{-1}(P_a) + 1.04/\min(m, n) + 2.09/\max(m, n) - 1.35/\sqrt{mn/(m+n)}$$

$$p\text{-value} = \Phi(Z)$$

where  $\Phi(\cdot)$  is the cumulative normal distribution.

Andrei Nikolayevich Kolmogorov (1903–1987), of Russia, was one of the great mathematicians of the twentieth century, making outstanding contributions in many different branches, including set theory, measure theory, probability and statistics, approximation theory, functional analysis, classical dynamics, and theory of turbulence. He was a faculty member at Moscow State University for more than 60 years.

Nikolai Vasilyevich Smirnov (1900–1966) was a Russian statistician whose work included contributions in nonparametric statistics, order statistics, and goodness of fit. After army service and the study of philosophy and philology, he turned to mathematics and eventually rose to be head of mathematical statistics at the Steklov Mathematical Institute in Moscow.

## References

- Aivazian, S. A. 1997. Smirnov, Nikolai Vasil'yevich. In *Leading Personalities in Statistical Sciences: From the Seventeenth Century to the Present*, ed. N. L. Johnson and S. Kotz, 208–210. New York: Wiley.
- Conover, W. J. 1999. *Practical Nonparametric Statistics*. 3rd ed. New York: Wiley.
- Gibbons, J. D. 1971. *Nonparametric Statistical Inference*. New York: McGraw–Hill.
- Jann, B. 2008. [Multinomial goodness-of-fit: Large-sample tests with survey design correction and exact tests for small samples](#). *Stata Journal* 8: 147–169.
- Johnson, N. L., and S. Kotz. 1997. Kolmogorov, Andrei Nikolayevich. In *Leading Personalities in Statistical Sciences: From the Seventeenth Century to the Present*, ed. N. L. Johnson and S. Kotz, 255–256. New York: Wiley.
- Kolmogorov, A. N. 1933. Sulla determinazione empirica di una legge di distribuzione. *Giornale dell' Istituto Italiano degli Attuari* 4: 83–91.
- Riffenburgh, R. H. 2005. *Statistics in Medicine*. 2nd ed. New York: Elsevier.
- Smirnov, N. V. 1933. Estimate of deviation between empirical distribution functions in two independent samples. *Bulletin Moscow University* 2: 3–16.

## Also see

- [R] [rntest](#) — Test for random order
- [R] [sktest](#) — Skewness and kurtosis test for normality
- [R] [swilk](#) — Shapiro–Wilk and Shapiro–Francia tests for normality

# Title

**kwallis** — Kruskal–Wallis equality-of-populations rank test

## Syntax

```
kwallis varname [if] [in] , by(groupvar)
```

## Menu

Statistics > Nonparametric analysis > Tests of hypotheses > Kruskal-Wallis rank test

## Description

**kwallis** tests the hypothesis that several samples are from the same population. In the syntax diagram above, *varname* refers to the variable recording the outcome, and *groupvar* refers to the variable denoting the population. `by()` is required.

## Option

`by(groupvar)` is required. It specifies a variable that identifies the groups.

## Remarks

### ➤ Example 1

We have data on the 50 states. The data contain the median age of the population, **medage**, and the region of the country, **region**, for each state. We wish to test for the equality of the median age distribution across all four regions simultaneously:

```
. use http://www.stata-press.com/data/r11/census
(1980 Census data by state)
. kwallis medage, by(region)
Kruskal-Wallis equality-of-populations rank test
```

| region  | Obs | Rank Sum |
|---------|-----|----------|
| NE      | 9   | 376.50   |
| N Cntrl | 12  | 294.00   |
| South   | 16  | 398.00   |
| West    | 13  | 206.50   |

```
chi-squared = 17.041 with 3 d.f.
probability = 0.0007
chi-squared with ties = 17.062 with 3 d.f.
probability = 0.0007
```

From the output, we see that we can reject the hypothesis that the populations are the same at any level below 0.07%.



## Saved results

kwallis saves the following in `r()`:

Scalars

|                    |                    |                          |                            |
|--------------------|--------------------|--------------------------|----------------------------|
| <code>r(df)</code> | degrees of freedom | <code>r(chi2)</code>     | $\chi^2$                   |
|                    |                    | <code>r(chi2_adj)</code> | $\chi^2$ adjusted for ties |

## Methods and formulas

kwallis is implemented as an ado-file.

The Kruskal–Wallis test (Kruskal and Wallis 1952, 1953; also see Altman [1991, 213–215]; Conover [1999, 288–297]; and Riffenburgh [2005, 287–291]) is a multiple-sample generalization of the two-sample Wilcoxon (also called Mann–Whitney) rank sum test (Wilcoxon 1945; Mann and Whitney 1947). Samples of sizes  $n_j$ ,  $j = 1, \dots, m$ , are combined and ranked in ascending order of magnitude. Tied values are assigned the average ranks. Let  $n$  denote the overall sample size, and let  $R_j = \sum_{i=1}^{n_j} R(X_{ji})$  denote the sum of the ranks for the  $j$ th sample. The Kruskal–Wallis one-way analysis-of-variance test,  $H$ , is defined as

$$H = \frac{1}{S^2} \left\{ \sum_{j=1}^m \frac{R_j^2}{n_j} - \frac{n(n+1)^2}{4} \right\}$$

where

$$S^2 = \frac{1}{n-1} \left\{ \sum_{\text{all ranks}} R(X_{ji})^2 - \frac{n(n+1)^2}{4} \right\}$$

If there are no ties, this equation simplifies to

$$H = \frac{12}{n(n+1)} \sum_{j=1}^m \frac{R_j^2}{n_j} - 3(n+1)$$

The sampling distribution of  $H$  is approximately  $\chi^2$  with  $m-1$  degrees of freedom.

William Henry Kruskal (1919–2005) was born in New York City. He studied mathematics and statistics at Antioch College, Harvard, and Columbia, and joined the University of Chicago in 1951. He made many outstanding contributions to linear models, nonparametric statistics, government statistics, and the history and methodology of statistics.

Wilson Allen Wallis (1912–1998) was born in Philadelphia. He studied psychology and economics at the Universities of Minnesota and Chicago and at Columbia. He taught at Yale, Stanford, and Chicago, before moving as President (later Chancellor) to the University of Rochester in 1962. He also served in several Republican administrations. Wallis served as editor of the *Journal of the American Statistical Association*, coauthored a popular introduction to statistics, and contributed to nonparametric statistics.

## References

- Altman, D. G. 1991. *Practical Statistics for Medical Research*. London: Chapman & Hall/CRC.
- Conover, W. J. 1999. *Practical Nonparametric Statistics*. 3rd ed. New York: Wiley.
- Fienberg, S. E., S. M. Stigler, and J. M. Tanur. 2007. The William Kruskal Legacy: 1919–2005. *Statistical Science* 22: 255–261.
- Kruskal, W. H., and W. A. Wallis. 1952. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association* 47: 583–621.
- . 1953. Errata: Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association* 48: 907–911.
- Mann, H. B., and D. R. Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics* 18: 50–60.
- Newson, R. 2006. [Confidence intervals for rank statistics: Somers' D and extensions](#). *Stata Journal* 6: 309–334.
- Olkin, I. 1991. A conversation with W. Allen Wallis. *Statistical Science* 6: 121–140.
- Riffenburgh, R. H. 2005. *Statistics in Medicine*. 2nd ed. New York: Elsevier.
- Wilcoxon, F. 1945. Individual comparisons by ranking methods. *Biometrics* 1: 80–83.
- Zabell, S. 1994. A conversation with William Kruskal. *Statistical Science* 9: 285–303.

## Also see

- [\[R\] \*\*nptrend\*\*](#) — Test for trend across ordered groups
- [\[R\] \*\*oneway\*\*](#) — One-way analysis of variance
- [\[R\] \*\*sdtest\*\*](#) — Variance-comparison tests
- [\[R\] \*\*signrank\*\*](#) — Equality tests on matched data

# Title

**ladder** — Ladder of powers

## Syntax

*Ladder of powers*

```
ladder varname [if] [in] [, generate(newvar) noadjust]
```

*Ladder-of-powers histograms*

```
gladder varname [if] [in] [, histogram_options combine_options]
```

*Ladder-of-powers quantile-normal plots*

```
qladder varname [if] [in] [, qnorm_options combine_options]
```

*by* is allowed with `ladder`; see [\[D\]](#) **by**.

## Menu

### **ladder**

Statistics > Summaries, tables, and tests > Distributional plots and tests > Ladder of powers

### **gladder**

Statistics > Summaries, tables, and tests > Distributional plots and tests > Ladder-of-powers histograms

### **qladder**

Statistics > Summaries, tables, and tests > Distributional plots and tests > Ladder-of-powers quantile-normal plots

## Description

`ladder` searches a subset of the ladder of powers ([Tukey 1977](#)) for a transform that converts *varname* into a normally distributed variable. `sktest` tests for normality; see [\[R\]](#) **sktest**. Also see [\[R\]](#) **boxcox**.

`gladder` displays nine histograms of transforms of *varname* according to the ladder of powers. `gladder` is useful pedagogically, but we do not advise looking at histograms for research work; `ladder` or `qnorm` (see [\[R\]](#) **diagnostic plots**) is preferred.

`qladder` displays the quantiles of transforms of *varname* according to the ladder of powers against the quantiles of a normal distribution.

## Options for ladder

Main

`generate(newvar)` saves the transformed values corresponding to the minimum chi-squared value from the table. We do not recommend using `generate()` because it is literal in interpreting the minimum, thus ignoring nearly equal but perhaps more interpretable transforms.

`noadjust` is the `noadjust` option to `sktest`; see [R] [sktest](#).

## Options for gladder

*histogram\_options* affect the rendition of the histograms across all relevant transformations; see [R] [histogram](#). Here the `normal` option is assumed, so you must supply the `nonnormal` option to suppress the overlaid normal density. Also, `gladder` does not allow the `width(#)` option of `histogram`.

*combine\_options* are any of the options documented in [G] [graph combine](#). These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

## Options for qladder

*qnorm\_options* affect the rendition of the quantile–normal plots across all relevant transformations. See [R] [diagnostic plots](#).

*combine\_options* are any of the options documented in [G] [graph combine](#). These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

## Remarks

### ► Example 1: ladder

We have data on the mileage rating of 74 automobiles and wish to find a transform that makes the variable normally distributed:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. ladder mpg
```

| Transformation  | formula        | chi2(2) | P(chi2) |
|-----------------|----------------|---------|---------|
| cubic           | $mpg^3$        | 43.59   | 0.000   |
| square          | $mpg^2$        | 27.03   | 0.000   |
| identity        | $mpg$          | 10.95   | 0.004   |
| square root     | $\sqrt{mpg}$   | 4.94    | 0.084   |
| log             | $\log(mpg)$    | 0.87    | 0.647   |
| 1/(square root) | $1/\sqrt{mpg}$ | 0.20    | 0.905   |
| inverse         | $1/mpg$        | 2.36    | 0.307   |
| 1/square        | $1/(mpg^2)$    | 11.99   | 0.002   |
| 1/cubic         | $1/(mpg^3)$    | 24.30   | 0.000   |

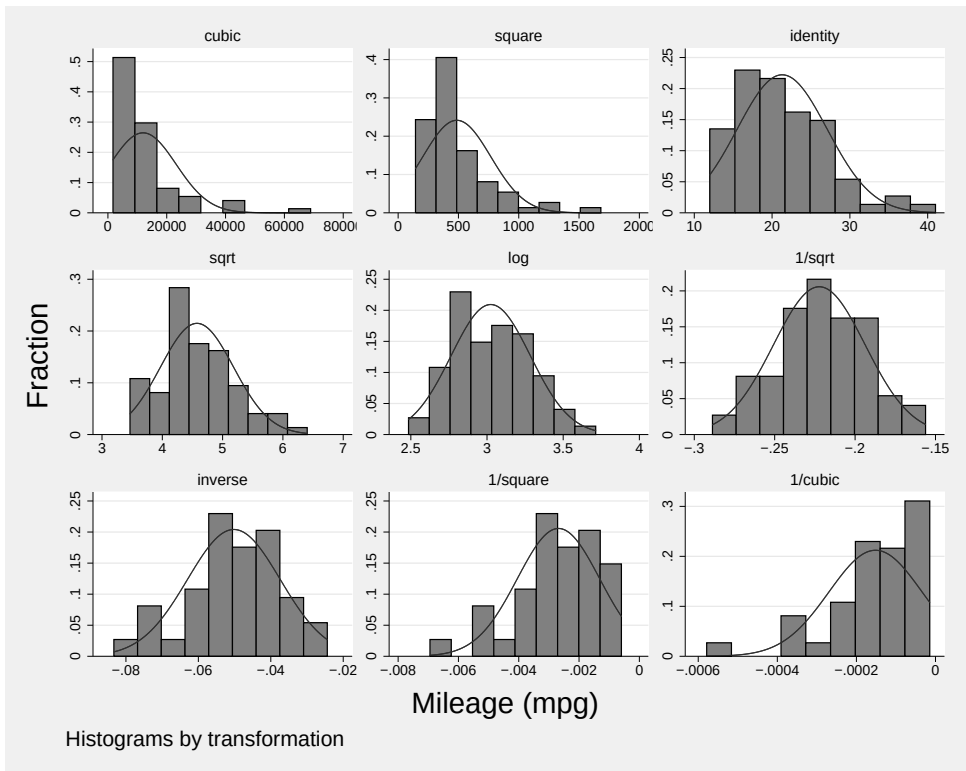
If we had typed `ladder mpg, gen(mpgx)`, the variable `mpgx` containing  $1/\sqrt{mpg}$  would have been automatically generated for us. This is the perfect example of why you should not, in general, specify the `generate()` option. We also cannot reject the hypothesis that the inverse of `mpg` is normally distributed and that  $1/mpg$ —gallons per mile—has a better interpretation. It is a measure of energy consumption.



## ► Example 2: gladder

`gladder` explores the same transforms as `ladder` but presents results graphically:

```
. gladder mpg, fraction
```



◀

## □ Technical note

`gladder` is useful pedagogically, but be careful when using it for research work, especially with many observations. For instance, consider the following data on the average July temperature in degrees Fahrenheit for 954 U.S. cities:

```
. use http://www.stata-press.com/data/r11/citytemp
(City Temperature Data)
```

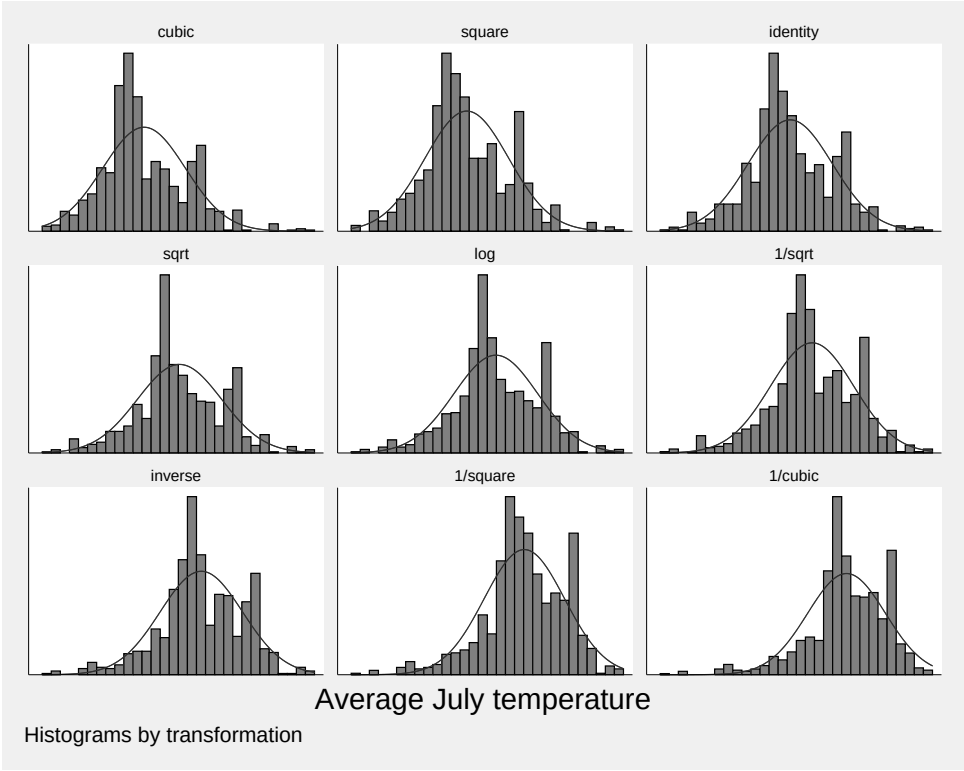
```
. ladder tempjuly
```

| Transformation  | formula          | chi2(2) | P(chi2) |
|-----------------|------------------|---------|---------|
| cubic           | tempjuly^3       | 47.49   | 0.000   |
| square          | tempjuly^2       | 19.70   | 0.000   |
| identity        | tempjuly         | 3.83    | 0.147   |
| square root     | sqrt(tempjuly)   | 1.83    | 0.400   |
| log             | log(tempjuly)    | 5.40    | 0.067   |
| 1/(square root) | 1/sqrt(tempjuly) | 13.72   | 0.001   |
| inverse         | 1/tempjuly       | 26.36   | 0.000   |
| 1/square        | 1/(tempjuly^2)   | 64.43   | 0.000   |
| 1/cubic         | 1/(tempjuly^3)   | .       | 0.000   |

The period in the last line indicates that the  $\chi^2$  is very large; see [R] `sktest`.

From the table, we see that there is certainly a difference in normality between the square and square-root transform. If, however, you can see the difference between the transforms in the diagram below, you have better eyes than we do:

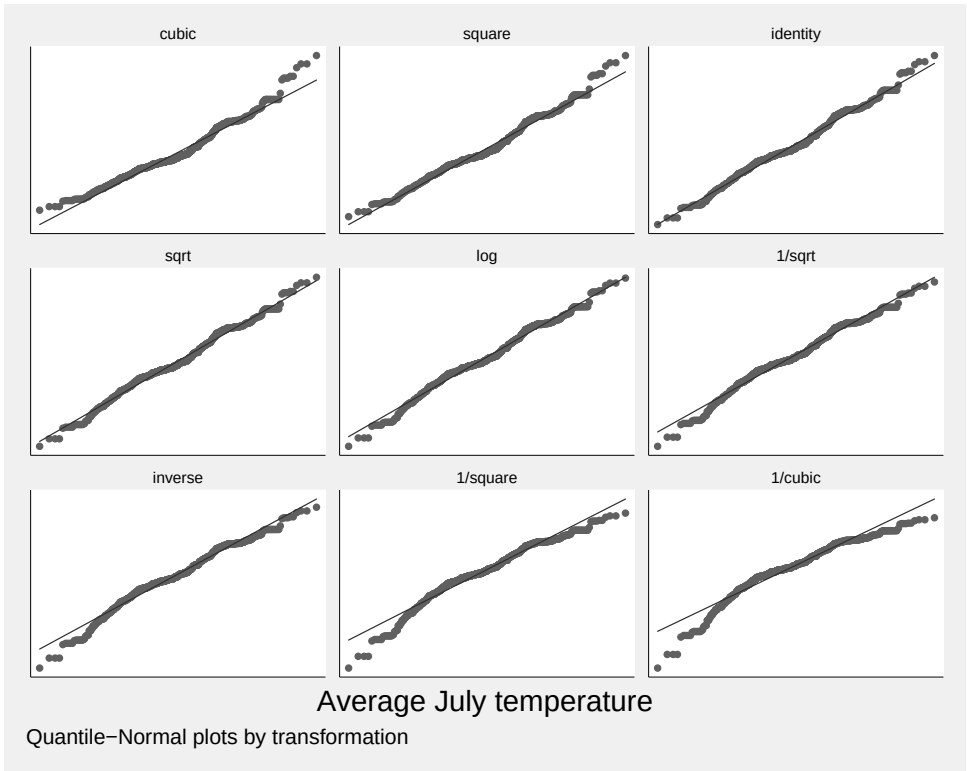
```
. gladder tempjuly, l1title("") ylabel(none) xlabel(none)
```



➤ Example 3: qladder

A better graph for seeing normality is the quantile–normal graph, which can be produced by `qladder`.

```
. qladder tempjuly, ylabel(none) xlabel(none)
```



This graph shows that for the square transform, the upper tail—and only the upper tail—diverges from what would be expected. This divergence is detected by `sktest` as a problem with skewness, as we would learn from using `sktest` to examine `tempjuly` squared and square rooted.

◀

*(Continued on next page)*

## Saved results

`ladder` saves the following in `r()`:

### Scalars

|                           |                                                      |
|---------------------------|------------------------------------------------------|
| <code>r(N)</code>         | number of observations                               |
| <code>r(invcube)</code>   | $\chi^2$ for inverse-cubic transformation            |
| <code>r(P_invcube)</code> | significance level for inverse-cubic transformation  |
| <code>r(invsq)</code>     | $\chi^2$ for inverse-square transformation           |
| <code>r(P_invsq)</code>   | significance level for inverse-square transformation |
| <code>r(inv)</code>       | $\chi^2$ for inverse transformation                  |
| <code>r(P_inv)</code>     | significance level for inverse transformation        |
| <code>r(invsqrt)</code>   | $\chi^2$ for inverse-root transformation             |
| <code>r(P_invsqrt)</code> | significance level for inverse-root transformation   |
| <code>r(log)</code>       | $\chi^2$ for log transformation                      |
| <code>r(P_log)</code>     | significance level for log transformation            |
| <code>r(sqrt)</code>      | $\chi^2$ for square-root transformation              |
| <code>r(P_sqrt)</code>    | significance level for square-root transformation    |
| <code>r(ident)</code>     | $\chi^2$ for untransformed data                      |
| <code>r(P_ident)</code>   | significance level for untransformed data            |
| <code>r(square)</code>    | $\chi^2$ for square transformation                   |
| <code>r(P_square)</code>  | significance level for square transformation         |
| <code>r(cube)</code>      | $\chi^2$ for cubic transformation                    |
| <code>r(P_cube)</code>    | significance level for cubic transformation          |

## Methods and formulas

`ladder`, `gladder`, and `qladder` are implemented as ado-files.

For `ladder`, results are as reported by `sktest`; see [R] [sktest](#). If `generate()` is specified, the transform with the minimum  $\chi^2$  value is chosen.

`gladder` sets the number of bins to  $\min(\sqrt{n}, 10 \log_{10} n)$ , rounded to the closest integer, where  $n$  is the number of unique values of *varname*. See [R] [histogram](#) for a discussion of the optimal number of bins.

Also see [Findley \(1990\)](#) for a ladder-of-powers variable transformation program that produces one-way graphs with overlaid box plots, in addition to histograms with overlaid normals. [Buchner and Findley \(1990\)](#) discuss ladder-of-powers transformations as one aspect of preliminary data analysis. Also see [Hamilton \(1992, 18–23\)](#) and [Hamilton \(2009, 142–144\)](#).

## Acknowledgment

`qladder` was written by Jeroen Weesie, Utrecht University, Utrecht, The Netherlands.

## References

- Buchner, D. M., and T. W. Findley. 1990. Research in physical medicine and rehabilitation: VIII. Preliminary data analysis. *American Journal of Physical Medicine and Rehabilitation* 69: 154–169.
- Cox, N. J. 2005. [Speaking Stata: Density probability plots](#). *Stata Journal* 5: 259–273.

- Findley, T. W. 1990. [sed3: Variable transformation and evaluation](#). *Stata Technical Bulletin* 2: 15. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 85–86. College Station, TX: Stata Press.
- Hamilton, L. C. 1992. *Regression with Graphics: A Second Course in Applied Statistics*. Belmont, CA: Duxbury.
- . 2009. *Statistics with Stata (Updated for Version 10)*. Belmont, CA: Brooks/Cole.
- Tukey, J. W. 1977. *Exploratory Data Analysis*. Reading, MA: Addison–Wesley.

## Also see

- [R] [diagnostic plots](#) — Distributional diagnostic plots
- [R] [lnskew0](#) — Find zero-skewness log or Box–Cox transform
- [R] [lv](#) — Letter-value displays
- [R] [sktest](#) — Skewness and kurtosis test for normality

# Title

level — Set default confidence level

## Syntax

```
set level # [, permanently]
```

## Description

set level specifies the default confidence level for confidence intervals for all commands that report confidence intervals. The initial value is 95, meaning 95% confidence intervals. # may be between 10.00 and 99.99, and # can have at most two digits after the decimal point.

## Option

permanently specifies that, in addition to making the change right now, the level setting be remembered and become the default setting when you invoke Stata.

## Remarks

To change the level of confidence intervals reported by a particular command, you need not reset the default confidence level. All commands that report confidence intervals have a level(#) option. When you do not specify the option, the confidence intervals are calculated for the default level set by set level, or for 95% if you have not reset set level.

### ► Example 1

We use the ci command to obtain the confidence interval for the mean of mpg:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. ci mpg
```

| Variable | Obs | Mean    | Std. Err. | [95% Conf. Interval] |          |
|----------|-----|---------|-----------|----------------------|----------|
| mpg      | 74  | 21.2973 | .6725511  | 19.9569              | 22.63769 |

To obtain 90% confidence intervals, we would type

```
. ci mpg, level(90)
```

| Variable | Obs | Mean    | Std. Err. | [90% Conf. Interval] |          |
|----------|-----|---------|-----------|----------------------|----------|
| mpg      | 74  | 21.2973 | .6725511  | 20.17683             | 22.41776 |

or

```
. set level 90
. ci mpg
```

| Variable | Obs | Mean    | Std. Err. | [90% Conf. Interval] |          |
|----------|-----|---------|-----------|----------------------|----------|
| mpg      | 74  | 21.2973 | .6725511  | 20.17683             | 22.41776 |

If we opt for the second alternative, the next time that we fit a model (say, with `regress`), 90% confidence intervals will be reported. If we wanted 95% confidence intervals, we could specify `level(95)` on the estimation command, or we could reset the default by typing `set level 95`.

The current setting of `level()` is stored as the c-class value `c(level)`; see [P] [creturn](#).



## Also see

[R] [query](#) — Display system parameters

[P] [creturn](#) — Return c-class values

[U] [20 Estimation and postestimation commands](#)

[U] [20.7 Specifying the width of confidence intervals](#)

Syntax

```
lincom exp [, options]
```

| options          | description                                             |
|------------------|---------------------------------------------------------|
| <u>e</u> form    | generic label; <code>exp(b)</code> ; the default        |
| <u>o</u> r       | odds ratio                                              |
| <u>h</u> r       | hazard ratio                                            |
| <u>s</u> hr      | subhazard ratio                                         |
| <u>i</u> rr      | incidence-rate ratio                                    |
| <u>r</u> rr      | relative-risk ratio                                     |
| <u>l</u> evel(#) | set confidence level; default is <code>level(95)</code> |

where *exp* is any linear combination of coefficients that is a valid syntax for `test`; see [\[R\]](#) `test`. *exp* must not contain an equal sign.

Menu

Statistics > Postestimation > Linear combinations of estimates

Description

`lincom` computes point estimates, standard errors, *t* or *z* statistics, *p*-values, and confidence intervals for linear combinations of coefficients after any estimation command. Results can optionally be displayed as odds ratios, hazard ratios, incidence-rate ratios, or relative-risk ratios.

`lincom` can be used with `svy` estimation results; see [\[SVY\]](#) `svy postestimation`.

Options

`eform`, `or`, `hr`, `shr`, `irr`, and `rrr` all report coefficient estimates as  $\exp(\hat{\beta})$  rather than  $\hat{\beta}$ . Standard errors and confidence intervals are similarly transformed. `or` is the default after `logistic`. The only difference in these options is how the output is labeled.

| Option             | Label                   | Explanation          | Example commands                           |
|--------------------|-------------------------|----------------------|--------------------------------------------|
| <code>eform</code> | <code>exp(b)</code>     | Generic label        |                                            |
| <code>or</code>    | <code>Odds Ratio</code> | Odds ratio           | <code>logistic</code> , <code>logit</code> |
| <code>hr</code>    | <code>Haz. Ratio</code> | Hazard ratio         | <code>stcox</code> , <code>streg</code>    |
| <code>shr</code>   | <code>SHR</code>        | Subhazard ratio      | <code>stcrreg</code>                       |
| <code>irr</code>   | <code>IRR</code>        | Incidence-rate ratio | <code>poisson</code>                       |
| <code>rrr</code>   | <code>RRR</code>        | Relative-risk ratio  | <code>mlogit</code>                        |



*exp* may not contain any additive constants when you use the *eform*, *or*, *hr*, *irr*, or *rrr* option. *level*(#) specifies the confidence level, as a percentage, for confidence intervals. The default is *level*(95) or as set by *set level*; see [U] 20.7 Specifying the width of confidence intervals.

## Remarks

Remarks are presented under the following headings:

*Using lincom*

*Odds ratios and incidence-rate ratios*

*Multiple-equation models*

## Using lincom

After fitting a model and obtaining estimates for coefficients  $\beta_1, \beta_2, \dots, \beta_k$ , you may want to view estimates for linear combinations of the  $\beta_i$ , such as  $\beta_1 - \beta_2$ . *lincom* can display estimates for any linear combination of the form  $c_0 + c_1\beta_1 + c_2\beta_2 + \dots + c_k\beta_k$ .

*lincom* works after any estimation command for which *test* works. Any valid expression for *test* syntax 1 (see [R] *test*) is a valid expression for *lincom*.

*lincom* is useful for viewing odds ratios, hazard ratios, etc., for one group (i.e., one set of covariates) relative to another group (i.e., another set of covariates). See the examples below.

### ► Example 1

We perform a linear regression:

```
. use http://www.stata.press.com/data/r11/regress
. regress y x1 x2 x3
```

| Source   | SS         | df  | MS         | Number of obs = 148    |  |  |
|----------|------------|-----|------------|------------------------|--|--|
| Model    | 3259.3561  | 3   | 1086.45203 | F( 3, 144) = 96.12     |  |  |
| Residual | 1627.56282 | 144 | 11.3025196 | Prob > F = 0.0000      |  |  |
|          |            |     |            | R-squared = 0.6670     |  |  |
|          |            |     |            | Adj R-squared = 0.6600 |  |  |
| Total    | 4886.91892 | 147 | 33.2443464 | Root MSE = 3.3619      |  |  |

| y     | Coef.    | Std. Err. | t      | P> t  | [95% Conf. Interval] |           |
|-------|----------|-----------|--------|-------|----------------------|-----------|
| x1    | 1.457113 | 1.07461   | 1.36   | 0.177 | -.666934             | 3.581161  |
| x2    | 2.221682 | .8610358  | 2.58   | 0.011 | .5197797             | 3.923583  |
| x3    | -.006139 | .0005543  | -11.08 | 0.000 | -.0072345            | -.0050435 |
| _cons | 36.10135 | 4.382693  | 8.24   | 0.000 | 27.43863             | 44.76407  |

To see the difference of the coefficients of *x2* and *x1*, we type

```
. lincom x2 - x1
(1) - x1 + x2 = 0
```

| y   | Coef.    | Std. Err. | t    | P> t  | [95% Conf. Interval] |          |
|-----|----------|-----------|------|-------|----------------------|----------|
| (1) | .7645682 | .9950282  | 0.77 | 0.444 | -1.20218             | 2.731316 |

The expression can be any linear combination.

```
. lincom 3*x1 + 500*x3
(1) 3 x1 + 500 x3 = 0
```

| y   | Coef.    | Std. Err. | t    | P> t  | [95% Conf. Interval] |          |
|-----|----------|-----------|------|-------|----------------------|----------|
| (1) | 1.301825 | 3.396624  | 0.38 | 0.702 | -5.411858            | 8.015507 |

Nonlinear expressions are not allowed.

```
. lincom x2/x1
not possible with test
r(131);
```

For information about estimating nonlinear expressions, see [\[R\] nlcom](#).

❑ Technical note

`lincom` uses the same shorthands for coefficients as does `test` (see [\[R\] test](#)). When you type `x1`, for instance, `lincom` knows that you mean the coefficient of `x1`. The formal syntax for referencing this coefficient is actually `_b[x1]`, or alternatively, `_coef[x1]`. So, more formally, in the last example we could have typed

```
. lincom 3*_b[x1] + 500*_b[x3]
(output omitted)
```

Odds ratios and incidence-rate ratios

After logistic regression, the `or` option can be specified with `lincom` to display odds ratios for any effect. Incidence-rate ratios after commands such as `poisson` can be similarly obtained by specifying the `irr` option.

➤ Example 2

Consider the low birthweight dataset from [Hosmer Jr. and Lemeshow \(2000, 25\)](#). We fit a logistic regression model of low birthweight (variable `low`) on the following variables:

| Variable   | Description                                                 | Coding                             |
|------------|-------------------------------------------------------------|------------------------------------|
| age        | age in years                                                |                                    |
| race       | race                                                        | 1 if white, 2 if black, 3 if other |
| smoke      | smoking status                                              | 1 if smoker, 0 if nonsmoker        |
| ht         | history of hypertension                                     | 1 if yes, 0 if no                  |
| ui         | uterine irritability                                        | 1 if yes, 0 if no                  |
| lwd        | maternal weight before pregnancy                            | 1 if weight < 110 lb., 0 otherwise |
| ptd        | history of premature labor                                  | 1 if yes, 0 if no                  |
| c.age##lwd | age main effects, lwd main effects, and their interaction   |                                    |
| smoke##lwd | smoke main effects, lwd main effects, and their interaction |                                    |

We first fit a model without the interaction terms by using `logit`.

```
. use http://www.stata-press.com/data/r11/lbw3
(Hosmer & Lemeshow data)
```

```
. logit low age lwd i.race smoke ptd ht ui
```

```
Iteration 0: log likelihood = -117.336
Iteration 1: log likelihood = -99.3982
Iteration 2: log likelihood = -98.780418
Iteration 3: log likelihood = -98.777998
Iteration 4: log likelihood = -98.777998
```

```
Logistic regression
```

```
Number of obs = 189
LR chi2(8) = 37.12
Prob > chi2 = 0.0000
Pseudo R2 = 0.1582
```

```
Log likelihood = -98.777998
```

| low   | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-------|-----------|-----------|-------|-------|----------------------|----------|
| age   | -.0464796 | .0373888  | -1.24 | 0.214 | -.1197603            | .0268011 |
| lwd   | .8420615  | .4055338  | 2.08  | 0.038 | .0472299             | 1.636893 |
| race  |           |           |       |       |                      |          |
| 2     | 1.073456  | .5150753  | 2.08  | 0.037 | .0639273             | 2.082985 |
| 3     | .815367   | .4452979  | 1.83  | 0.067 | -.0574008            | 1.688135 |
| smoke | .8071996  | .404446   | 2.00  | 0.046 | .0145001             | 1.599899 |
| ptd   | 1.281678  | .4621157  | 2.77  | 0.006 | .3759478             | 2.187408 |
| ht    | 1.435227  | .6482699  | 2.21  | 0.027 | .1646414             | 2.705813 |
| ui    | .6576256  | .4666192  | 1.41  | 0.159 | -.2569313            | 1.572182 |
| _cons | -1.216781 | .9556797  | -1.27 | 0.203 | -3.089878            | .656317  |

To get the odds ratio for black smokers relative to white nonsmokers (the reference group), we type

```
. lincom 2.race + smoke, or
```

```
(1) [low]2.race + [low]smoke = 0
```

| low | Odds Ratio | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|-----|------------|-----------|------|-------|----------------------|----------|
| (1) | 6.557805   | 4.744692  | 2.60 | 0.009 | 1.588176             | 27.07811 |

lincom computed  $\exp(\beta_{2.\text{race}} + \beta_{\text{smoke}}) = 6.56$ . To see the odds ratio for white smokers relative to black nonsmokers, we type

```
. lincom smoke - 2.race, or
```

```
(1) - [low]2.race + [low]smoke = 0
```

| low | Odds Ratio | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-----|------------|-----------|-------|-------|----------------------|----------|
| (1) | .7662425   | .4430176  | -0.46 | 0.645 | .2467334             | 2.379603 |

Now let's add the interaction terms to the model (Hosmer Jr. and Lemeshow 1989, table 4.10). This time, we will use `logistic` rather than `logit`. By default, `logistic` displays odds ratios.

```

. logistic low i.race ht ui ptd c.age##lwd smoke##lwd
Logistic regression
Number of obs = 189
LR chi2(10) = 42.66
Prob > chi2 = 0.0000
Pseudo R2 = 0.1818
Log likelihood = -96.00616

```

|           | low | Odds Ratio | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-----------|-----|------------|-----------|-------|-------|----------------------|----------|
| race      |     |            |           |       |       |                      |          |
|           | 2   | 2.95383    | 1.532789  | 2.09  | 0.037 | 1.068277             | 8.167465 |
|           | 3   | 2.137589   | .9919138  | 1.64  | 0.102 | .8608708             | 5.307752 |
| ht        |     | 3.893141   | 2.575201  | 2.05  | 0.040 | 1.064768             | 14.2346  |
| ui        |     | 2.071284   | .9931388  | 1.52  | 0.129 | .8092926             | 5.301192 |
| ptd       |     | 3.426633   | 1.615282  | 2.61  | 0.009 | 1.360252             | 8.632089 |
| age       |     | .9194513   | .041896   | -1.84 | 0.065 | .8408967             | 1.005344 |
| 1.lwd     |     | .1772934   | .3312384  | -0.93 | 0.354 | .0045539             | 6.902367 |
|           |     |            |           |       |       |                      |          |
| lwd#c.age |     |            |           |       |       |                      |          |
| 1         |     | 1.15883    | .09602    | 1.78  | 0.075 | .9851215             | 1.36317  |
| 1.smoke   |     | 3.168096   | 1.452378  | 2.52  | 0.012 | 1.289956             | 7.78076  |
|           |     |            |           |       |       |                      |          |
| smoke#lwd |     |            |           |       |       |                      |          |
| 1 1       |     | .2447849   | .2003996  | -1.72 | 0.086 | .0491956             | 1.217988 |

Hosmer Jr. and Lemeshow (1989, table 4.13) consider the effects of smoking (smoke = 1) and low maternal weight before pregnancy (lwd = 1). The effect of smoking among non–low-weight mothers (lwd = 0) is given by the odds ratio 3.17 for smoke in the logistic output. The effect of smoking among low-weight mothers is given by

```

. lincom 1.smoke + 1.smoke#1.lwd
(1) [low]1.smoke + [low]1.smoke#1.lwd = 0

```

|     | low | Odds Ratio | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-----|-----|------------|-----------|-------|-------|----------------------|----------|
| (1) |     | .7755022   | .574951   | -0.34 | 0.732 | .1813465             | 3.316323 |

We did not have to specify the or option. After logistic, lincom assumes or by default.

The effect of low weight (lwd = 1) is more complicated because we fit an age × lwd interaction. We must specify the age of mothers for the effect. The effect among 30-year-old nonsmokers is given by

```

. lincom 1.lwd + 30*1.lwd#c.age
(1) [low]1.lwd + 30*[low]1.lwd#c.age = 0

```

|     | low | Odds Ratio | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|-----|-----|------------|-----------|------|-------|----------------------|----------|
| (1) |     | 14.7669    | 13.5669   | 2.93 | 0.003 | 2.439264             | 89.39633 |

lincom computed  $\exp(\beta_{lwd} + 30\beta_{age1lwd}) = 14.8$ . It may seem odd that we entered it as 1.lwd + 30\*1.lwd#c.age, but remember that these terms are just lincom’s (and test’s) shorthands for `_b[1.lwd]` and `_b[1.lwd#c.age]`. We could have typed

```
. lincom _b[1.lwd] + 30*_b[1.lwd#c.age]
(1) [low]1.lwd + 30*[low]1.lwd#c.age = 0
```

|  | low | Odds Ratio | Std. Err. | z    | P> z  | [95% Conf. Interval] |
|--|-----|------------|-----------|------|-------|----------------------|
|  | (1) | 14.7669    | 13.5669   | 2.93 | 0.003 | 2.439264 89.39633    |

◀

## Multiple-equation models

lincom also works with multiple-equation models. The only difference is how you refer to the coefficients. Recall that for multiple-equation models, coefficients are referenced using the syntax

`[eqno]varname`

where *eqno* is the equation number or equation name and *varname* is the corresponding variable name for the coefficient; see [U] 13.5 Accessing coefficients and standard errors and [R] test for details.

### ► Example 3

Let's consider the example from [R] mlogit (Tarlov et al. 1989; Wells et al. 1989).

```
. use http://www.stata-press.com/data/r11/sysdsn1
(Health insurance data)
. mlogit insure age male nonwhite i.site, nolog
```

Multinomial logistic regression

```
Number of obs = 615
LR chi2(10) = 42.99
Prob > chi2 = 0.0000
Pseudo R2 = 0.0387
```

Log likelihood = -534.36165

| insure    | Coef.          | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-----------|----------------|-----------|-------|-------|----------------------|-----------|
| Indemnity | (base outcome) |           |       |       |                      |           |
| Prepaid   |                |           |       |       |                      |           |
| age       | -.011745       | .0061946  | -1.90 | 0.058 | -.0238862            | .0003962  |
| male      | .5616934       | .2027465  | 2.77  | 0.006 | .1643175             | .9590693  |
| nonwhite  | .9747768       | .2363213  | 4.12  | 0.000 | .5115955             | 1.437958  |
| site      |                |           |       |       |                      |           |
| 2         | .1130359       | .2101903  | 0.54  | 0.591 | -.2989296            | .5250013  |
| 3         | -.5879879      | .2279351  | -2.58 | 0.010 | -1.034733            | -.1412433 |
| _cons     | .2697127       | .3284422  | 0.82  | 0.412 | -.3740222            | .9134476  |
| Uninsure  |                |           |       |       |                      |           |
| age       | -.0077961      | .0114418  | -0.68 | 0.496 | -.0302217            | .0146294  |
| male      | .4518496       | .3674867  | 1.23  | 0.219 | -.268411             | 1.17211   |
| nonwhite  | .2170589       | .4256361  | 0.51  | 0.610 | -.6171725            | 1.05129   |
| site      |                |           |       |       |                      |           |
| 2         | -1.211563      | .4705127  | -2.57 | 0.010 | -2.133751            | -.2893747 |
| 3         | -.2078123      | .3662926  | -0.57 | 0.570 | -.9257327            | .510108   |
| _cons     | -1.286943      | .5923219  | -2.17 | 0.030 | -2.447872            | -.1260134 |

To see the estimate of the sum of the coefficient of `male` and the coefficient of `nonwhite` for the `Prepaid` outcome, we type

```
. lincom [Prepaid]male + [Prepaid]nonwhite
(1) [Prepaid]male + [Prepaid]nonwhite = 0
```

| insure | Coef.   | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|--------|---------|-----------|------|-------|----------------------|----------|
| (1)    | 1.53647 | .3272489  | 4.70 | 0.000 | .8950741             | 2.177866 |

To view the estimate as a ratio of relative risks (see [\[R\] mlogit](#) for the definition and interpretation), we specify the `rrr` option.

```
. lincom [Prepaid]male + [Prepaid]nonwhite, rrr
(1) [Prepaid]male + [Prepaid]nonwhite = 0
```

| insure | RRR      | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|--------|----------|-----------|------|-------|----------------------|----------|
| (1)    | 4.648154 | 1.521103  | 4.70 | 0.000 | 2.447517             | 8.827451 |



## Saved results

`lincom` saves the following in `r()`:

| Scalars                  |                            |
|--------------------------|----------------------------|
| <code>r(estimate)</code> | point estimate             |
| <code>r(se)</code>       | estimate of standard error |
| <code>r(df)</code>       | degrees of freedom         |

## Methods and formulas

`lincom` is implemented as an ado-file.

## References

Hosmer Jr., D. W., and S. Lemeshow. 1989. *Applied Logistic Regression*. New York: Wiley.

———. 2000. *Applied Logistic Regression*. 2nd ed. New York: Wiley.

Tarlov, A. R., J. E. Ware Jr., S. Greenfield, E. C. Nelson, E. Perrin, and M. Zubkoff. 1989. The medical outcomes study. An application of methods for monitoring the results of medical care. *Journal of the American Medical Association* 262: 925–930.

Wells, K. B., R. D. Hays, M. A. Burnam, W. H. Rogers, S. Greenfield, and J. E. Ware Jr. 1989. Detection of depressive disorder for patients receiving prepaid or fee-for-service care. Results from the Medical Outcomes Survey. *Journal of the American Medical Association* 262: 3298–3302.

## Also see

- [\[R\] nlcom](#) — Nonlinear combinations of estimators
- [\[R\] test](#) — Test linear hypotheses after estimation
- [\[R\] testnl](#) — Test nonlinear hypotheses after estimation
- [\[U\] 13.5 Accessing coefficients and standard errors](#)
- [\[U\] 20 Estimation and postestimation commands](#)

Title

linktest — Specification link test for single-equation models

Syntax

```
linktest [if] [in] [, cmd_options]
```

When `if` and `in` are not specified, the link test is performed on the same sample as the previous estimation.

Menu

Statistics > Postestimation > Tests > Specification link test for single-equation models

Description

`linktest` performs a link test for model specification after any single-equation estimation command, such as `logistic`, `regress`, `stcox`, etc.

Option

Main

*cmd\_options* must be the same options specified with the underlying estimation command.

Remarks

The form of the link test implemented here is based on an idea of [Tukey \(1949\)](#), which was further described by [Pregibon \(1980\)](#), elaborating on work in his unpublished thesis ([Pregibon 1979](#)). See [Methods and formulas](#) below for more details.

Example 1

We want to explain the mileage ratings of cars in our automobile dataset by using the weight, engine displacement, and whether the car is manufactured outside the United States:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. regress mpg weight displ foreign
```

| Source   | SS         | df | MS         |
|----------|------------|----|------------|
| Model    | 1619.71935 | 3  | 539.906448 |
| Residual | 823.740114 | 70 | 11.7677159 |
| Total    | 2443.45946 | 73 | 33.4720474 |

|                 |        |
|-----------------|--------|
| Number of obs = | 74     |
| F( 3, 70) =     | 45.88  |
| Prob > F =      | 0.0000 |
| R-squared =     | 0.6629 |
| Adj R-squared = | 0.6484 |
| Root MSE =      | 3.4304 |

| mpg          | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|--------------|-----------|-----------|-------|-------|----------------------|-----------|
| weight       | -.0067745 | .0011665  | -5.81 | 0.000 | -.0091011            | -.0044479 |
| displacement | .0019286  | .0100701  | 0.19  | 0.849 | -.0181556            | .0220129  |
| foreign      | -1.600631 | 1.113648  | -1.44 | 0.155 | -3.821732            | .6204699  |
| _cons        | 41.84795  | 2.350704  | 17.80 | 0.000 | 37.15962             | 46.53628  |

On the basis of the  $R^2$ , we are reasonably pleased with this model.

If our model really is specified correctly, then if we were to regress mpg on the prediction and the prediction squared, the prediction squared would have no explanatory power. This is what linktest does:

```
. linktest
```

| Source   | SS         | df | MS         | Number of obs = | 74     |
|----------|------------|----|------------|-----------------|--------|
| Model    | 1670.71514 | 2  | 835.357572 | F( 2, 71) =     | 76.75  |
| Residual | 772.744316 | 71 | 10.8837228 | Prob > F =      | 0.0000 |
|          |            |    |            | R-squared =     | 0.6837 |
|          |            |    |            | Adj R-squared = | 0.6748 |
| Total    | 2443.45946 | 73 | 33.4720474 | Root MSE =      | 3.299  |

| mpg    | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|--------|-----------|-----------|-------|-------|----------------------|----------|
| _hat   | -.4127198 | .6577736  | -0.63 | 0.532 | -1.724283            | .8988434 |
| _hatsq | .0338198  | .015624   | 2.16  | 0.034 | .0026664             | .0649732 |
| _cons  | 14.00705  | 6.713276  | 2.09  | 0.041 | .6211539             | 27.39294 |

We find that the prediction squared does have explanatory power, so our specification is not as good as we thought.

Although linktest is formally a test of the specification of the dependent variable, it is often interpreted as a test that, conditional on the specification, the independent variables are specified incorrectly. We will follow that interpretation and now include weight squared in our model:

```
. regress mpg weight c.weight#c.weight displ foreign
```

| Source   | SS         | df | MS         | Number of obs = | 74     |
|----------|------------|----|------------|-----------------|--------|
| Model    | 1699.02634 | 4  | 424.756584 | F( 4, 69) =     | 39.37  |
| Residual | 744.433124 | 69 | 10.7888859 | Prob > F =      | 0.0000 |
|          |            |    |            | R-squared =     | 0.6953 |
|          |            |    |            | Adj R-squared = | 0.6777 |
| Total    | 2443.45946 | 73 | 33.4720474 | Root MSE =      | 3.2846 |

| mpg               | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|-------------------|-----------|-----------|-------|-------|----------------------|-----------|
| weight            | -.0173257 | .0040488  | -4.28 | 0.000 | -.0254028            | -.0092486 |
| c.weight#c.weight | 1.87e-06  | 6.89e-07  | 2.71  | 0.008 | 4.93e-07             | 3.24e-06  |
| displacement      | -.0101625 | .0106236  | -0.96 | 0.342 | -.031356             | .011031   |
| foreign           | -2.560016 | 1.123506  | -2.28 | 0.026 | -4.801349            | -.3186832 |
| _cons             | 58.23575  | 6.449882  | 9.03  | 0.000 | 45.36859             | 71.10291  |

Now we perform the link test on our new model:



```
. linktest
```

| Source   | SS         | df | MS         | Number of obs = 74     |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | 1699.39489 | 2  | 849.697445 | F( 2, 71) = 81.08      |  |  |
| Residual | 744.06457  | 71 | 10.4797827 | Prob > F = 0.0000      |  |  |
|          |            |    |            | R-squared = 0.6955     |  |  |
|          |            |    |            | Adj R-squared = 0.6869 |  |  |
| Total    | 2443.45946 | 73 | 33.4720474 | Root MSE = 3.2372      |  |  |

| mpg    | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|--------|-----------|-----------|-------|-------|----------------------|----------|
| _hat   | 1.141987  | .7612218  | 1.50  | 0.138 | -.3758456            | 2.659821 |
| _hatsq | -.0031916 | .0170194  | -0.19 | 0.852 | -.0371272            | .0307441 |
| _cons  | -1.50305  | 8.196444  | -0.18 | 0.855 | -17.84629            | 14.84019 |

We now pass the link test.

◀

## ► Example 2

Above we followed a standard misinterpretation of the link test—when we discovered a problem, we focused on the explanatory variables of our model. We might consider varying exactly what the link test tests. The link test told us that our dependent variable was misspecified. For those with an engineering background, mpg is indeed a strange measure. It would make more sense to model energy consumption—gallons per mile—in terms of weight and displacement:

```
. gen gpm = 1/mpg
. regress gpm weight displ foreign
```

| Source   | SS         | df | MS         | Number of obs = 74     |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | .009157962 | 3  | .003052654 | F( 3, 70) = 76.33      |  |  |
| Residual | .002799666 | 70 | .000039995 | Prob > F = 0.0000      |  |  |
|          |            |    |            | R-squared = 0.7659     |  |  |
|          |            |    |            | Adj R-squared = 0.7558 |  |  |
| Total    | .011957628 | 73 | .000163803 | Root MSE = .00632      |  |  |

| gpm          | Coef.    | Std. Err. | t    | P> t  | [95% Conf. Interval] |          |
|--------------|----------|-----------|------|-------|----------------------|----------|
| weight       | .0000144 | 2.15e-06  | 6.72 | 0.000 | .0000102             | .0000187 |
| displacement | .0000186 | .0000186  | 1.00 | 0.319 | -.0000184            | .0000557 |
| foreign      | .0066981 | .0020531  | 3.26 | 0.002 | .0026034             | .0107928 |
| _cons        | .0008917 | .0043337  | 0.21 | 0.838 | -.0077515            | .009535  |

This model looks every bit as reasonable as our original model:

```
. linktest
```

| Source   | SS         | df | MS         | Number of obs = 74     |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | .009175219 | 2  | .004587609 | F( 2, 71) = 117.06     |  |  |
| Residual | .002782409 | 71 | .000039189 | Prob > F = 0.0000      |  |  |
|          |            |    |            | R-squared = 0.7673     |  |  |
|          |            |    |            | Adj R-squared = 0.7608 |  |  |
| Total    | .011957628 | 73 | .000163803 | Root MSE = .00626      |  |  |

| gpm    | Coef.    | Std. Err. | t    | P> t  | [95% Conf. Interval] |          |
|--------|----------|-----------|------|-------|----------------------|----------|
| _hat   | .6608413 | .515275   | 1.28 | 0.204 | -.3665877            | 1.68827  |
| _hatsq | 3.275857 | 4.936655  | 0.66 | 0.509 | -6.567553            | 13.11927 |
| _cons  | .008365  | .0130468  | 0.64 | 0.523 | -.0176496            | .0343795 |

Specifying the model in terms of gallons per mile also solves the specification problem and results in a more parsimonious specification.



➤ Example 3

The link test can be used with any single-equation estimation procedure, not solely regression. Let’s turn our problem around and attempt to explain whether a car is manufactured outside the United States by its mileage rating and weight. To save paper, we will specify `logit`’s `nolog` option, which suppresses the iteration log:

```
. logit foreign mpg weight, nolog
Logistic regression Number of obs = 74
 LR chi2(2) = 35.72
 Prob > chi2 = 0.0000
Log likelihood = -27.175156 Pseudo R2 = 0.3966
```

| foreign | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|---------|-----------|-----------|-------|-------|----------------------|----------|
| mpg     | -.1685869 | .0919174  | -1.83 | 0.067 | -.3487418            | .011568  |
| weight  | -.0039067 | .0010116  | -3.86 | 0.000 | -.0058894            | -.001924 |
| _cons   | 13.70837  | 4.518707  | 3.03  | 0.002 | 4.851864             | 22.56487 |

When we run `linktest` after `logit`, the result is another `logit` specification:

```
. linktest, nolog
Logistic regression Number of obs = 74
 LR chi2(2) = 36.83
 Prob > chi2 = 0.0000
Log likelihood = -26.615714 Pseudo R2 = 0.4090
```

| foreign | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|---------|-----------|-----------|-------|-------|----------------------|----------|
| _hat    | .8438531  | .2738759  | 3.08  | 0.002 | .3070661             | 1.38064  |
| _hatsq  | -.1559115 | .1568642  | -0.99 | 0.320 | -.4633596            | .1515366 |
| _cons   | .2630557  | .4299598  | 0.61  | 0.541 | -.57965              | 1.105761 |

The link test reveals no problems with our specification.

If there had been a problem, we would have been virtually forced to accept the misinterpretation of the link test—we would have reconsidered our specification of the independent variables. When using `logit`, we have no control over the specification of the dependent variable other than to change likelihood functions.

We admit to having seen a dataset once for which the link test rejected the `logit` specification. We did change the likelihood function, refitting the model using `probit`, and satisfied the link test. `Probit` has thinner tails than `logit`. In general, however, you will not be so lucky.



## □ Technical note

You should specify the same options with `linktest` that you do with the estimation command, although you do not have to follow this advice as literally as we did in the preceding example. `logit`'s `nolog` option merely suppresses a part of the output, not what is estimated. We specified `nolog` both times to save paper.

If you are testing a tobit model, you must specify the censoring points just as you do with the `tobit` command.

If you are not sure which options are important, duplicate exactly what you specified on the estimation command.

If you do not specify `if exp` or `in range` with `linktest`, Stata will by default perform the link test on the same sample as the previous estimation. Suppose that you omitted some data when performing your estimation, but want to calculate the link test on all the data, which you might do if you believe the model is appropriate for all the data. You would type `linktest if e(sample) < .` to do this.

□

## Saved results

`linktest` saves the following in `r()`:

Scalars

`r(t)` *t* statistic on `_hatsq`      `r(df)` degrees of freedom

`linktest` is *not* an estimation command in the sense that it leaves previous estimation results unchanged. For instance, after running a regression and performing the link test, typing `regress` without arguments after the link test still replays the original regression.

For integrating an estimation command with `linktest`, `linktest` assumes that the name of the estimation command is stored in `e(cmd)` and that the name of the dependent variable is stored in `e(depvar)`. After estimation, it assumes that the number of degrees of freedom for the *t* test is given by `e(df_m)` if the macro is defined.

If the estimation command reports *z* statistics instead of *t* statistics, `linktest` will also report *z* statistics. The *z* statistic, however, is still returned in `r(t)`, and `r(df)` is set to a missing value.

## Methods and formulas

`linktest` is implemented as an ado-file.

The link test is based on the idea that if a regression or regression-like equation is properly specified, you should be able to find no additional independent variables that are significant except by chance. One kind of specification error is called a link error. In regression, this means that the dependent variable needs a transformation or “link” function to properly relate to the independent variables. The idea of a link test is to add an independent variable to the equation that is especially likely to be significant if there is a link error.

Let

$$y = f(\mathbf{X}\beta)$$

be the model and  $\hat{\beta}$  be the parameter estimates. `linktest` calculates

$$\_hat = X\hat{\beta}$$

and

$$\_hatsq = \_hat^2$$

The model is then refit with these two variables, and the test is based on the significance of `\_hatsq`. This is the form suggested by [Pregibon \(1979\)](#) based on an idea of [Tukey \(1949\)](#). [Pregibon \(1980\)](#) suggests a slightly different method that has come to be known as “Pregibon’s goodness-of-link test”. We prefer the older version because it is universally applicable, straightforward, and a good second-order approximation. It can be applied to any single-equation estimation technique, whereas Pregibon’s more recent tests are estimation-technique specific.

## References

- Pregibon, D. 1979. Data analytic methods for generalized linear models. PhD diss., University of Toronto.
- . 1980. Goodness of link tests for generalized linear models. *Applied Statistics* 29: 15–24.
- Tukey, J. W. 1949. One degree of freedom for non-additivity. *Biometrics* 5: 232–242.

## Also see

[\[R\] regress postestimation](#) — Postestimation tools for regress

Syntax

Zero-skewness log

```
lnskew0 newvar = exp [if] [in] [, options]
```

Zero-skewness Box–Cox transform

```
bcskew0 newvar = exp [if] [in] [, options]
```

| options               | description                                                                                                                                                       |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Main                  |                                                                                                                                                                   |
| <code>delta(#)</code> | increment for derivative of skewness function; default is <code>delta(0.02)</code> for <code>lnskew0</code> and <code>delta(0.01)</code> for <code>bcskew0</code> |
| <code>zero(#)</code>  | value for determining convergence; default is <code>zero(0.001)</code>                                                                                            |
| <code>level(#)</code> | set confidence level; default is <code>level(95)</code>                                                                                                           |

Menu

Inskew0

Data > Create or change data > Other variable-creation commands > Zero-skewness log transform

bcskew0

Data > Create or change data > Other variable-creation commands > Box-Cox transform

Description

`lnskew0` creates  $newvar = \ln(\pm exp - k)$ , choosing  $k$  and the sign of  $exp$  so that the skewness of  $newvar$  is zero.

`bcskew0` creates  $newvar = (exp^\lambda - 1)/\lambda$ , the Box–Cox power transformation (Box and Cox 1964), choosing  $\lambda$  so that the skewness of  $newvar$  is zero.  $exp$  must be strictly positive. Also see [R] [boxcox](#) for maximum likelihood estimation of  $\lambda$ .

Options

|                                                                                                                                                                                                                                                                                         |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Main                                                                                                                                                                                                                                                                                    |
| <code>delta(#)</code> specifies the increment used for calculating the derivative of the skewness function with respect to $k$ ( <code>lnskew0</code> ) or $\lambda$ ( <code>bcskew0</code> ). The default values are 0.02 for <code>lnskew0</code> and 0.01 for <code>bcskew0</code> . |

`zero(#)` specifies a value for skewness to determine convergence that is small enough to be considered zero and is, by default, 0.001.

`level(#)` specifies the confidence level for the confidence interval for  $k$  (`lnskew0`) or  $\lambda$  (`bcskew0`). The confidence interval is calculated only if `level()` is specified. `#` is specified as an integer; 95 means 95% confidence intervals. The `level()` option is honored only if the number of observations exceeds 7.

Remarks

► Example 1: Inskew0

Using our automobile dataset (see [U] 1.2.2 Example datasets), we want to generate a new variable equal to  $\ln(\text{mpg} - k)$  to be approximately normally distributed. `mpg` records the miles per gallon for each of our cars. One feature of the normal distribution is that it has skewness 0.

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. lnskew0 lnmpg = mpg
```

| Transform | k        | [95% Conf. Interval] | Skewness  |
|-----------|----------|----------------------|-----------|
| ln(mpg-k) | 5.383659 | (not calculated)     | -7.05e-06 |

This created the new variable `lnmpg = ln(mpg - 5.384)`:

```
. describe lnmpg
```

| variable name | storage type | display format | value label | variable label   |
|---------------|--------------|----------------|-------------|------------------|
| lnmpg         | float        | %9.0g          |             | ln(mpg-5.383659) |

Because we did not specify the `level()` option, no confidence interval was calculated. At the outset, we could have typed

```
. use http://www.stata-press.com/data/r11/auto, clear
(Automobile Data)
. lnskew0 lnmpg = mpg, level(95)
```

| Transform | k        | [95% Conf. Interval] | Skewness  |
|-----------|----------|----------------------|-----------|
| ln(mpg-k) | 5.383659 | -17.12339 9.892416   | -7.05e-06 |

The confidence interval is calculated under the assumption that  $\ln(\text{mpg} - k)$  really does have a normal distribution. It would be perfectly reasonable to use `lnskew0`, even if we did not believe that the transformed variable would have a normal distribution—if we literally wanted the zero-skewness transform—although, then the confidence interval would be an approximation of unknown quality to the true confidence interval. If we now wanted to test the believability of the confidence interval, we could also test our new variable `lnmpg` by using `swilk` with the `lnnormal` option.

## □ Technical note

`lnskew0` and `bcskew0` report the resulting skewness of the variable merely to reassure you of the accuracy of its results. In our example above, `lnskew0` found  $k$  such that the resulting skewness was  $-7 \times 10^{-6}$ , a number close enough to zero for all practical purposes. If we wanted to make it even smaller, we could specify the `zero()` option. Typing `lnskew0 new=mpg, zero(1e-8)` changes the estimated  $k$  to 5.383552 from 5.383659 and reduces the calculated skewness to  $-2 \times 10^{-11}$ .

When you request a confidence interval, `lnskew0` may report the lower confidence interval as ‘.’, which should be taken as indicating the lower confidence limit  $k_L = -\infty$ . (This cannot happen with `bcskew0`.)

As an example, consider a sample of size  $n$  on  $x$  and assume that the skewness of  $x$  is positive, but not significantly so, at the desired significance level—say, 5%. Then no matter how large and negative you make  $k_L$ , there is no value extreme enough to make the skewness of  $\ln(x - k_L)$  equal the corresponding percentile (97.5 for a 95% confidence interval) of the distribution of skewness in a normal distribution of the same sample size. You cannot do this because the distribution of  $\ln(x - k_L)$  tends to that of  $x$ —apart from location and scale shift—as  $x \rightarrow \infty$ . This “problem” never applies to the upper confidence limit,  $k_U$ , because the skewness of  $\ln(x - k_U)$  tends to  $-\infty$  as  $k$  tends upward to the minimum value of  $x$ .

□

## ▷ Example 2: `bcskew0`

In [example 1](#), using `lnskew0` with a variable such as `mpg` is probably undesirable. `mpg` has a natural zero, and we are shifting that zero arbitrarily. On the other hand, use of `lnskew0` with a variable such as temperature measured in Fahrenheit or Celsius would be more appropriate, as the zero is indeed arbitrary.

For a variable like `mpg`, it makes more sense to use the Box–Cox power transform ([Box and Cox 1964](#)):

$$y^{(\lambda)} = \frac{y^\lambda - 1}{\lambda}$$

$\lambda$  is free to take on any value, but  $y^{(1)} = y - 1$ ,  $y^{(0)} = \ln(y)$ , and  $y^{(-1)} = 1 - 1/y$ .

`bcskew0` works like `lnskew0`:

```
. bcskew0 bcmpg = mpg, level(95)
```

| Transform   | L         | [95% Conf. Interval] |          | Skewness |
|-------------|-----------|----------------------|----------|----------|
| (mpg^L-1)/L | -.3673283 | -1.212752            | .4339645 | .0001898 |

The 95% confidence interval includes  $\lambda = -1$  ( $\lambda$  is labeled L in the output), which has a rather more pleasing interpretation—gallons per mile—than  $(\text{mpg}^{-0.3673} - 1)/(-0.3673)$ . The confidence interval, however, is calculated assuming that the power transformed variable is normally distributed. It makes perfect sense to use `bcskew0`, even when you do not believe that the transformed variable will be normally distributed, but then the confidence interval is an approximation of unknown quality. If you believe that the transformed data are normally distributed, you can alternatively use `boxcox` to estimate  $\lambda$ ; see [\[R\] boxcox](#).

◀

## Saved results

lnskew0 and bcskew0 save the following in `r()`:

|                          |                                            |
|--------------------------|--------------------------------------------|
| Scalars                  |                                            |
| <code>r(gamma)</code>    | $k$ (lnskew0)                              |
| <code>r(lambda)</code>   | $\lambda$ (bcskew0)                        |
| <code>r(lb)</code>       | lower bound of confidence interval         |
| <code>r(ub)</code>       | upper bound of confidence interval         |
| <code>r(skewness)</code> | resulting skewness of transformed variable |

## Methods and formulas

lnskew0 and bcskew0 are implemented as ado-files.

Skewness is as calculated by `summarize`; see [\[R\] summarize](#). Newton’s method with numeric, uncentered derivatives is used to estimate  $k$  (lnskew0) and  $\lambda$  (bcskew0). For lnskew0, the initial value is chosen so that the minimum of  $x - k$  is 1, and thus  $\ln(x - k)$  is 0. bcskew0 starts with  $\lambda = 1$ .

## Acknowledgment

lnskew0 and bcskew0 were written by Patrick Royston of the MRC Clinical Trials Unit, London.

## Reference

Box, G. E. P., and D. R. Cox. 1964. An analysis of transformations. *Journal of the Royal Statistical Society, Series B* 26: 211–252.

## Also see

- [\[R\] ladder](#) — Ladder of powers
- [\[R\] boxcox](#) — Box–Cox regression models
- [\[R\] swilk](#) — Shapiro–Wilk and Shapiro–Francia tests for normality



# Title

**log** — Echo copy of session to file

## Syntax

*Report status of log file*

`log`

`log query [ logname | _all ]`

*Open log file*

`log using filename [ , append replace [ _text | _smcl ] name(logname) ]`

*Close log*

`log _close [ logname | _all ]`

*Temporarily suspend logging or resume logging*

`log { _off | on } [ logname ]`

*Report status of command log file*

`cmdlog`

*Open command log file*

`cmdlog using filename [ , append replace ]`

*Close command log, temporarily suspend logging, or resume logging*

`cmdlog { _close | on | _off }`

*Set default format for logs*

`set logtype { _text | _smcl } [ , _permanently ]`

*Specify screen width*

`_set _linesize #`

In addition to using the `log` command, you may access the capabilities of `log` by selecting **File > Log** from the menu and choosing one of the options in the list.

## Menu

File > Log

## Description

`log` allows you to make a full record of your Stata session. A log is a file containing what you type and Stata's output. You may start multiple log files at the same time, and you may refer to them with a *logname*. If you do not specify a *logname*, Stata will use the name `<unnamed>`.

`cmdlog` allows you to make a record of what you type during your Stata session. A command log contains only what you type, so it is a subset of a full log.

You can make full logs, command logs, or both simultaneously. Neither is produced until you tell Stata to start logging.

Command logs are always ASCII text files, making them easy to convert into do-files. (In this respect, it would make more sense if the default extension of a command log file was `.do` because command logs are do-files. The default is `.txt`, not `.do`, however, to keep you from accidentally overwriting your important do-files.)

Full logs are recorded in one of two formats: Stata Markup and Control Language (SMCL) or text (meaning ASCII). The default is SMCL, but you can use `set logtype` to change that, or you can specify an option to state the format you wish. We recommend SMCL because it preserves fonts and colors. SMCL logs can be converted to ASCII text or to other formats by using the `translate` command; see [R] [translate](#). You can also use `translate` to produce printable versions of SMCL logs. SMCL logs can be viewed and printed from the Viewer, as can any text file; see [R] [view](#).

When using multiple log files, you may have up to five SMCL logs and five text logs open at the same time.

`log` or `cmdlog`, typed without arguments, reports the status of logging. `log query`, when passed an optional *logname*, reports the status of that log.

`log using` and `cmdlog using` open a log file. `log close` and `cmdlog close` close the file. Between times, `log off` and `cmdlog off`, and `log on` and `cmdlog on`, can temporarily suspend and resume logging.

If *filename* is specified without an extension, one of the suffixes `.smcl`, `.log`, or `.txt` is added. The extension `.smcl` or `.log` is added by `log`, depending on whether the file format is SMCL or ASCII text. The extension `.txt` is added by `cmdlog`. If *filename* contains embedded spaces, remember to enclose it in double quotes.

`set logtype` specifies the default format in which full logs are to be recorded. Initially, full logs are recorded in SMCL format.

`set linesize` specifies the maximum width, in characters, of Stata output. Most commands in Stata do not respect `linesize`, because it is not important for most commands. Most users never need to `set linesize`, because it will automatically be reset if you resize your Results window. This is also why there is no `permanently` option allowed with `set linesize`. `set linesize` is for use with commands such as `list` and `display` and is typically used by programmers who wish the output of those commands to be wider or narrower than the current width of the Results window.

## Options for use with both log and cmdlog

**append** specifies that results be appended to an existing file. If the file does not already exist, a new file is created.

**replace** specifies that *filename*, if it already exists, be overwritten. When you do not specify either **replace** or **append**, the file is assumed to be new. If the specified file already exists, an error message is issued and logging is not started.

## Options for use with log

**text** and **smcl** specify the format in which the log is to be recorded. The default is complicated to describe but is what you would expect:

If you specify the file as *filename.smcl*, the default is to write the log in SMCL format (regardless of the value of **set logtype**).

If you specify the file as *filename.log*, the default is to write the log in text format (regardless of the value of **set logtype**).

If you type *filename* without an extension and specify neither the **smcl** option nor the **text** option, the default is to write the file according to the value of **set logtype**. If you have not **set logtype**, then that default is SMCL. Also, the *filename* you specified will be fixed to read *filename.smcl* if a SMCL log is being created or *filename.log* if a text log is being created.

If you specify either the **text** or **smcl** option, then what you specify determines how the log is written. If *filename* was specified without an extension, the appropriate extension is added for you.

If you open multiple log files, you may choose a different format for each file.

**name(logname)** specifies an optional name you may use to refer to the log while it is open. You can start multiple log files, give each a different *logname*, and then close, temporarily suspend, or resume them each individually.

## Option for use with set logtype

**permanently** specifies that, in addition to making the change right now, the **logtype** setting be remembered and become the default setting when you invoke Stata.

## Remarks

For a detailed explanation of logs, see [\[U\] 15 Saving and printing output—log files](#).

When you open a full log, the default is to show the name of the file and a time and date stamp:

```
. log using myfile

name: <unnamed>
log: C:\data\proj1\myfile.smcl
log type: smcl
opened on: 12 Jan 2009, 12:28:23
.
```

The above information will appear in the log. If you do not want this information to appear, precede the command by **quietly**:

```
. quietly log using myfile
```

quietly will not suppress any error messages or anything else you need to know.

Similarly, when you close a full log, the default is to show the full information,

```
. log close
 name: <unnamed>
 log: C:\data\proj1\myfile.smcl
 log type: smcl
 closed on: 12 Jan 2009, 12:32:41
```

---

and that information will also appear in the log. If you want to suppress that, type `quietly log close`.

## Saved results

`log` and `cmdlog` save the following in `r()`:

### Macros

|                          |                |
|--------------------------|----------------|
| <code>r(name)</code>     | <i>logname</i> |
| <code>r(filename)</code> | name of file   |
| <code>r(status)</code>   | on or off      |
| <code>r(type)</code>     | smcl or text   |

`log query _all` saves the following in `r()`:

### Scalars

|                         |                          |
|-------------------------|--------------------------|
| <code>r(numlogs)</code> | number of open log files |
|-------------------------|--------------------------|

For each open log file, `log query _all` also saves

|                           |                |
|---------------------------|----------------|
| <code>r(name#)</code>     | <i>logname</i> |
| <code>r(filename#)</code> | name of file   |
| <code>r(status#)</code>   | on or off      |
| <code>r(type#)</code>     | smcl or text   |

where `#` varies between 1 and the value of `r(numlogs)`. Be aware that `#` will not necessarily represent the order in which the log files were first opened, nor will it necessarily remain constant for a given log file upon multiple calls to `log query`.

## Also see

[R] [translate](#) — Print and translate logs

[R] [query](#) — Display system parameters

[GSM] [16 Saving and printing results by using logs](#)

[GSW] [16 Saving and printing results by using logs](#)

[GSU] [16 Saving and printing results by using logs](#)

[U] [15 Saving and printing output—log files](#)

Syntax

```
logistic depvar indepvars [if] [in] [weight] [, options]
```

| <i>options</i>                               | description                                                                                                                                              |
|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                        |                                                                                                                                                          |
| <code>offset(<i>varname</i>)</code>          | include <i>varname</i> in model with coefficient constrained to 1                                                                                        |
| <code>asis</code>                            | retain perfect predictor variables                                                                                                                       |
| <code>constraints(<i>constraints</i>)</code> | apply specified linear constraints                                                                                                                       |
| <code>collinear</code>                       | keep collinear variables                                                                                                                                 |
| SE/Robust                                    |                                                                                                                                                          |
| <code>vce(<i>vcetype</i>)</code>             | <i>vcetype</i> may be <code>oim</code> , <code>robust</code> , <code>cluster <i>clustvar</i></code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                                    |                                                                                                                                                          |
| <code>level(#)</code>                        | set confidence level; default is <code>level(95)</code>                                                                                                  |
| <code>coef</code>                            | report estimated coefficients                                                                                                                            |
| <code>nocnsreport</code>                     | do not display constraints                                                                                                                               |
| <code>display_options</code>                 | control spacing and display of omitted variables and base and empty cells                                                                                |
| Maximization                                 |                                                                                                                                                          |
| <code>maximize_options</code>                | control the maximization process; seldom used                                                                                                            |
| <sup>†</sup> <code>coeflegend</code>         | display coefficients' legend instead of coefficient table                                                                                                |

<sup>†</sup>`coeflegend` does not appear in the dialog box.

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

`bootstrap`, `by`, `fracpoly`, `jackknife`, `mfp`, `mi estimate`, `nestreg`, `rolling`, `statsby`, `stepwise`, and `svy` are allowed; see [U] 11.1.10 Prefix commands.

`vce(bootstrap)` and `vce(jackknife)` are not allowed with the `mi estimate` prefix.

Weights are not allowed with the `bootstrap` prefix.

`vce()` and weights are not allowed with the `svy` prefix.

`fwweights`, `iweights`, and `pweights` are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Binary outcomes > Logistic regression (reporting odds ratios)

## Description

`logistic` fits a logistic regression model of *depvar* on *varlist*, where *depvar* is a 0/1 variable (or, more precisely, a 0/non-0 variable). Without arguments, `logistic` redisplay the last `logistic` estimates. `logistic` displays estimates as odds ratios; to view coefficients, type `logit` after running `logistic`. To obtain odds ratios for any covariate pattern relative to another, see [\[R\] `lincom`](#).

## Options

### Model

`offset(varname)`, `constraints(constraints)`, `collinear`; see [\[R\] `estimation options`](#).

`asis` forces retention of perfect predictor variables and their associated perfectly predicted observations and may produce instabilities in maximization; see [\[R\] `probit`](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [\[R\] `vce\_option`](#).

### Reporting

`level(#)`; see [\[R\] `estimation options`](#).

`coef` causes `logistic` to report the estimated coefficients rather than the odds ratios (exponentiated coefficients). `coef` may be specified when the model is fit or may be used later to redisplay results. `coef` affects only how results are displayed and not how they are estimated.

`nocnsreport`; see [\[R\] `estimation options`](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [\[R\] `estimation options`](#).

### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [\[R\] `maximize`](#). These options are seldom used.

The following option is available with `logistic` but is not shown in the dialog box:

`coeflegend`; see [\[R\] `estimation options`](#).

## Remarks

Remarks are presented under the following headings:

*logistic and logit*

*Robust estimate of variance*

## logistic and logit

`logistic` provides an alternative and preferred way to fit maximum-likelihood logit models, the other choice being `logit` ([R] [logit](#)).

First, let's dispose of some confusing terminology. We use the words `logit` and `logistic` to mean the same thing: maximum likelihood estimation. To some, one or the other of these words connotes transforming the dependent variable and using weighted least squares to fit the model, but that is not how we use either word here. Thus the `logit` and `logistic` commands produce the same results.

The `logistic` command is generally preferred to `logit` because `logistic` presents the estimates in terms of odds ratios rather than coefficients. To a few people, this may seem a disadvantage, but you can type `logit` without arguments after `logistic` to see the underlying coefficients.

Nevertheless, [R] [logit](#) is still worth reading because `logistic` shares the same features as `logit`, including omitting variables due to collinearity or one-way causation.

For an introduction to logistic regression, see [Lemeshow and Hosmer Jr. \(2005\)](#), [Pagano and Gauvreau \(2000, 470–487\)](#), or [Pampel \(2000\)](#); for a complete but nonmathematical treatment, see [Kleinbaum and Klein \(2002\)](#); and for a thorough discussion, see [Hosmer Jr. and Lemeshow \(2000\)](#). See [Gould \(2000\)](#) for a discussion of the interpretation of logistic regression. See [Dupont \(2009\)](#) for a discussion of logistic regression with examples using Stata. For a discussion using Stata with an emphasis on model specification, see [Vittinghoff et al. \(2005\)](#).

Stata has a variety of commands for performing estimation when the dependent variable is dichotomous or polytomous. See [Long and Freese \(2006\)](#) for a book devoted to fitting these models with Stata. Here is a list of some estimation commands that may be of interest. See [1] [estimation commands](#) for a complete list of all of Stata's estimation commands.

*(Continued on next page)*

---

|                         |                                      |                                                                                                   |
|-------------------------|--------------------------------------|---------------------------------------------------------------------------------------------------|
| <code>asclogit</code>   | [R] <a href="#">asclogit</a>         | Alternative-specific conditional logit (McFadden's choice) model                                  |
| <code>asmprobit</code>  | [R] <a href="#">asmprobit</a>        | Alternative-specific multinomial probit regression                                                |
| <code>asroprobit</code> | [R] <a href="#">asroprobit</a>       | Alternative-specific rank-ordered probit regression                                               |
| <code>binreg</code>     | [R] <a href="#">binreg</a>           | Generalized linear models for the binomial family                                                 |
| <code>biprobit</code>   | [R] <a href="#">biprobit</a>         | Bivariate probit regression                                                                       |
| <code>blogit</code>     | [R] <a href="#">glogit</a>           | Logit regression for grouped data                                                                 |
| <code>bprobit</code>    | [R] <a href="#">glogit</a>           | Probit regression for grouped data                                                                |
| <code>clogit</code>     | [R] <a href="#">clogit</a>           | Conditional (fixed-effects) logistic regression                                                   |
| <code>cloglog</code>    | [R] <a href="#">cloglog</a>          | Complementary log-log regression                                                                  |
| <code>exlogistic</code> | [R] <a href="#">exlogistic</a>       | Exact logistic regression                                                                         |
| <code>glm</code>        | [R] <a href="#">glm</a>              | Generalized linear models                                                                         |
| <code>glogit</code>     | [R] <a href="#">glogit</a>           | Weighted least-squares logistic regression for grouped data                                       |
| <code>gprobit</code>    | [R] <a href="#">glogit</a>           | Weighted least-squares probit regression for grouped data                                         |
| <code>heckprob</code>   | [R] <a href="#">heckprob</a>         | Probit model with selection                                                                       |
| <code>hetprob</code>    | [R] <a href="#">hetprob</a>          | Heteroskedastic probit model                                                                      |
| <code>ivprobit</code>   | [R] <a href="#">ivprobit</a>         | Probit model with endogenous regressors                                                           |
| <code>logit</code>      | [R] <a href="#">logit</a>            | Logistic regression, reporting coefficients                                                       |
| <code>mlogit</code>     | [R] <a href="#">mlogit</a>           | Multinomial (polytomous) logistic regression                                                      |
| <code>mprobit</code>    | [R] <a href="#">mprobit</a>          | Multinomial probit regression                                                                     |
| <code>nlogit</code>     | [R] <a href="#">nlogit</a>           | Nested logit regression (RUM-consistent and nonnormalized)                                        |
| <code>ologit</code>     | [R] <a href="#">ologit</a>           | Ordered logistic regression                                                                       |
| <code>oprobit</code>    | [R] <a href="#">oprobit</a>          | Ordered probit regression                                                                         |
| <code>probit</code>     | [R] <a href="#">probit</a>           | Probit regression                                                                                 |
| <code>rologit</code>    | [R] <a href="#">rologit</a>          | Rank-ordered logistic regression                                                                  |
| <code>scobit</code>     | [R] <a href="#">scobit</a>           | Skewed logistic regression                                                                        |
| <code>slogit</code>     | [R] <a href="#">slogit</a>           | Stereotype logistic regression                                                                    |
| <code>svy: cmd</code>   | [SVY] <a href="#">svy estimation</a> | Survey versions of many of these commands are available; see <a href="#">[SVY] svy estimation</a> |
| <code>xtcloglog</code>  | [XT] <a href="#">xtcloglog</a>       | Random-effects and population-averaged cloglog models                                             |
| <code>xtgee</code>      | [XT] <a href="#">xtgee</a>           | GEE population-averaged generalized linear models                                                 |
| <code>xtlogit</code>    | [XT] <a href="#">xtlogit</a>         | Fixed-effects, random-effects, and population-averaged logit models                               |
| <code>xtprobit</code>   | [XT] <a href="#">xtprobit</a>        | Random-effects and population-averaged probit models                                              |

---



## ► Example 1

Consider the following dataset from a study of risk factors associated with low birthweight described in Hosmer Jr. and Lemeshow (2000, 25).

```
. use http://www.stata-press.com/data/r11/lbw
(Hosmer & Lemeshow data)

. describe

Contains data from http://www.stata-press.com/data/r11/lbw.dta
obs: 189 Hosmer & Lemeshow data
vars: 11 15 Jan 2009 05:01
size: 3,402 (99.9% of memory free)
```

| variable name | storage type | display format | value label | variable label                                     |
|---------------|--------------|----------------|-------------|----------------------------------------------------|
| id            | int          | %8.0g          |             | identification code                                |
| low           | byte         | %8.0g          |             | birthweight<2500g                                  |
| age           | byte         | %8.0g          |             | age of mother                                      |
| lwt           | int          | %8.0g          |             | weight at last menstrual period                    |
| race          | byte         | %8.0g          | race        | race                                               |
| smoke         | byte         | %8.0g          |             | smoked during pregnancy                            |
| ptl           | byte         | %8.0g          |             | premature labor history (count)                    |
| ht            | byte         | %8.0g          |             | has history of hypertension                        |
| ui            | byte         | %8.0g          |             | presence, uterine irritability                     |
| ftv           | byte         | %8.0g          |             | number of visits to physician during 1st trimester |
| bwt           | int          | %8.0g          |             | birthweight (grams)                                |

Sorted by:

We want to investigate the causes of low birthweight. Here race is a categorical variable indicating whether a person is white (race = 1), black (race = 2), or some other race (race = 3). We want indicator (dummy) variables for race included in the regression, so we will use factor variables.

```
. logistic low age lwt i.race smoke ptl ht ui
```

|                           |               |   |        |
|---------------------------|---------------|---|--------|
| Logistic regression       | Number of obs | = | 189    |
|                           | LR chi2(8)    | = | 33.22  |
|                           | Prob > chi2   | = | 0.0001 |
| Log likelihood = -100.724 | Pseudo R2     | = | 0.1416 |

|       | low | Odds Ratio | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-------|-----|------------|-----------|-------|-------|----------------------|----------|
| age   |     | .9732636   | .0354759  | -0.74 | 0.457 | .9061578             | 1.045339 |
| lwt   |     | .9849634   | .0068217  | -2.19 | 0.029 | .9716834             | .9984249 |
| race  |     |            |           |       |       |                      |          |
| 2     |     | 3.534767   | 1.860737  | 2.40  | 0.016 | 1.259736             | 9.918406 |
| 3     |     | 2.368079   | 1.039949  | 1.96  | 0.050 | 1.001356             | 5.600207 |
| smoke |     | 2.517698   | 1.00916   | 2.30  | 0.021 | 1.147676             | 5.523162 |
| ptl   |     | 1.719161   | .5952579  | 1.56  | 0.118 | .8721455             | 3.388787 |
| ht    |     | 6.249602   | 4.322408  | 2.65  | 0.008 | 1.611152             | 24.24199 |
| ui    |     | 2.1351     | .9808153  | 1.65  | 0.099 | .8677528             | 5.2534   |

The odds ratios are for a one-unit change in the variable. If we wanted the odds ratio for age to be in terms of 4-year intervals, we would type

```
. gen age4 = age/4
. logistic low age4 lwt i.race smoke ptl ht ui
(output omitted)
```

After `logistic`, we can type `logit` to see the model in terms of coefficients and standard errors:

```
. logit
Logistic regression Number of obs = 189
 LR chi2(8) = 33.22
 Prob > chi2 = 0.0001
Log likelihood = -100.724 Pseudo R2 = 0.1416
```

| low   | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-------|-----------|-----------|-------|-------|----------------------|-----------|
| age4  | -.1084012 | .1458017  | -0.74 | 0.457 | -.3941673            | .1773649  |
| lwt   | -.0151508 | .0069259  | -2.19 | 0.029 | -.0287253            | -.0015763 |
| race  |           |           |       |       |                      |           |
| 2     | 1.262647  | .5264101  | 2.40  | 0.016 | .2309024             | 2.294392  |
| 3     | .8620792  | .4391532  | 1.96  | 0.050 | .0013548             | 1.722804  |
| smoke | .9233448  | .4008266  | 2.30  | 0.021 | .137739              | 1.708951  |
| ptl   | .5418366  | .346249   | 1.56  | 0.118 | -.136799             | 1.220472  |
| ht    | 1.832518  | .6916292  | 2.65  | 0.008 | .4769494             | 3.188086  |
| ui    | .7585135  | .4593768  | 1.65  | 0.099 | -.1418484            | 1.658875  |
| _cons | .4612239  | 1.20459   | 0.38  | 0.702 | -1.899729            | 2.822176  |

If we wanted to see the `logistic` output again, we would type `logistic` without arguments.

➤ Example 2

We can specify the confidence interval for the odds ratios with the `level()` option, and we can do this either at estimation time or when replaying the model. For instance, to see our first model in [example 1](#) with narrower, 90% confidence intervals, we might type

```
. logistic, level(90)
Logistic regression Number of obs = 189
 LR chi2(8) = 33.22
 Prob > chi2 = 0.0001
Log likelihood = -100.724 Pseudo R2 = 0.1416
```

| low   | Odds Ratio | Std. Err. | z     | P> z  | [90% Conf. Interval] |          |
|-------|------------|-----------|-------|-------|----------------------|----------|
| age   | .9732636   | .0354759  | -0.74 | 0.457 | .9166258             | 1.033401 |
| lwt   | .9849634   | .0068217  | -2.19 | 0.029 | .9738063             | .9962483 |
| race  |            |           |       |       |                      |          |
| 2     | 3.534767   | 1.860737  | 2.40  | 0.016 | 1.487028             | 8.402379 |
| 3     | 2.368079   | 1.039949  | 1.96  | 0.050 | 1.149971             | 4.876471 |
| smoke | 2.517698   | 1.00916   | 2.30  | 0.021 | 1.302185             | 4.867819 |
| ptl   | 1.719161   | .5952579  | 1.56  | 0.118 | .9726876             | 3.038505 |
| ht    | 6.249602   | 4.322408  | 2.65  | 0.008 | 2.003487             | 19.49478 |
| ui    | 2.1351     | .9808153  | 1.65  | 0.099 | 1.00291              | 4.545424 |

## Robust estimate of variance

If you specify `vce(robust)`, Stata reports the robust estimate of variance described in [U] 20.16 **Obtaining robust variance estimates**. Here is the model previously fit with the robust estimate of variance:

```
. logistic low age lwt i.race smoke ptl ht ui, vce(robust)
Logistic regression Number of obs = 189
 Wald chi2(8) = 29.02
 Prob > chi2 = 0.0003
Log pseudolikelihood = -100.724 Pseudo R2 = 0.1416
```

|       | low | Odds Ratio | Robust<br>Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-------|-----|------------|---------------------|-------|-------|----------------------|----------|
| age   |     | .9732636   | .0329376            | -0.80 | 0.423 | .9108015             | 1.040009 |
| lwt   |     | .9849634   | .0070209            | -2.13 | 0.034 | .9712984             | .9988206 |
| race  |     |            |                     |       |       |                      |          |
| 2     |     | 3.534767   | 1.793616            | 2.49  | 0.013 | 1.307504             | 9.556051 |
| 3     |     | 2.368079   | 1.026563            | 1.99  | 0.047 | 1.012512             | 5.538501 |
| smoke |     | 2.517698   | .9736417            | 2.39  | 0.017 | 1.179852             | 5.372537 |
| ptl   |     | 1.719161   | .7072902            | 1.32  | 0.188 | .7675715             | 3.850476 |
| ht    |     | 6.249602   | 4.102026            | 2.79  | 0.005 | 1.726445             | 22.6231  |
| ui    |     | 2.1351     | 1.042775            | 1.55  | 0.120 | .8197749             | 5.560858 |

Also you can specify `vce(cluster clustvar)` and then, within cluster, relax the assumption of independence. To illustrate this, we have made some fictional additions to the low-birthweight data.

Say that these data are not a random sample of mothers but instead are a random sample of mothers from a random sample of hospitals. In fact, that may be true—we do not know the history of these data.

Hospitals specialize, and it would not be too incorrect to say that some hospitals specialize in more difficult cases. We are going to show two extremes. In one, all hospitals are alike, but we are going to estimate under the possibility that they might differ. In the other, hospitals are strikingly different. In both cases, we assume that patients are drawn from 20 hospitals.

In both examples, we will fit the same model, and we will type the same command to fit it. Below are the same data we have been using but with a new variable, `hospid`, that identifies from which of the 20 hospitals each patient was drawn (and which we have made up):

(Continued on next page)

```
. use http://www.stata-press.com/data/r11/hospid1, clear
. logistic low age lwt i.race smoke ptl ht ui, vce(cluster hospid)
Logistic regression Number of obs = 189
 Wald chi2(8) = 49.67
 Prob > chi2 = 0.0000
Log pseudolikelihood = -100.724 Pseudo R2 = 0.1416
 (Std. Err. adjusted for 20 clusters in hospid)
```

| low   | Odds Ratio | Robust Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-------|------------|------------------|-------|-------|----------------------|----------|
| age   | .9732636   | .0397476         | -0.66 | 0.507 | .898396              | 1.05437  |
| lwt   | .9849634   | .0057101         | -2.61 | 0.009 | .9738352             | .9962187 |
| race  |            |                  |       |       |                      |          |
| 2     | 3.534767   | 2.013285         | 2.22  | 0.027 | 1.157563             | 10.79386 |
| 3     | 2.368079   | .8451325         | 2.42  | 0.016 | 1.176562             | 4.766257 |
| smoke | 2.517698   | .8284259         | 2.81  | 0.005 | 1.321062             | 4.79826  |
| ptl   | 1.719161   | .6676221         | 1.40  | 0.163 | .8030814             | 3.680219 |
| ht    | 6.249602   | 4.066275         | 2.82  | 0.005 | 1.74591              | 22.37086 |
| ui    | 2.1351     | 1.093144         | 1.48  | 0.138 | .7827337             | 5.824014 |

The standard errors are similar to the standard errors we have previously obtained, whether we used the robust or conventional estimators. In this example, we invented the hospital IDs randomly.

Here are the results of the estimation with the same data but with a different set of hospital IDs:

```
. use http://www.stata-press.com/data/r11/hospid2
. logistic low age lwt i.race smoke ptl ht ui, vce(cluster hospid)
Logistic regression Number of obs = 189
 Wald chi2(8) = 7.19
 Prob > chi2 = 0.5167
Log pseudolikelihood = -100.724 Pseudo R2 = 0.1416
 (Std. Err. adjusted for 20 clusters in hospid)
```

| low   | Odds Ratio | Robust Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-------|------------|------------------|-------|-------|----------------------|----------|
| age   | .9732636   | .0293064         | -0.90 | 0.368 | .9174862             | 1.032432 |
| lwt   | .9849634   | .0106123         | -1.41 | 0.160 | .9643817             | 1.005984 |
| race  |            |                  |       |       |                      |          |
| 2     | 3.534767   | 3.120338         | 1.43  | 0.153 | .6265521             | 19.9418  |
| 3     | 2.368079   | 1.297738         | 1.57  | 0.116 | .8089594             | 6.932114 |
| smoke | 2.517698   | 1.570287         | 1.48  | 0.139 | .7414969             | 8.548655 |
| ptl   | 1.719161   | .6799153         | 1.37  | 0.171 | .7919045             | 3.732161 |
| ht    | 6.249602   | 7.165454         | 1.60  | 0.110 | .660558              | 59.12808 |
| ui    | 2.1351     | 1.411977         | 1.15  | 0.251 | .5841231             | 7.804266 |

Note the strikingly larger standard errors. What happened? In these data, women most likely to have low-birthweight babies are sent to certain hospitals, and the decision on likeliness is based not just on age, smoking history, etc., but on other things that doctors can see but that are not recorded in our data. Thus merely because a woman is at one of the centers identifies her to be more likely to have a low-birthweight baby.

## Saved results

`logistic` saves the following in `e()`:

### Scalars

|                            |                                                |
|----------------------------|------------------------------------------------|
| <code>e(N)</code>          | number of observations                         |
| <code>e(N_cds)</code>      | number of completely determined successes      |
| <code>e(N_cdf)</code>      | number of completely determined failures       |
| <code>e(k)</code>          | number of parameters                           |
| <code>e(k_eq)</code>       | number of equations                            |
| <code>e(k_eq_model)</code> | number of equations in model Wald test         |
| <code>e(k_dv)</code>       | number of dependent variables                  |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints |
| <code>e(df_m)</code>       | model degrees of freedom                       |
| <code>e(r2_p)</code>       | pseudo- <i>R</i> -squared                      |
| <code>e(ll)</code>         | log likelihood                                 |
| <code>e(ll_0)</code>       | log likelihood, constant-only model            |
| <code>e(N_clust)</code>    | number of clusters                             |
| <code>e(chi2)</code>       | $\chi^2$                                       |
| <code>e(p)</code>          | significance of model test                     |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                      |
| <code>e(ic)</code>         | number of iterations                           |
| <code>e(rc)</code>         | return code                                    |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                    |

### Macros

|                                 |                                                                          |
|---------------------------------|--------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>logistic</code>                                                    |
| <code>e(cmdline)</code>         | command as typed                                                         |
| <code>e(depvar)</code>          | name of dependent variable                                               |
| <code>e(wtype)</code>           | weight type                                                              |
| <code>e(wexp)</code>            | weight expression                                                        |
| <code>e(title)</code>           | title in estimation output                                               |
| <code>e(clustvar)</code>        | name of cluster variable                                                 |
| <code>e(offset)</code>          | offset                                                                   |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                  |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                           |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                            |
| <code>e(opt)</code>             | type of optimization                                                     |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>       | type of ml method                                                        |
| <code>e(user)</code>            | name of likelihood-evaluator program                                     |
| <code>e(technique)</code>       | maximization technique                                                   |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular              |
| <code>e(critttype)</code>       | optimization criterion                                                   |
| <code>e(properties)</code>      | b V                                                                      |
| <code>e(estat_cmd)</code>       | program used to implement <code>estat</code>                             |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                           |
| <code>e(marginsnotok)</code>    | predictions disallowed by <code>margins</code>                           |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>           |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>           |

|                 |                                              |
|-----------------|----------------------------------------------|
| Matrices        |                                              |
| e(b)            | coefficient vector                           |
| e(Cns)          | constraints matrix                           |
| e(ilog)         | iteration log (up to 20 iterations)          |
| e(gradient)     | gradient vector                              |
| e(mns)          | vector of means of the independent variables |
| e(rules)        | information about perfect predictors         |
| e(V)            | variance–covariance matrix of the estimators |
| e(V_modelbased) | model-based variance                         |
| Functions       |                                              |
| e(sample)       | marks estimation sample                      |

Methods and formulas

logistic is implemented as an ado-file.

Define  $\mathbf{x}_j$  as the (row) vector of independent variables, augmented by 1, and  $\mathbf{b}$  as the corresponding estimated parameter (column) vector. The logistic regression model is fit by `logit`; see [R] [logit](#) for details of estimation.

The odds ratio corresponding to the  $i$ th coefficient is  $\psi_i = \exp(b_i)$ . The standard error of the odds ratio is  $s_i^{\psi} = \psi_i s_i$ , where  $s_i$  is the standard error of  $b_i$  estimated by `logit`.

Define  $I_j = \mathbf{x}_j \mathbf{b}$  as the predicted index of the  $j$ th observation. The predicted probability of a positive outcome is

$$p_j = \frac{\exp(I_j)}{1 + \exp(I_j)}$$

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

`logistic` also supports estimation with survey data. For details on VCEs with survey data, see [SVY] [variance estimation](#).

References

Archer, K. J., and S. Lemeshow. 2006. [Goodness-of-fit test for a logistic regression model fitted using survey sample data](#). *Stata Journal* 6: 97–105.

Brady, A. R. 1998. [sbe21: Adjusted population attributable fractions from logistic regression](#). *Stata Technical Bulletin* 42: 8–12. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 137–143. College Station, TX: Stata Press.

Cleves, M. A., and A. Tosetto. 2000. [sg139: Logistic regression when binary outcome is measured with uncertainty](#). *Stata Technical Bulletin* 55: 20–23. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 152–156. College Station, TX: Stata Press.

Collett, D. 2003. *Modelling Survival Data in Medical Research*. 2nd ed. London: Chapman & Hall/CRC.

de Irala-Estévez, J., and M. A. Martínez. 2000. [sg125: Automatic estimation of interaction effects and their confidence intervals](#). *Stata Technical Bulletin* 53: 29–31. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 270–273. College Station, TX: Stata Press.

Dupont, W. D. 2009. *Statistical Modeling for Biomedical Researchers: A Simple Introduction to the Analysis of Complex Data*. 2nd ed. Cambridge: Cambridge University Press.

Freese, J. 2002. [Least likely observations in regression models for categorical outcomes](#). *Stata Journal* 2: 296–300.

- Garrett, J. M. 1997. [sbe14: Odds ratios and confidence intervals for logistic regression models with effect modification](#). *Stata Technical Bulletin* 36: 15–22. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 104–114. College Station, TX: Stata Press.
- Gould, W. W. 2000. [sg124: Interpreting logistic regression in all its forms](#). *Stata Technical Bulletin* 53: 19–29. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 257–270. College Station, TX: Stata Press.
- Hilbe, J. M. 1997. [sg63: Logistic regression: Standardized coefficients and partial correlations](#). *Stata Technical Bulletin* 35: 21–22. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 162–163. College Station, TX: Stata Press.
- . 2009. *Logistic Regression Models*. Boca Raton, FL: Chapman & Hill/CRC.
- Hosmer Jr., D. W., and S. Lemeshow. 2000. *Applied Logistic Regression*. 2nd ed. New York: Wiley.
- Kleinbaum, D. G., and M. Klein. 2002. *Logistic Regression: A Self-Learning Text*. 2nd ed. New York: Springer.
- Lemeshow, S., and J.-R. L. Gall. 1994. Modeling the severity of illness of ICU patients: A systems update. *Journal of the American Medical Association* 272: 1049–1055.
- Lemeshow, S., and D. W. Hosmer Jr. 2005. Logistic regression. In Vol. 2 of *Encyclopedia of Biostatistics*, ed. P. Armitage and T. Colton, 2870–2880. Chichester, UK: Wiley.
- Long, J. S., and J. Freese. 2006. *Regression Models for Categorical Dependent Variables Using Stata*. 2nd ed. College Station, TX: Stata Press.
- Miranda, A., and S. Rabe-Hesketh. 2006. Maximum likelihood estimation of endogenous switching and sample selection models for binary, ordinal, and count variables. *Stata Journal* 6: 285–308.
- Mitchell, M. N., and X. Chen. 2005. Visualizing main effects and interactions for binary logit models. *Stata Journal* 5: 64–82.
- Pagano, M., and K. Gauvreau. 2000. *Principles of Biostatistics*. 2nd ed. Belmont, CA: Duxbury.
- Pampel, F. C. 2000. *Logistic Regression: A Primer*. Thousand Oaks, CA: Sage.
- Paul, C. 1998. [sg92: Logistic regression for data including multiple imputations](#). *Stata Technical Bulletin* 45: 28–30. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 180–183. College Station, TX: Stata Press.
- Pearce, M. S. 2000. [sg148: Profile likelihood confidence intervals for explanatory variables in logistic regression](#). *Stata Technical Bulletin* 56: 45–47. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 211–214. College Station, TX: Stata Press.
- Pregibon, D. 1981. Logistic regression diagnostics. *Annals of Statistics* 9: 705–724.
- Reilly, M., and A. Salim. 2000. [sg156: Mean score method for missing covariate data in logistic regression models](#). *Stata Technical Bulletin* 58: 25–27. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 256–258. College Station, TX: Stata Press.
- Schonlau, M. 2005. Boosted regression (boosting): An introductory tutorial and a Stata plugin. *Stata Journal* 5: 330–354.
- Vittinghoff, E., D. V. Glidden, S. C. Shiboski, and C. E. McCulloch. 2005. *Regression Methods in Biostatistics: Linear Logistic, Survival, and Repeated Measures Models*. New York: Springer.
- Xu, J., and J. S. Long. 2005. Confidence intervals for predicted outcomes in regression models for categorical outcomes. *Stata Journal* 5: 537–559.

## Also see

- [R] [logistic postestimation](#) — Postestimation tools for logistic
- [R] [roc](#) — Receiver operating characteristic (ROC) analysis
- [R] [brier](#) — Brier score decomposition
- [R] [exlogistic](#) — Exact logistic regression
- [R] [logit](#) — Logistic regression, reporting coefficients
- [SVY] [svy estimation](#) — Estimation commands for survey data
- [XT] [xtlogit](#) — Fixed-effects, random-effects, and population-averaged logit models
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are of special interest after `logistic`:

| command                 | description                                                                                              |
|-------------------------|----------------------------------------------------------------------------------------------------------|
| <code>estat clas</code> | <code>estat classification</code> reports various summary statistics, including the classification table |
| <code>estat gof</code>  | Pearson or Hosmer–Lemeshow goodness-of-fit test                                                          |
| <code>lroc</code>       | graphs the ROC curve and calculates the area under the curve                                             |
| <code>lsens</code>      | graphs sensitivity and specificity versus probability cutoff                                             |

These commands are not appropriate after the `svy` prefix.

For information about these commands, see below.

The following standard postestimation commands are also available:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>            | link test for model specification                                                                   |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.



## Special-interest postestimation commands

`estat classification` reports various summary statistics, including the classification table.

`estat gof` reports the Pearson goodness-of-fit test or the Hosmer–Lemeshow goodness-of-fit test.

`lroc` graphs the ROC curve and calculates the area under the curve.

`lsens` graphs sensitivity and specificity versus probability cutoff and optionally creates new variables containing these data.

`estat classification`, `estat gof`, `lroc`, and `lsens` produce statistics and graphs either for the estimation sample or for any set of observations. However, they always use the estimation sample by default. When `weights`, `if`, or `in` is used with `logistic`, it is not necessary to repeat the qualifier with these commands when you want statistics computed for the estimation sample. Specify `if`, `in`, or the `all` option only when you want statistics computed for a set of observations other than the estimation sample. Specify `weights` (only `fweights` are allowed with these commands) only when you want to use a different set of weights.

By default, `estat classification`, `estat gof`, `lroc`, and `lsens` use the last model fit by `logistic`. You may also directly specify the model to the `lroc` and `lsens` commands by inputting a vector of coefficients with the `beta()` option and passing the name of the dependent variable *depvar*.

`estat classification` and `estat gof` require that the current estimation results be from `logistic`, `logit`, or `probit`. `lroc` and `lsens` commands may also be used after `logit` or `probit`. `estat classification`, `lroc`, and `lsens` may also be used after `ivprobit`.

## Syntax for predict

```
predict [type] newvar [if] [in] [, statistic nooffset rules asif]
```

| <i>statistic</i>         | description                                                                   |
|--------------------------|-------------------------------------------------------------------------------|
| Main                     |                                                                               |
| <code>pr</code>          | probability of a positive outcome; the default                                |
| <code>xb</code>          | linear prediction                                                             |
| <code>stdp</code>        | standard error of the linear prediction                                       |
| * <code>dbeta</code>     | Pregibon (1981) $\Delta\hat{\beta}$ influence statistic                       |
| * <code>deviance</code>  | deviance residual                                                             |
| * <code>dx2</code>       | Hosmer Jr. and Lemeshow (2000, 174) $\Delta\chi^2$ influence statistic        |
| * <code>ddeviance</code> | Hosmer Jr. and Lemeshow (2000, 174) $\Delta D$ influence statistic            |
| * <code>hat</code>       | Pregibon (1981) leverage                                                      |
| * <code>number</code>    | sequential number of the covariate pattern                                    |
| * <code>residuals</code> | Pearson residuals; adjusted for number sharing covariate pattern              |
| * <code>rstandard</code> | standardized Pearson residuals; adjusted for number sharing covariate pattern |
| <code>score</code>       | first derivative of the log likelihood with respect to $\mathbf{x}_j\beta$    |

Unstarred statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample. Starred statistics are calculated only for the estimation sample, even when `if e(sample)` is not specified.

`pr`, `xb`, `stdp`, and `score` are the only options allowed with `svy` estimation results.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

### Main

**pr**, the default, calculates the probability of a positive outcome.

**xb** calculates the linear prediction.

**stdp** calculates the standard error of the linear prediction.

**dbeta** calculates the [Pregibon \(1981\)](#)  $\Delta\hat{\beta}$  influence statistic, a standardized measure of the difference in the coefficient vector that is due to deletion of the observation along with all others that share the same covariate pattern. In [Hosmer Jr. and Lemeshow \(2000, 144–145\)](#) jargon, this statistic is *M*-asymptotic; that is, it is adjusted for the number of observations that share the same covariate pattern.

**deviance** calculates the deviance residual.

**dx2** calculates the [Hosmer Jr. and Lemeshow \(2000, 174\)](#)  $\Delta\chi^2$  influence statistic, reflecting the decrease in the Pearson  $\chi^2$  because of deletion of the observation and all others that share the same covariate pattern.

**ddeviance** calculates the [Hosmer Jr. and Lemeshow \(2000, 174\)](#)  $\Delta D$  influence statistic, which is the change in the deviance residual that is due to deletion of the observation and all others that share the same covariate pattern.

**hat** calculates the [Pregibon \(1981\)](#) leverage or the diagonal elements of the hat matrix adjusted for the number of observations that share the same covariate pattern.

**number** numbers the covariate patterns—observations with the same covariate pattern have the same **number**. Observations not used in estimation have **number** set to missing. The first covariate pattern is numbered 1, the second 2, and so on.

**residuals** calculates the Pearson residual as given by [Hosmer Jr. and Lemeshow \(2000, 145\)](#) and adjusted for the number of observations that share the same covariate pattern.

**rstandard** calculates the standardized Pearson residual as given by [Hosmer Jr. and Lemeshow \(2000, 173\)](#) and adjusted for the number of observations that share the same covariate pattern.

**score** calculates the equation-level score,  $\partial \ln L / \partial (\mathbf{x}_j \boldsymbol{\beta})$ .

### Options

**nooffset** is relevant only if you specified **offset(varname)** for **logistic**. It modifies the calculations made by **predict** so that they ignore the offset variable; the linear prediction is treated as  $\mathbf{x}_j \mathbf{b}$  rather than as  $\mathbf{x}_j \mathbf{b} + \text{offset}_j$ .

**rules** requests that Stata use any rules that were used to identify the model when making the prediction. By default, Stata calculates missing for excluded observations. See [\[R\] \*\*logit\*\*](#) for an example.

**asif** requests that Stata ignore the rules and the exclusion criteria and calculate predictions for all observations possible by using the estimated parameter from the model. See [\[R\] \*\*logit\*\*](#) for an example.

## Syntax for estat classification

`estat classification [if] [in] [weight] [, class_options]`

| <i>class_options</i> | description |
|----------------------|-------------|
|----------------------|-------------|

---

|                        |                                                                 |
|------------------------|-----------------------------------------------------------------|
| Main                   |                                                                 |
| <code>all</code>       | display summary statistics for all observations in the data     |
| <code>cutoff(#)</code> | positive outcome threshold; default is <code>cutoff(0.5)</code> |

---

`fweights` are allowed; see [\[U\] 11.1.6 weight](#).

## Menu

Statistics > Postestimation > Reports and statistics

## Options for estat classification

Main

`all` requests that the statistic be computed for all observations in the data, ignoring any `if` or `in` restrictions specified by `logistic`.

`cutoff(#)` specifies the value for determining whether an observation has a predicted positive outcome. An observation is classified as positive if its predicted probability is  $\geq \#$ . The default is 0.5.

## Syntax for estat gof

`estat gof [if] [in] [weight] [, gof_options]`

| <i>gof_options</i> | description |
|--------------------|-------------|
|--------------------|-------------|

---

|                        |                                                                             |
|------------------------|-----------------------------------------------------------------------------|
| Main                   |                                                                             |
| <code>group(#)</code>  | perform Hosmer–Lemeshow goodness-of-fit test using <code>#</code> quantiles |
| <code>all</code>       | execute test for all observations in the data                               |
| <code>outsample</code> | adjust degrees of freedom for samples outside estimation sample             |
| <code>table</code>     | display table of groups used for test                                       |

---

`fweights` are allowed; see [\[U\] 11.1.6 weight](#).

## Menu

Statistics > Postestimation > Reports and statistics

# Options for estat gof

Main

`group(#)` specifies the number of quantiles to be used to group the data for the Hosmer–Lemeshow goodness-of-fit test. `group(10)` is typically specified. If this option is not given, the Pearson goodness-of-fit test is computed using the covariate patterns in the data as groups.

`all` requests that the statistic be computed for all observations in the data, ignoring any `if` or `in` restrictions specified with `logistic`.

`outsample` adjusts the degrees of freedom for the Pearson and Hosmer–Lemeshow goodness-of-fit tests for samples outside the estimation sample. See [Samples other than the estimation sample](#) later in this entry.

`table` displays a table of the groups used for the Hosmer–Lemeshow or Pearson goodness-of-fit test with predicted probabilities, observed and expected counts for both outcomes, and totals for each group.

## Syntax for lroc

`lroc` [*depvar*] [*if*] [*in*] [*weight*] [, *lroc\_options*]

| <i>lroc_options</i>                                                        | description                                                                               |
|----------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| Main                                                                       |                                                                                           |
| <code>all</code>                                                           | compute area under ROC curve and graph curve for all observations                         |
| <code>nograph</code>                                                       | suppress graph                                                                            |
| Advanced                                                                   |                                                                                           |
| <code>beta(matname)</code>                                                 | row vector containing coefficients for a logistic model                                   |
| Plot                                                                       |                                                                                           |
| <code>cline_options</code>                                                 | change the look of the line                                                               |
| <code>marker_options</code>                                                | change look of markers (color, size, etc.)                                                |
| <code>marker_label_options</code>                                          | add marker labels; change look or position                                                |
| Reference line                                                             |                                                                                           |
| <code>rlopts(cline_options)</code>                                         | affect rendition of the reference line                                                    |
| Add plots                                                                  |                                                                                           |
| <code>addplot(plot)</code>                                                 | add other plots to the generated graph                                                    |
| Y axis, X axis, Titles, Legend, Overall                                    |                                                                                           |
| <code>twoway_options</code>                                                | any options other than <code>by()</code> documented in <a href="#">[G] twoway_options</a> |
| <code>fweights</code> are allowed; see <a href="#">[U] 11.1.6 weight</a> . |                                                                                           |

## Menu

Statistics > Binary outcomes > Postestimation > ROC curve after logistic/logit/probit/ivprobit

## Options for lroc

### Main

**all** requests that the statistic be computed for all observations in the data, ignoring any **if** or **in** restrictions specified by **logistic**.

**nograph** suppresses graphical output.

### Advanced

**beta**(*matname*) specifies a row vector containing coefficients for a logistic model. The columns of the row vector must be labeled with the corresponding names of the independent variables in the data. The dependent variable *depvar* must be specified immediately after the command name. See *Models other than last fitted model* later in this entry.

### Plot

*cline\_options*, *marker\_options*, and *marker\_label\_options* affect the rendition of the ROC curve—the plotted points connected by lines. These options affect the size and color of markers, whether and how the markers are labeled, and whether and how the points are connected; see [G] [cline\\_options](#), [G] [marker\\_options](#), and [G] [marker\\_label\\_options](#).

### Reference line

**rlopts**(*cline\_options*) affects the rendition of the reference line; see [G] [cline\\_options](#).

### Add plots

**addplot**(*plot*) provides a way to add other plots to the generated graph; see [G] [addplot\\_option](#).

### Y axis, X axis, Titles, Legend, Overall

*twoway\_options* are any of the options documented in [G] [twoway\\_options](#), excluding **by()**. These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

## Syntax for lsens

```
lsens [depvar] [if] [in] [weight] [, lsens_options]
```

(Continued on next page)

| <i>lsens_options</i>                                                       | description                                                                               |
|----------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| Main                                                                       |                                                                                           |
| <code>all</code>                                                           | graph all observations in the data                                                        |
| <code>genprob(<i>varname</i>)</code>                                       | create variable containing probability cutoffs                                            |
| <code>gensens(<i>varname</i>)</code>                                       | create variable containing sensitivity                                                    |
| <code>genspec(<i>varname</i>)</code>                                       | create variable containing specificity                                                    |
| <code>replace</code>                                                       | overwrite existing variables                                                              |
| <code>nograph</code>                                                       | suppress the graph                                                                        |
| Advanced                                                                   |                                                                                           |
| <code>beta(<i>matname</i>)</code>                                          | row vector containing coefficients for the model                                          |
| Plot                                                                       |                                                                                           |
| <code>connect_options</code>                                               | affect rendition of the plotted points connected by lines                                 |
| Add plots                                                                  |                                                                                           |
| <code>addplot(<i>plot</i>)</code>                                          | add other plots to the generated graph                                                    |
| Y axis, X axis, Titles, Legend, Overall                                    |                                                                                           |
| <code>twoway_options</code>                                                | any options other than <code>by()</code> documented in <a href="#">[G] twoway_options</a> |
| <code>fweights</code> are allowed; see <a href="#">[U] 11.1.6 weight</a> . |                                                                                           |

## Menu

Statistics > Binary outcomes > Postestimation > Sensitivity/specificity plot

## Options for lsens

Main

`all` requests that the statistic be computed for all observations in the data, ignoring any `if` or `in` restrictions specified with `logistic`.

`genprob(varname)`, `gensens(varname)`, and `genspec(varname)` specify the names of new variables created to contain, respectively, the probability cutoffs and the corresponding sensitivity and specificity. These new variables will be added to the dataset and, in the process, may result in up to two new observations being added to the dataset in memory. Values are recorded for  $p = 0$ , for each of the observed predicted probabilities, and for  $p = 1$ . The total number of observations required to do this can be fewer than  $\_N$ , the same as  $\_N$ , or  $\_N + 1$ , or  $\_N + 2$ . If more observations are added, they are added at the end of the dataset and the values of the original variables are set to missing in the added observations. How the values added align with existing observations is irrelevant.

`replace` requests that existing variables specified for `genprob()`, `gensens()`, or `genspec()` be overwritten.

`nograph` suppresses graphical output.

## Advanced

`beta(matname)` specifies a row vector containing coefficients for a logistic model. The columns of the row vector must be labeled with the corresponding names of the independent variables in the data. The dependent variable *devar* must be specified immediately after the command name. See *Models other than last fitted model* later in this entry.

## Plot

*connect\_options* affect the rendition of the plotted points connected by lines; see *connect\_options* in [G] [graph twoway scatter](#).

## Add plots

`addplot(plot)` provides a way to add other plots to the generated graph. See [G] [addplot\\_option](#).

## Y axis, X axis, Titles, Legend, Overall

*twoway\_options* are any of the options documented in [G] [twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

## Remarks

Remarks are presented under the following headings:

- predict after logistic*
  - predict without options*
  - predict with the xb and stdp options*
  - predict with the residuals option*
  - predict with the number option*
  - predict with the deviance option*
  - predict with the rstandard option*
  - predict with the hat option*
  - predict with the dx2 option*
  - predict with the ddeviance option*
  - predict with the dbeta option*
- estat classification*
- estat gof*
- lroc*
- lsens*
- Samples other than the estimation sample*
- Models other than the last fitted model*

## predict after logistic

`predict` is used after `logistic` to obtain predicted probabilities, residuals, and influence statistics for the estimation sample. The suggested diagnostic graphs below are from [Hosmer Jr. and Lemeshow \(2000\)](#), where they are more elaborately explained. Also see [Collett \(2003, 129–168\)](#) for a thorough discussion of model checking.

### predict without options

Typing `predict newvar` after estimation calculates the predicted probability of a positive outcome.

We previously ran the model `logistic low age lwt i.race smoke ptl ht ui`. We obtain the predicted probabilities of a positive outcome by typing

```
. use http://www.stata-press.com/data/r11/lbw
(Hosmer & Lemeshow data)

. logistic low age lwt i.race smoke ptl ht ui
(output omitted)
. predict p
(option pr assumed; Pr(low))

. summarize p low
```

| Variable | Obs | Mean     | Std. Dev. | Min      | Max      |
|----------|-----|----------|-----------|----------|----------|
| p        | 189 | .3121693 | .1913915  | .0272559 | .8391283 |
| low      | 189 | .3121693 | .4646093  | 0        | 1        |

predict with the xb and stdp options

`predict` with the `xb` option calculates the linear combination  $x_j\mathbf{b}$ , where  $x_j$  are the independent variables in the  $j$ th observation and  $\mathbf{b}$  is the estimated parameter vector. This is sometimes known as the index function because the cumulative distribution function indexed at this value is the probability of a positive outcome.

With the `stdp` option, `predict` calculates the standard error of the prediction, which is *not* adjusted for replicated covariate patterns in the data. The influence statistics described below are adjusted for replicated covariate patterns in the data.

predict with the residuals option

`predict` can calculate more than predicted probabilities. The Pearson residual is defined as the square root of the contribution of the covariate pattern to the Pearson  $\chi^2$  goodness-of-fit statistic, signed according to whether the observed number of positive responses within the covariate pattern is less than or greater than expected. For instance,

```
. predict r, residuals
. summarize r, detail
```

| Pearson residual |           |           |             |           |  |
|------------------|-----------|-----------|-------------|-----------|--|
| Percentiles      |           | Smallest  |             |           |  |
| 1%               | -1.750923 | -2.283885 |             |           |  |
| 5%               | -1.129907 | -1.750923 |             |           |  |
| 10%              | -.9581174 | -1.636279 | Obs         | 189       |  |
| 25%              | -.6545911 | -1.636279 | Sum of Wgt. | 189       |  |
| 50%              | -.3806923 |           | Mean        | -.0242299 |  |
|                  |           | Largest   | Std. Dev.   | .9970949  |  |
| 75%              | .8162894  | 2.23879   |             |           |  |
| 90%              | 1.510355  | 2.317558  | Variance    | .9941981  |  |
| 95%              | 1.747948  | 3.002206  | Skewness    | .8618271  |  |
| 99%              | 3.002206  | 3.126763  | Kurtosis    | 3.038448  |  |



We notice the prevalence of a few large positive residuals:

```
. sort r
. list id r low p age race in -5/1
```

|      | id | r        | low | p        | age | race  |
|------|----|----------|-----|----------|-----|-------|
| 185. | 33 | 2.224501 | 1   | .1681123 | 19  | white |
| 186. | 57 | 2.23879  | 1   | .166329  | 15  | white |
| 187. | 16 | 2.317558 | 1   | .1569594 | 27  | other |
| 188. | 77 | 3.002206 | 1   | .0998678 | 26  | white |
| 189. | 36 | 3.126763 | 1   | .0927932 | 24  | white |

## predict with the number option

Covariate patterns play an important role in logistic regression. Two observations are said to share the same covariate pattern if the independent variables for the two observations are identical. Although we might think of having individual observations, the statistical information in the sample can be summarized by the covariate patterns, the number of observations with that covariate pattern, and the number of positive outcomes within the pattern. Depending on the model, the number of covariate patterns can approach or be equal to the number of observations, or it can be considerably less.

Stata calculates all the residual and diagnostic statistics in terms of covariate patterns, not observations. That is, all observations with the same covariate pattern are given the same residual and diagnostic statistics. [Hosmer Jr. and Lemeshow \(2000, 145–145\)](#) argue that such “ $M$ -asymptotic” statistics are more useful than “ $N$ -asymptotic” statistics.

To understand the difference, think of an observed positive outcome with predicted probability of 0.8. Taking the observation in isolation, the residual must be positive—we expected 0.8 positive responses and observed 1. This may indeed be the correct residual, but not necessarily. Under the  $M$ -asymptotic definition, we ask how many successes we observed across all observations with this covariate pattern. If that number were, say, six, and there were a total of 10 observations with this covariate pattern, then the residual is negative for the covariate pattern—we expected eight positive outcomes but observed six. `predict` makes this kind of calculation and then attaches the same residual to all observations in the covariate pattern.

Occasionally, you might want to find all observations sharing a covariate pattern. `number` allows you to do this:

```
. predict pattern, number
. summarize pattern
```

| Variable | Obs | Mean    | Std. Dev. | Min | Max |
|----------|-----|---------|-----------|-----|-----|
| pattern  | 189 | 89.2328 | 53.16573  | 1   | 182 |

We previously fit the model `logistic low age lwt i.race smoke ptl ht ui` over 189 observations. There are 182 covariate patterns in our data.

## predict with the deviance option

The deviance residual is defined as the square root of the contribution to the likelihood-ratio test statistic of a saturated model versus the fitted model. It has slightly different properties from the Pearson residual (see [Hosmer Jr. and Lemeshow \[2000, 145–147\]](#)):

```
. predict d, deviance
. summarize d, detail
```

|     |             | deviance residual |                    |
|-----|-------------|-------------------|--------------------|
|     | Percentiles | Smallest          |                    |
| 1%  | -1.843472   | -1.911621         |                    |
| 5%  | -1.33477    | -1.843472         |                    |
| 10% | -1.148316   | -1.843472         | Obs 189            |
| 25% | -.8445325   | -1.674869         | Sum of Wgt. 189    |
| 50% | -.5202702   |                   | Mean -.1228811     |
|     |             | Largest           | Std. Dev. 1.049237 |
| 75% | .9129041    | 1.894089          |                    |
| 90% | 1.541558    | 1.924457          | Variance 1.100898  |
| 95% | 1.673338    | 2.146583          | Skewness .6598857  |
| 99% | 2.146583    | 2.180542          | Kurtosis 2.036938  |

predict with the rstandard option

Pearson residuals do not have a standard deviation equal to 1. `rstandard` generates Pearson residuals normalized to have an *expected* standard deviation equal to 1.

```
. predict rs, rstandard
. summarize r rs
```

| Variable | Obs | Mean      | Std. Dev. | Min       | Max      |
|----------|-----|-----------|-----------|-----------|----------|
| r        | 189 | -.0242299 | .9970949  | -2.283885 | 3.126763 |
| rs       | 189 | -.0279135 | 1.026406  | -2.4478   | 3.149081 |

```
. correlate r rs
(obs=189)
```

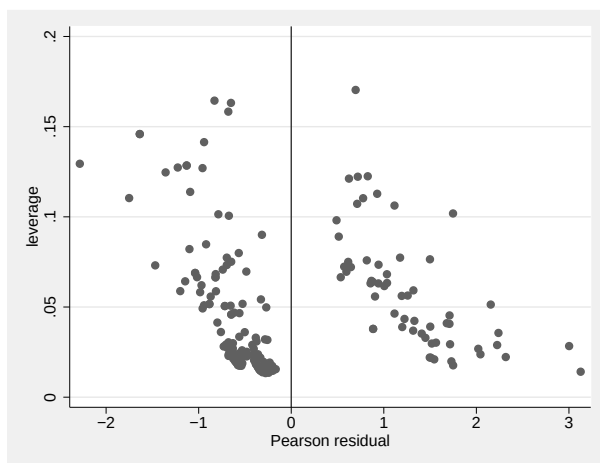
|    | r      | rs     |
|----|--------|--------|
| r  | 1.0000 |        |
| rs | 0.9998 | 1.0000 |

Remember that we previously created `r` containing the (unstandardized) Pearson residuals. In these data, whether we use standardized or unstandardized residuals does not matter much.

predict with the hat option

`hat` calculates the leverage of a covariate pattern—a scaled measure of distance in terms of the independent variables. Large values indicate covariate patterns far from the average covariate pattern that can have a large effect on the fitted model even if the corresponding residual is small. Consider the following graph:

```
. predict h, hat
. scatter h r, xline(0)
```



The points to the left of the vertical line are observed negative outcomes; here our data contain almost as many covariate patterns as observations, so most covariate patterns are unique. In such unique patterns, we observe either 0 or 1 success and expect  $p$ , thus forcing the sign of the residual. If we had fewer covariate patterns—if we did not have continuous variables in our model—there would be no such interpretation, and we would not have drawn the vertical line at 0.

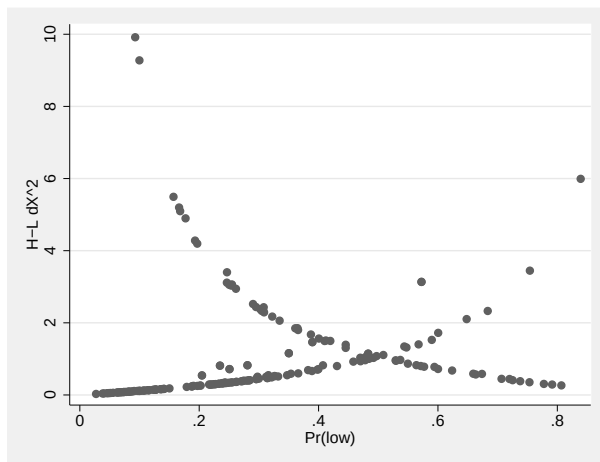
Points on the left and right edges of the graph represent large residuals—covariate patterns that are not fit well by our model. Points at the top of our graph represent high leverage patterns. When analyzing the influence of observations on the model, we are most interested in patterns with high leverage and small residuals—patterns that might otherwise escape our attention.

### predict with the dx2 option

There are many ways to measure influence, and `hat` is one example. `dx2` measures the decrease in the Pearson  $\chi^2$  goodness-of-fit statistic that would be caused by deleting an observation (and all others sharing the covariate pattern):

*(Continued on next page)*

```
. predict dx2, dx2
. scatter dx2 p
```



Paraphrasing [Hosmer Jr. and Lemeshow \(2000, 178–179\)](#), the points going from the top left to the bottom right correspond to covariate patterns with the number of positive outcomes equal to the number in the group; the points on the other curve correspond to 0 positive outcomes. In our data, most of the covariate patterns are unique, so the points tend to lie along one or the other curves; the points that are off the curves correspond to the few repeated covariate patterns in our data in which all the outcomes are not the same.

We examine this graph for large values of `dx2`—there are two at the top left.

### predict with the `ddeviance` option

Another measure of influence is the change in the deviance residuals due to deletion of a covariate pattern:

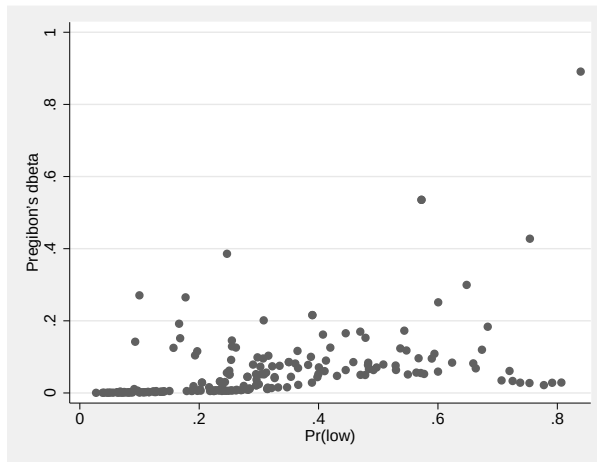
```
. predict dd, ddeviance
```

As with `dx2`, we typically graph `ddeviance` against the probability of a positive outcome. We direct you to [Hosmer Jr. and Lemeshow \(2000, 178\)](#) for an example and for the interpretation of this graph.

### predict with the `dbeta` option

One of the more direct measures of influence of interest to model fitters is the [Pregibon \(1981\)](#) `dbeta` measure, a measure of the change in the coefficient vector that would be caused by deleting an observation (and all others sharing the covariate pattern):

```
. predict db, dbeta
. scatter db p
```



One observation has a large effect on the estimated coefficients. We can easily find this point:

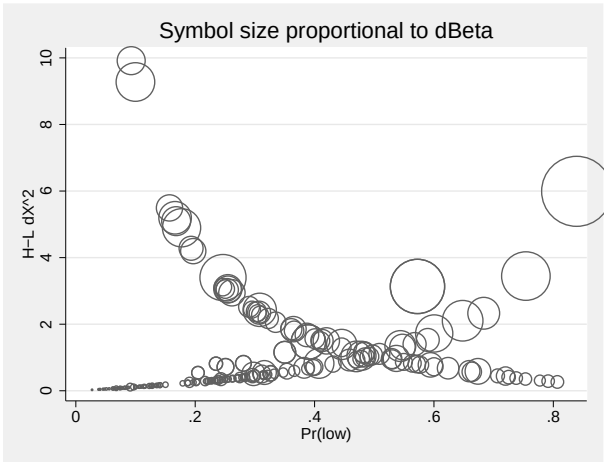
```
. sort db
. list in 1
```

|      |                 |          |                |           |                |            |                |                |               |          |               |
|------|-----------------|----------|----------------|-----------|----------------|------------|----------------|----------------|---------------|----------|---------------|
| 189. | id<br>188       | low<br>0 | age<br>25      | lwt<br>95 | race<br>white  | smoke<br>1 | ptl<br>3       | ht<br>0        | ui<br>1       | ftv<br>0 | bwt<br>3637   |
|      | p<br>.8391283   |          | r<br>-2.283885 |           | pattern<br>117 |            | d<br>-1.911621 |                | rs<br>-2.4478 |          | h<br>.1294439 |
|      | dx2<br>5.991726 |          |                |           | dd<br>4.197658 |            |                | db<br>.8909163 |               |          |               |

Hosmer Jr. and Lemeshow (2000, 180) suggest a graph that combines two of the influence measures:

*(Continued on next page)*

```
. scatter dx2 p [w=db], title("Symbol size proportional to dBeta") mfcolor(none)
(analytic weights assumed)
```



We can easily spot the most influential points by the dbeta and dx2 measures.

estat classification

➤ Example 1

estat classification presents the classification statistics and classification table after logistic.

```
. use http://www.stata-press.com/data/r11/lbw, clear
(Hosmer & Lemeshow data)
. logistic low age lwt i.race smoke ptl ht ui
(output omitted)
. estat classification
Logistic model for low
```

| Classified | True |     | Total |
|------------|------|-----|-------|
|            | D    | ~D  |       |
| +          | 21   | 12  | 33    |
| -          | 38   | 118 | 156   |
| Total      | 59   | 130 | 189   |

Classified + if predicted Pr(D) >= .5  
True D defined as low != 0

|                               |            |        |
|-------------------------------|------------|--------|
| Sensitivity                   | Pr( +  D)  | 35.59% |
| Specificity                   | Pr( -  ~D) | 90.77% |
| Positive predictive value     | Pr( D  +)  | 63.64% |
| Negative predictive value     | Pr( ~D  -) | 75.64% |
| False + rate for true ~D      | Pr( +  ~D) | 9.23%  |
| False - rate for true D       | Pr( -  D)  | 64.41% |
| False + rate for classified + | Pr( ~D  +) | 36.36% |
| False - rate for classified - | Pr( D  -)  | 24.36% |
| Correctly classified          |            | 73.54% |

By default, `estat classification` uses a cutoff of 0.5, although you can vary this with the `cutoff()` option. You can use the `lsens` command to review the potential cutoffs; see *lsens* below.



## estat gof

`estat gof` computes goodness-of-fit tests: either the Pearson  $\chi^2$  test or the Hosmer–Lemeshow test.

By default, `estat classification`, `estat gof`, `lroc`, and `lsens` compute statistics for the estimation sample by using the last model fit by `logistic`. However, samples other than the estimation sample can be specified; see *Samples other than the estimation sample* later in this entry.

### ► Example 2

`estat gof`, typed without options, presents the Pearson  $\chi^2$  goodness-of-fit test for the fitted model. The Pearson  $\chi^2$  goodness-of-fit test is a test of the observed against expected number of responses using cells defined by the covariate patterns; see *predict with the number option* earlier in this entry for the definition of covariate patterns.

```
. estat gof
Logistic model for low, goodness-of-fit test
 number of observations = 189
number of covariate patterns = 182
 Pearson chi2(173) = 179.24
 Prob > chi2 = 0.3567
```

Our model fits reasonably well. However, the number of covariate patterns is close to the number of observations, making the applicability of the Pearson  $\chi^2$  test questionable but not necessarily inappropriate. Hosmer Jr. and Lemeshow (2000, 147–150) suggest regrouping the data by ordering on the predicted probabilities and then forming, say, 10 nearly equal-sized groups. `estat gof` with the `group()` option does this:

```
. estat gof, group(10)
Logistic model for low, goodness-of-fit test
(Table collapsed on quantiles of estimated probabilities)
 number of observations = 189
 number of groups = 10
Hosmer-Lemeshow chi2(8) = 9.65
 Prob > chi2 = 0.2904
```

Again we cannot reject our model. If we specify the `table` option, `estat gof` displays the groups along with the expected and observed number of positive responses (low-birthweight babies):

(Continued on next page)

```
. estat gof, group(10) table
```

**Logistic model for low, goodness-of-fit test**

(Table collapsed on quantiles of estimated probabilities)

| Group | Prob   | Obs_1 | Exp_1 | Obs_0 | Exp_0 | Total |
|-------|--------|-------|-------|-------|-------|-------|
| 1     | 0.0827 | 0     | 1.2   | 19    | 17.8  | 19    |
| 2     | 0.1276 | 2     | 2.0   | 17    | 17.0  | 19    |
| 3     | 0.2015 | 6     | 3.2   | 13    | 15.8  | 19    |
| 4     | 0.2432 | 1     | 4.3   | 18    | 14.7  | 19    |
| 5     | 0.2792 | 7     | 4.9   | 12    | 14.1  | 19    |
| 6     | 0.3138 | 7     | 5.6   | 12    | 13.4  | 19    |
| 7     | 0.3872 | 6     | 6.5   | 13    | 12.5  | 19    |
| 8     | 0.4828 | 7     | 8.2   | 12    | 10.8  | 19    |
| 9     | 0.5941 | 10    | 10.3  | 9     | 8.7   | 19    |
| 10    | 0.8391 | 13    | 12.8  | 5     | 5.2   | 18    |

```
number of observations = 189
number of groups = 10
Hosmer-Lemeshow chi2(8) = 9.65
Prob > chi2 = 0.2904
```



❑ Technical note

`estat gof` with the `group()` option puts all observations with the same predicted probabilities into the same group. If, as in the previous example, we request 10 groups, the groups that `estat gof` makes are  $[p_0, p_{10}]$ ,  $(p_{10}, p_{20}]$ ,  $(p_{20}, p_{30}]$ ,  $\dots$ ,  $(p_{90}, p_{100}]$ , where  $p_k$  is the  $k$ th percentile of the predicted probabilities, with  $p_0$  the minimum and  $p_{100}$  the maximum.

If there are many ties at the quantile boundaries, as will often happen if all independent variables are categorical and there are only a few of them, the sizes of the groups will be uneven. If the totals in some of the groups are small, the  $\chi^2$  statistic for the Hosmer–Lemeshow test may be unreliable. In this case, fewer groups should be specified, or the Pearson goodness-of-fit test may be a better choice.



➤ Example 3

The `table` option can be used without the `group()` option. We would not want to specify this for our current model because there were 182 covariate patterns in the data, caused by including the two continuous variables, `age` and `lwt`, in the model. As an aside, we fit a simpler model and specify `table` with `estat gof`:



```
. logistic low i.race smoke ui
```

```
Logistic regression Number of obs = 189
 LR chi2(4) = 18.80
 Prob > chi2 = 0.0009
Log likelihood = -107.93404 Pseudo R2 = 0.0801
```

|       | low | Odds Ratio | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|-------|-----|------------|-----------|------|-------|----------------------|----------|
| race  |     |            |           |      |       |                      |          |
| 2     |     | 3.052746   | 1.498087  | 2.27 | 0.023 | 1.166747             | 7.987382 |
| 3     |     | 2.922593   | 1.189229  | 2.64 | 0.008 | 1.316457             | 6.488285 |
| smoke |     |            |           |      |       |                      |          |
| ui    |     | 2.945742   | 1.101838  | 2.89 | 0.004 | 1.415167             | 6.131715 |
|       |     | 2.419131   | 1.047359  | 2.04 | 0.041 | 1.035459             | 5.651788 |

```
. estat gof, table
```

**Logistic model for low, goodness-of-fit test**

| Group | Prob   | Obs_1 | Exp_1 | Obs_0 | Exp_0 | Total |
|-------|--------|-------|-------|-------|-------|-------|
| 1     | 0.1230 | 3     | 4.9   | 37    | 35.1  | 40    |
| 2     | 0.2533 | 1     | 1.0   | 3     | 3.0   | 4     |
| 3     | 0.2907 | 16    | 13.7  | 31    | 33.3  | 47    |
| 4     | 0.2923 | 15    | 12.6  | 28    | 30.4  | 43    |
| 5     | 0.2997 | 3     | 3.9   | 10    | 9.1   | 13    |
| 6     | 0.4978 | 4     | 4.0   | 4     | 4.0   | 8     |
| 7     | 0.4998 | 4     | 4.5   | 5     | 4.5   | 9     |
| 8     | 0.5087 | 2     | 1.5   | 1     | 1.5   | 3     |
| 9     | 0.5469 | 2     | 4.4   | 6     | 3.6   | 8     |
| 10    | 0.5577 | 6     | 5.6   | 4     | 4.4   | 10    |
| 11    | 0.7449 | 3     | 3.0   | 1     | 1.0   | 4     |

| Group | Prob   | race  | smoke | ui |
|-------|--------|-------|-------|----|
| 1     | 0.1230 | white | 0     | 0  |
| 2     | 0.2533 | white | 0     | 1  |
| 3     | 0.2907 | other | 0     | 0  |
| 4     | 0.2923 | white | 1     | 0  |
| 5     | 0.2997 | black | 0     | 0  |
| 6     | 0.4978 | other | 0     | 1  |
| 7     | 0.4998 | white | 1     | 1  |
| 8     | 0.5087 | black | 0     | 1  |
| 9     | 0.5469 | other | 1     | 0  |
| 10    | 0.5577 | black | 1     | 0  |
| 11    | 0.7449 | other | 1     | 1  |

```
number of observations = 189
number of covariate patterns = 11
Pearson chi2(6) = 5.71
Prob > chi2 = 0.4569
```

## □ Technical note

`logistic` and `estat gof` keep track of the estimation sample. If you type `logistic ... if x==1`, then when you type `estat gof`, the statistics will be calculated on the `x==1` subsample of the data automatically.

You should specify `if` or `in` with `estat gof` only when you wish to calculate statistics for a set of observations other than the estimation sample. See [Samples other than the estimation sample](#) later in this entry.

If the `logistic` model was fit with `fweights`, `estat gof` properly accounts for the weights in its calculations. (`estat gof` does not allow `pweights`.) You do not have to specify the weights when you run `estat gof`. Weights should be specified with `estat gof` only when you wish to use a different set of weights.

□

## lroc

For other receiver operating characteristic (ROC) commands and a complete description, see [\[R\] roc](#), [\[R\] rocfit](#), and [\[R\] rocfit postestimation](#).

`lroc` graphs the ROC curve—a graph of sensitivity versus one minus specificity as the cutoff  $c$  is varied—and calculates the area under it. Sensitivity is the fraction of observed positive-outcome cases that are correctly classified; specificity is the fraction of observed negative-outcome cases that are correctly classified. When the purpose of the analysis is classification, you must choose a cutoff.

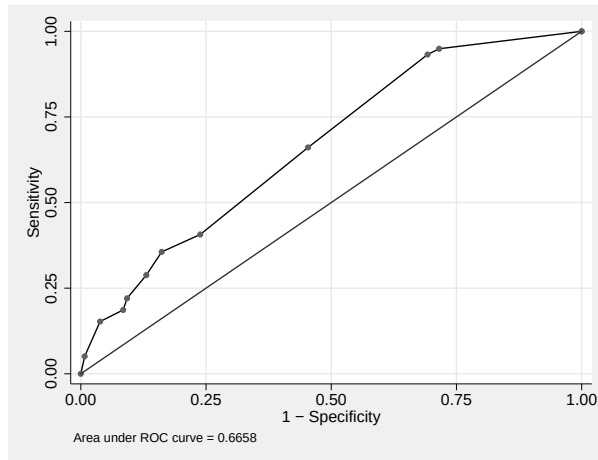
The curve starts at  $(0, 0)$ , corresponding to  $c = 1$ , and continues to  $(1, 1)$ , corresponding to  $c = 0$ . A model with no predictive power would be a  $45^\circ$  line. The greater the predictive power, the more bowed the curve, and hence the area beneath the curve is often used as a measure of the predictive power. A model with no predictive power has area 0.5; a perfect model has area 1.

The ROC curve was first discussed in signal detection theory (Peterson, Birdsall, and Fox 1954) and then was quickly introduced into psychology (Tanner Jr. and Swets 1954). It has since been applied in other fields, particularly medicine (for instance, Metz [1978]). For a classic text on ROC techniques, see Green and Swets (1966).

## ▷ Example 4

ROC curves are typically used when the point of the analysis is classification—which it is not in our low-birthweight model. Nevertheless, the ROC curve is

```
. lroc
Logistic model for low
number of observations = 189
area under ROC curve = 0.6658
```



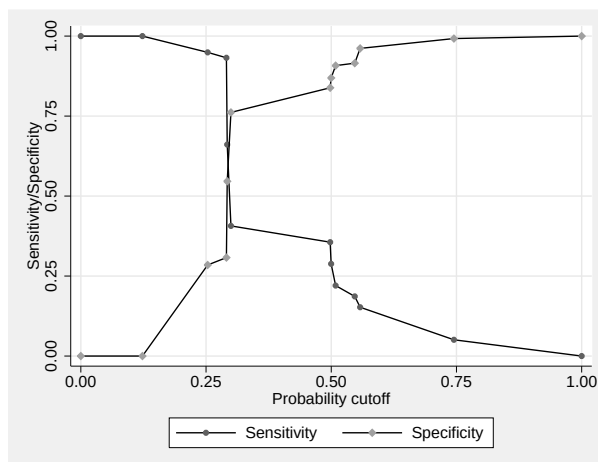
We see that the area under the curve is 0.6658.

4

## lsens

`lsens` also plots sensitivity and specificity; it plots both sensitivity and specificity versus probability cutoff  $c$ . The graph is equivalent to what you would get from `estat classification` if you varied the cutoff probability from 0 to 1.

```
. lsens
```



`lsens` optionally creates new variables containing the probability cutoff, sensitivity, and specificity.

```
. lsens, genprob(p) gensens(sens) genspec(spec) nograph
```

The variables created will have  $M + 2$  distinct nonmissing values: one for each of the  $M$  covariate patterns, one for  $c = 0$ , and another for  $c = 1$ .

## Samples other than the estimation sample

`estat gof`, `estat classification`, `lroc`, and `lsens` can be used with samples other than the estimation sample. By default, these commands remember the estimation sample used with the last `logistic` command. To override this, simply use an `if` or `in` restriction to select another set of observations, or specify the `all` option to force the command to use all the observations in the dataset.

If you use `estat gof` with a sample that is completely different from the estimation sample (i.e., no overlap), you should also specify the `outsample` option so that the  $\chi^2$  statistic properly adjusts the degrees of freedom upward. For an overlapping sample, the conservative thing to do is to leave the degrees of freedom the same as they are for the estimation sample.

### ► Example 5

We want to develop a model for predicting low-birthweight babies. One approach would be to divide our data into two groups, a developmental sample and a validation sample. See [Lemeshow and Gall \(1994\)](#) and [Tilford, Roberson, and Fiser \(1995\)](#) for more information on developing prediction models and severity-scoring systems.

We will do this with the low-birthweight data that we considered previously. First, we randomly divide the data into two samples.

```
. use http://www.stata-press.com/data/r11/lbw, clear
(Hosmer & Lemeshow data)
. set seed 1
. gen r = runiform()
. sort r
. gen group = 1 if _n <= _N/2
(95 missing values generated)
. replace group = 2 if group >=.
(95 real changes made)
```

Then we fit a model using the first sample (`group = 1`), which is our developmental sample.

```
. logistic low age lwt i.race smoke ptl ht ui if group==1
Logistic regression Number of obs = 94
 LR chi2(8) = 29.14
 Prob > chi2 = 0.0003
Log likelihood = -44.293342 Pseudo R2 = 0.2475
```

|       | low | Odds Ratio | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-------|-----|------------|-----------|-------|-------|----------------------|----------|
| age   |     | .91542     | .0553937  | -1.46 | 0.144 | .8130414             | 1.03069  |
| lwt   |     | .9744276   | .0112295  | -2.25 | 0.025 | .9526649             | .9966874 |
| race  |     |            |           |       |       |                      |          |
| 2     |     | 5.063678   | 3.78442   | 2.17  | 0.030 | 1.170327             | 21.90913 |
| 3     |     | 2.606209   | 1.657608  | 1.51  | 0.132 | .7492483             | 9.065522 |
| smoke |     | .909912    | .5252898  | -0.16 | 0.870 | .2934966             | 2.820953 |
| ptl   |     | 3.033543   | 1.507048  | 2.23  | 0.025 | 1.145718             | 8.03198  |
| ht    |     | 21.07656   | 22.64788  | 2.84  | 0.005 | 2.565304             | 173.1652 |
| ui    |     | .988479    | .6699458  | -0.02 | 0.986 | .2618557             | 3.731409 |

To test calibration in the developmental sample, we calculate the Hosmer–Lemeshow goodness-of-fit test by using `estat gof`.

```
. estat gof, group(10)
Logistic model for low, goodness-of-fit test
 (Table collapsed on quantiles of estimated probabilities)
 number of observations = 94
 number of groups = 10
 Hosmer-Lemeshow chi2(8) = 6.67
 Prob > chi2 = 0.5721
```

We did not specify an `if` statement with `estat gof` because we wanted to use the estimation sample. Because the test is not significant, we are satisfied with the fit of our model.

Running `lroc` gives a measure of the discrimination:

```
. lroc, nograph
Logistic model for low
number of observations = 94
area under ROC curve = 0.8156
```

Now we test the calibration of our model by performing a goodness-of-fit test on the validation sample. We specify the `outsample` option so that the number of degrees of freedom is 10 rather than 8.

(Continued on next page)

```
. estat gof if group==2, group(10) table outsample
Logistic model for low, goodness-of-fit test
(Table collapsed on quantiles of estimated probabilities)
```

| Group | Prob   | Obs_1 | Exp_1 | Obs_0 | Exp_0 | Total |
|-------|--------|-------|-------|-------|-------|-------|
| 1     | 0.0725 | 1     | 0.4   | 9     | 9.6   | 10    |
| 2     | 0.1202 | 4     | 0.8   | 5     | 8.2   | 9     |
| 3     | 0.1549 | 3     | 1.3   | 7     | 8.7   | 10    |
| 4     | 0.1888 | 1     | 1.5   | 8     | 7.5   | 9     |
| 5     | 0.2609 | 3     | 2.2   | 7     | 7.8   | 10    |
| 6     | 0.3258 | 4     | 2.7   | 5     | 6.3   | 9     |
| 7     | 0.4217 | 2     | 3.7   | 8     | 6.3   | 10    |
| 8     | 0.4915 | 3     | 4.1   | 6     | 4.9   | 9     |
| 9     | 0.6265 | 4     | 5.5   | 6     | 4.5   | 10    |
| 10    | 0.9737 | 4     | 7.1   | 5     | 1.9   | 9     |

```
number of observations = 95
number of groups = 10
Hosmer-Lemeshow chi2(10) = 28.03
Prob > chi2 = 0.0018
```

We must acknowledge that our model does not fit well on the validation sample. The model’s discrimination in the validation sample is appreciably lower, as well.

```
. lroc if group==2, nograph
Logistic model for low
number of observations = 95
area under ROC curve = 0.5839
```



Models other than the last fitted model

By default, estat classification, estat gof, lroc, and lsens use the last model fit by logistic. You may also directly specify the model to lroc and lsens by inputting a vector of coefficients with the beta() option and passing the name of the dependent variable *depvar* to these commands.

Example 6

Suppose that someone publishes the following logistic model of low birthweight:

$$\text{Pr}(\text{low} = 1) = F(-0.02 \text{ age} - 0.01 \text{ lwt} + 1.3 \text{ black} + 1.1 \text{ smoke} + 0.5 \text{ ptl} + 1.8 \text{ ht} + 0.8 \text{ ui} + 0.5)$$

where *F* is the cumulative logistic distribution. These coefficients are not odds ratios; they are the equivalent of what logit produces.

We can see whether this model fits our data. First we enter the coefficients as a row vector and label its columns with the names of the independent variables plus *\_cons* for the constant (see [P] matrix define and [P] matrix rownames).

```
. use http://www.stata-press.com/data/r11/lbw3, clear
(Hosmer & Lemeshow data)
. matrix input b = (-.02, -.01, 1.3, 1.1, .5, 1.8, .8, .5)
. matrix colnames b = age lwt black smoke ptl ht ui _cons
```

Here we use `lroc` to examine the predictive ability of the model:

```
. lroc low, beta(b) nograph
Logistic model for low
number of observations = 189
area under ROC curve = 0.7275
```

The area under the curve indicates that this model does have some predictive power. We could also obtain a graph of sensitivity and specificity as a function of the cutoff probability by typing

```
. lsens low, beta(b)
```

◀

## Saved results

`estat classification` saves the following in `r()`:

Scalars

|                        |                                         |                      |                                               |
|------------------------|-----------------------------------------|----------------------|-----------------------------------------------|
| <code>r(P_corr)</code> | percent correctly classified            | <code>r(P_1p)</code> | positive predictive value                     |
| <code>r(P_p1)</code>   | sensitivity                             | <code>r(P_0n)</code> | negative predictive value                     |
| <code>r(P_n0)</code>   | specificity                             | <code>r(P_0p)</code> | false-positive rate given classified positive |
| <code>r(P_p0)</code>   | false-positive rate given true negative | <code>r(P_1n)</code> | false-negative rate given classified negative |
| <code>r(P_n1)</code>   | false-negative rate given true positive |                      |                                               |

`estat gof` saves the following in `r()`:

Scalars

|                   |                                        |                      |                    |
|-------------------|----------------------------------------|----------------------|--------------------|
| <code>r(N)</code> | number of observations                 | <code>r(df)</code>   | degrees of freedom |
| <code>r(m)</code> | number of covariate patterns or groups | <code>r(chi2)</code> | $\chi^2$           |

`lroc` saves the following in `r()`:

Scalars

|                   |                        |                      |                          |
|-------------------|------------------------|----------------------|--------------------------|
| <code>r(N)</code> | number of observations | <code>r(area)</code> | area under the ROC curve |
|-------------------|------------------------|----------------------|--------------------------|

`lsens` saves the following in `r()`:

Scalars

|                   |                        |
|-------------------|------------------------|
| <code>r(N)</code> | number of observations |
|-------------------|------------------------|

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

Methods and formulas are presented under the following headings:

*estat gof*  
*predict after logistic*  
*estat classification and lsens*  
*lroc*

## estat gof

Let  $M$  be the total number of covariate patterns among the  $N$  observations. View the data as collapsed on covariate patterns  $j = 1, 2, \dots, M$ , and define  $m_j$  as the total number of observations having covariate pattern  $j$  and  $y_j$  as the total number of positive responses among observations with covariate pattern  $j$ . Define  $p_j$  as the predicted probability of a positive outcome in covariate pattern  $j$ .

The Pearson  $\chi^2$  goodness-of-fit statistic is

$$\chi^2 = \sum_{j=1}^M \frac{(y_j - m_j p_j)^2}{m_j p_j (1 - p_j)}$$

This  $\chi^2$  statistic has approximately  $M - k$  degrees of freedom for the estimation sample, where  $k$  is the number of independent variables, including the constant. For a sample outside the estimation sample, the statistic has  $M$  degrees of freedom.

The Hosmer–Lemeshow goodness-of-fit  $\chi^2$  (Hosmer Jr. and Lemeshow 1980; Lemeshow and Hosmer Jr. 1982; Hosmer Jr., Lemeshow, and Klar 1988) is calculated similarly, except that rather than using the  $M$  covariate patterns as the group definition, the quantiles of the predicted probabilities are used to form groups. Let  $G = \#$  be the number of quantiles requested with `group(#)`. The smallest index  $1 \leq q(i) \leq M$ , such that

$$W_{q(i)} = \sum_{j=1}^{q(i)} m_j \geq \frac{N}{G}$$

gives  $p_{q(i)}$  as the upper boundary of the  $i$ th quantile for  $i = 1, 2, \dots, G$ . Let  $q(0) = 1$  denote the first index.

The groups are then

$$[p_{q(0)}, p_{q(1)}], (p_{q(1)}, p_{q(2)}], \dots, (p_{q(G-1)}, p_{q(G)})]$$

If the `table` option is given, the upper boundaries  $p_{q(1)}, \dots, p_{q(G)}$  of the groups appear next to the group number on the output.

The resulting  $\chi^2$  statistic has approximately  $G - 2$  degrees of freedom for the estimation sample. For a sample outside the estimation sample, the statistic has  $G$  degrees of freedom.

## predict after logistic

Index  $j$  will now be used to index observations, not covariate patterns. Define  $M_j$  for each observation as the total number of observations sharing  $j$ 's covariate pattern. Define  $Y_j$  as the total number of positive responses among observations sharing  $j$ 's covariate pattern.

The Pearson residual for the  $j$ th observation is defined as

$$r_j = \frac{Y_j - M_j p_j}{\sqrt{M_j p_j (1 - p_j)}}$$

For  $M_j > 1$ , the deviance residual  $d_j$  is defined as

$$d_j = \pm \left( 2 \left[ Y_j \ln \left( \frac{Y_j}{M_j p_j} \right) + (M_j - Y_j) \ln \left\{ \frac{M_j - Y_j}{M_j (1 - p_j)} \right\} \right] \right)^{1/2}$$

where the sign is the same as the sign of  $(Y_j - M_j p_j)$ . In the limiting cases, the deviance residual is given by



$$d_j = \begin{cases} -\sqrt{2M_j|\ln(1-p_j)|} & \text{if } Y_j = 0 \\ \sqrt{2M_j|\ln p_j|} & \text{if } Y_j = M_j \end{cases}$$

The *unadjusted* diagonal elements of the hat matrix  $h_{Uj}$  are given by  $h_{Uj} = (\mathbf{XVX}')_{jj}$ , where  $V$  is the estimated covariance matrix of parameters. The adjusted diagonal elements  $h_j$  created by `hat` are then  $h_j = M_j p_j (1 - p_j) h_{Uj}$ .

The standardized Pearson residual  $r_{Sj}$  is  $r_j / \sqrt{1 - h_j}$ .

The Pregibon (1981)  $\Delta\hat{\beta}_j$  influence statistic is

$$\Delta\hat{\beta}_j = \frac{r_j^2 h_j}{(1 - h_j)^2}$$

The corresponding change in the Pearson  $\chi^2$  is  $r_{Sj}^2$ . The corresponding change in the deviance residual is  $\Delta D_j = d_j^2 / (1 - h_j)$ .

## estat classification and lsens

Again let  $j$  index observations. Define  $c$  as the `cutoff()` specified by the user or, if not specified, as 0.5. Let  $p_j$  be the predicted probability of a positive outcome and  $y_j$  be the actual outcome, which we will treat as 0 or 1, although Stata treats it as 0 and non-0, excluding missing observations.

A prediction is classified as *positive* if  $p_j \geq c$  and otherwise is classified as *negative*. The classification is *correct* if it is *positive* and  $y_j = 1$  or if it is *negative* and  $y_j = 0$ .

*Sensitivity* is the fraction of  $y_j = 1$  observations that are correctly classified. *Specificity* is the percentage of  $y_j = 0$  observations that are correctly classified.

## lroc

The ROC curve is a graph of *specificity* against  $(1 - \text{sensitivity})$ . This is guaranteed to be a monotone nondecreasing function because the number of correctly predicted successes increases and the number of correctly predicted failures decreases as the classification cutoff  $c$  decreases.

The area under the ROC curve is the area on the bottom of this graph and is determined by integrating the curve. The vertices of the curve are determined by sorting the data according to the predicted index, and the integral is computed using the trapezoidal rule.

## References

- Archer, K. J., and S. Lemeshow. 2006. [Goodness-of-fit test for a logistic regression model fitted using survey sample data](#). *Stata Journal* 6: 97–105.
- Collett, D. 2003. *Modelling Survival Data in Medical Research*. 2nd ed. London: Chapman & Hall/CRC.
- Garrett, J. M. 2000. [sg157: Predicted values calculated from linear or logistic regression models](#). *Stata Technical Bulletin* 58: 27–30. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 258–261. College Station, TX: Stata Press.
- Green, D. M., and J. A. Swets. 1966. *Signal Detection Theory and Psychophysics*. New York: Wiley.
- Hosmer Jr., D. W., and S. Lemeshow. 1980. Goodness-of-fit tests for the multiple logistic regression model. *Communications in Statistics A9*: 1043–1069.

- . 2000. *Applied Logistic Regression*. 2nd ed. New York: Wiley.
- Hosmer Jr., D. W., S. Lemeshow, and J. Klar. 1988. Goodness-of-fit testing for the logistic regression model when the estimated probabilities are small. *Biometrical Journal* 30: 911–924.
- Lemeshow, S., and J.-R. L. Gall. 1994. Modeling the severity of illness of ICU patients: A systems update. *Journal of the American Medical Association* 272: 1049–1055.
- Lemeshow, S., and D. W. Hosmer Jr. 1982. A review of goodness of fit statistics for the use in the development of logistic regression models. *American Journal of Epidemiology* 115: 92–106.
- Metz, C. E. 1978. Basic principles of ROC analysis. *Seminars in Nuclear Medicine* 8: 283–298.
- Mitchell, M. N., and X. Chen. 2005. [Visualizing main effects and interactions for binary logit models](#). *Stata Journal* 5: 64–82.
- Peterson, W. W., T. G. Birdsall, and W. C. Fox. 1954. The theory of signal detectability. *Transactions IRE Professional Group on Information Theory* PGIT-4: 171–212.
- Pregibon, D. 1981. Logistic regression diagnostics. *Annals of Statistics* 9: 705–724.
- Seed, P. T., and A. Tobías. 2001. [sbe36.1: Summary statistics for diagnostic tests](#). *Stata Technical Bulletin* 59: 25–27. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 90–93. College Station, TX: Stata Press.
- Tanner Jr., W. P., and J. A. Swets. 1954. A decision-making theory of visual detection. *Psychological Review* 61: 401–409.
- Tilford, J. M., P. K. Roberson, and D. H. Fiser. 1995. [sbe12: Using lfit and lroc to evaluate mortality prediction models](#). *Stata Technical Bulletin* 28: 14–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 77–81. College Station, TX: Stata Press.
- Tobías, A. 2000. [sbe36: Summary statistics report for diagnostic tests](#). *Stata Technical Bulletin* 56: 16–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 87–90. College Station, TX: Stata Press.
- Tobías, A., and M. J. Campbell. 1998. [sg90: Akaike's information criterion and Schwarz's criterion](#). *Stata Technical Bulletin* 45: 23–25. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 174–177. College Station, TX: Stata Press.
- Wang, Z. 2007. [Two postestimation commands for assessing confounding effects in epidemiological studies](#). *Stata Journal* 7: 183–196.
- Weesie, J. 1997. [sg66: Enhancements to the alpha command](#). *Stata Technical Bulletin* 35: 32–34. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 176–179. College Station, TX: Stata Press.

## Also see

[R] [logistic](#) — Logistic regression, reporting odds ratios

[U] [20 Estimation and postestimation commands](#)

Syntax

```
logit depvar [indepvars] [if] [in] [weight] [, options]
```

| <i>options</i>                               | description                                                                                                                                              |
|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                        |                                                                                                                                                          |
| <code>noconstant</code>                      | suppress constant term                                                                                                                                   |
| <code>offset(<i>varname</i>)</code>          | include <i>varname</i> in model with coefficient constrained to 1                                                                                        |
| <code>asis</code>                            | retain perfect predictor variables                                                                                                                       |
| <code>constraints(<i>constraints</i>)</code> | apply specified linear constraints                                                                                                                       |
| <code>collinear</code>                       | keep collinear variables                                                                                                                                 |
| SE/Robust                                    |                                                                                                                                                          |
| <code>vce(<i>vcetype</i>)</code>             | <i>vcetype</i> may be <code>oim</code> , <code>robust</code> , <code>cluster <i>clustvar</i></code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                                    |                                                                                                                                                          |
| <code>level(#)</code>                        | set confidence level; default is <code>level(95)</code>                                                                                                  |
| <code>or</code>                              | report odds ratios                                                                                                                                       |
| <code>nocnsreport</code>                     | do not display constraints                                                                                                                               |
| <code>display_options</code>                 | control spacing and display of omitted variables and base and empty cells                                                                                |
| Maximization                                 |                                                                                                                                                          |
| <code>maximize_options</code>                | control the maximization process; seldom used                                                                                                            |
| <sup>†</sup> <code>nocoeff</code>            | do not display coefficient table; seldom used                                                                                                            |
| <sup>†</sup> <code>coeflegend</code>         | display coefficients' legend instead of coefficient table                                                                                                |

<sup>†</sup>`nocoeff` and `coeflegend` do not appear in the dialog box.

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

`bootstrap`, `by`, `fracpoly`, `jackknife`, `mfp`, `mi estimate`, `nestreg`, `rolling`, `statsby`, `stepwise`, and `svy` are allowed; see [U] 11.1.10 Prefix commands.

`vce(bootstrap)` and `vce(jackknife)` are not allowed with the `mi estimate` prefix.

Weights are not allowed with the `bootstrap` prefix.

`vce()`, `nocoeff`, and `weights` are not allowed with the `svy` prefix.

`fweights`, `iweights`, and `pweights` are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Binary outcomes > Logistic regression

## Description

`logit` fits a maximum-likelihood logit model. `depvar = 0` indicates a negative outcome; `depvar ≠ 0` and `depvar ≠ .` (typically `depvar = 1`) indicate a positive outcome.

Also see [R] [logistic](#); `logistic` displays estimates as odds ratios. Many users prefer the `logistic` command to `logit`. Results are the same regardless of which you use—both are the maximum-likelihood estimator. Several auxiliary commands that can be run after `logit`, `probit`, or `logistic` estimation are described in [R] [logistic postestimation](#). A list of related estimation commands is given in [R] [logistic](#).

If estimating on grouped data, see [R] [glogit](#).

## Options

### Model

`noconstant`, `offset(varname)`, `constraints(constraints)`, `collinear`; see [R] [estimation options](#).

`asis` forces retention of perfect predictor variables and their associated perfectly predicted observations and may produce instabilities in maximization; see [R] [probit](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

### Reporting

`level(#)`; see [R] [estimation options](#).

`or` reports the estimated coefficients transformed to odds ratios, i.e.,  $e^b$  rather than  $b$ . Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated. `or` may be specified at estimation or when replaying previously estimated results.

`nocnsreport`; see [R] [estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrntolerance`, `from(init_specs)`; see [R] [maximize](#). These options are seldom used.

The following options are available with `logit` but are not shown in the dialog box:

`nocoef` specifies that the coefficient table not be displayed. This option is sometimes used by program writers but is of no use interactively.

`coeflegend`; see [R] [estimation options](#).

## Remarks

Remarks are presented under the following headings:

*Basic usage*

*Model identification*

## Basic usage

logit fits maximum likelihood models with dichotomous dependent (left-hand-side) variables coded as 0/1 (or, more precisely, coded as 0 and not-0).

### ► Example 1

We have data on the make, weight, and mileage rating of 22 foreign and 52 domestic automobiles. We wish to fit a logit model explaining whether a car is foreign on the basis of its weight and mileage. Here is an overview of our data:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. keep make mpg weight foreign
. describe
Contains data from http://www.stata-press.com/data/r11/auto.dta
 obs: 74 1978 Automobile Data
 vars: 4 13 Apr 2009 17:45
 size: 1,998 (99.9% of memory free) (_dta has notes)
```

| variable name | storage<br>type | display<br>format | value<br>label | variable label |
|---------------|-----------------|-------------------|----------------|----------------|
| make          | str18           | %-18s             |                | Make and Model |
| mpg           | int             | %8.0g             |                | Mileage (mpg)  |
| weight        | int             | %8.0gc            |                | Weight (lbs.)  |
| foreign       | byte            | %8.0g             | origin         | Car type       |

Sorted by: foreign

Note: dataset has changed since last saved

```
. inspect foreign
```

|     |          | Number of Observations |          |             |
|-----|----------|------------------------|----------|-------------|
|     |          | Total                  | Integers | Nonintegers |
| #   | Negative | -                      | -        | -           |
| #   | Zero     | 52                     | 52       | -           |
| #   | Positive | 22                     | 22       | -           |
| #   |          |                        |          |             |
| # # | Total    | 74                     | 74       | -           |
| # # | Missing  | -                      |          |             |
|     |          | 74                     |          |             |
| 0   | 1        |                        |          |             |

(2 unique values)

foreign is labeled and all values are documented in the label.

The variable `foreign` takes on two unique values, 0 and 1. The value 0 denotes a domestic car, and 1 denotes a foreign car.

The model that we wish to fit is

$$\Pr(\text{foreign} = 1) = F(\beta_0 + \beta_1 \text{weight} + \beta_2 \text{mpg})$$

where  $F(z) = e^z / (1 + e^z)$  is the cumulative logistic distribution.

To fit this model, we type

```
. logit foreign weight mpg
Iteration 0: log likelihood = -45.03321
Iteration 1: log likelihood = -29.238536
Iteration 2: log likelihood = -27.244139
Iteration 3: log likelihood = -27.175277
Iteration 4: log likelihood = -27.175156
Iteration 5: log likelihood = -27.175156

Logistic regression Number of obs = 74
 LR chi2(2) = 35.72
 Prob > chi2 = 0.0000
 Pseudo R2 = 0.3966

Log likelihood = -27.175156
```

| foreign | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|---------|-----------|-----------|-------|-------|----------------------|----------|
| weight  | -.0039067 | .0010116  | -3.86 | 0.000 | -.0058894            | -.001924 |
| mpg     | -.1685869 | .0919175  | -1.83 | 0.067 | -.3487418            | .011568  |
| _cons   | 13.70837  | 4.518709  | 3.03  | 0.002 | 4.851859             | 22.56487 |

We find that heavier cars are less likely to be foreign and that cars yielding better gas mileage are also less likely to be foreign, at least holding the weight of the car constant.

□ Technical note

Stata interprets a value of 0 as a negative outcome (failure) and treats all other values (except missing) as positive outcomes (successes). Thus if your dependent variable takes on the values 0 and 1, then 0 is interpreted as failure and 1 as success. If your dependent variable takes on the values 0, 1, and 2, then 0 is still interpreted as failure, but both 1 and 2 are treated as successes.

If you prefer a more formal mathematical statement, when you type `logit y x`, Stata fits the model

$$\Pr(y_j \neq 0 \mid \mathbf{x}_j) = \frac{\exp(\mathbf{x}_j\boldsymbol{\beta})}{1 + \exp(\mathbf{x}_j\boldsymbol{\beta})}$$

Model identification

The `logit` command has one more feature, and it is probably the most useful. `logit` automatically checks the model for identification and, if it is underidentified, drops whatever variables and observations are necessary for estimation to proceed. (`logistic`, `probit`, and `ivprobit` do this as well.)

► Example 2

Have you ever fit a logit model where one or more of your independent variables perfectly predicted one or the other outcome?

For instance, consider the following data:

| Outcome $y$ | Independent variable $x$ |
|-------------|--------------------------|
| 0           | 1                        |
| 0           | 1                        |
| 0           | 0                        |
| 1           | 0                        |

Say that we wish to predict the outcome on the basis of the independent variable. The outcome is always zero whenever the independent variable is one. In our data,  $\Pr(y = 0 \mid x = 1) = 1$ , which means that the logit coefficient on  $x$  must be minus infinity with a corresponding infinite standard error. At this point, you may suspect that we have a problem.

Unfortunately, not all such problems are so easily detected, especially if you have a lot of independent variables in your model. If you have ever had such difficulties, you have experienced one of the more unpleasant aspects of computer optimization. The computer has no idea that it is trying to solve for an infinite coefficient as it begins its iterative process. All it knows is that at each step, making the coefficient a little bigger, or a little smaller, works wonders. It continues on its merry way until either 1) the whole thing comes crashing to the ground when a numerical overflow error occurs or 2) it reaches some predetermined cutoff that stops the process. In the meantime, you have been waiting. The estimates that you finally receive, if you receive any at all, may be nothing more than numerical roundoff.

Stata watches for these sorts of problems, alerts us, fixes them, and properly fits the model.

Let's return to our automobile data. Among the variables we have in the data is one called `repair`, which takes on three values. A value of 1 indicates that the car has a poor repair record, 2 indicates an average record, and 3 indicates a better-than-average record. Here is a tabulation of our data:

```
. use http://www.stata-press.com/data/r11/repair, clear
(1978 Automobile Data)
. tabulate foreign repair
```

| foreign  | repair |    |    | Total |
|----------|--------|----|----|-------|
|          | 1      | 2  | 3  |       |
| Domestic | 10     | 27 | 9  | 46    |
| Foreign  | 0      | 3  | 9  | 12    |
| Total    | 10     | 30 | 18 | 58    |

All the cars with poor repair records (`repair = 1`) are domestic. If we were to attempt to predict `foreign` on the basis of the repair records, the predicted probability for the `repair = 1` category would have to be zero. This in turn means that the logit coefficient must be minus infinity, and that would set most computer programs buzzing.

(Continued on next page)

Let's try Stata on this problem.

```
. logit foreign b3.repair
note: 1.repair != 0 predicts failure perfectly
 1.repair dropped and 10 obs not used

Iteration 0: log likelihood = -26.992087
Iteration 1: log likelihood = -22.483187
Iteration 2: log likelihood = -22.230498
Iteration 3: log likelihood = -22.229139
Iteration 4: log likelihood = -22.229138

Logistic regression Number of obs = 48
 LR chi2(1) = 9.53
 Prob > chi2 = 0.0020
 Pseudo R2 = 0.1765

Log likelihood = -22.229138
```

| foreign | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|---------|-----------|-----------|-------|-------|----------------------|-----------|
| repair  |           |           |       |       |                      |           |
| 1       | (empty)   |           |       |       |                      |           |
| 2       | -2.197225 | .7698003  | -2.85 | 0.004 | -3.706005            | -.6884436 |
| _cons   | 7.94e-17  | .4714045  | 0.00  | 1.000 | -.9239359            | .9239359  |

Remember that all the cars with poor repair records (`repair = 1`) are domestic, so the model cannot be fit, or at least it cannot be fit if we restrict ourselves to finite coefficients. Stata noted that fact “note: 1.repair !=0 predicts failure perfectly”. This is Stata’s mathematically precise way of saying what we said in English. When `repair` is 1, the car is domestic.

Stata then went on to say “1.repair dropped and 10 obs not used”. This is Stata eliminating the problem. First `1.repair` had to be removed from the model because it would have an infinite coefficient. Then the 10 observations that led to the problem had to be eliminated, as well, so as not to bias the remaining coefficients in the model. The 10 observations that are not used are the 10 domestic cars that have poor repair records.

Stata then fit what was left of the model, using the remaining observations. Because no observations remained for cars with poor repair records, Stata reports “(empty)” in the row for `repair = 1`.



□ Technical note

Stata is pretty smart about catching problems like this. It will catch “one-way causation by a dummy variable”, as we demonstrated above.

Stata also watches for “two-way causation”, that is, a variable that perfectly determines the outcome, both successes and failures. Here Stata says, “so-and-so predicts outcome perfectly” and stops. Statistics dictates that no model can be fit.

Stata also checks your data for collinear variables; it will say, “so-and-so omitted because of collinearity”. No observations need to be eliminated in this case, and model fitting will proceed without the offending variable.

It will also catch a subtle problem that can arise with continuous data. For instance, if we were estimating the chances of surviving the first year after an operation, and if we included in our model `age`, and if all the persons over 65 died within the year, Stata would say, “age > 65 predicts failure perfectly”. It would then inform us about the fix-up it takes and fit what can be fit of our model.



logit (and logistic, probit, and ivprobit) will also occasionally display messages such as

Note: 4 failures and 0 successes completely determined.

There are two causes for a message like this. The first—and most unlikely—case occurs when a continuous variable (or a combination of a continuous variable with other continuous or dummy variables) is simply a great predictor of the dependent variable. Consider Stata's `auto.dta` dataset with 6 observations removed.

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. drop if foreign==0 & gear_ratio > 3.1
(6 observations deleted)

. logit foreign mpg weight gear_ratio, nolog
Logistic regression
```

|               |   |        |
|---------------|---|--------|
| Number of obs | = | 68     |
| LR chi2(3)    | = | 72.64  |
| Prob > chi2   | = | 0.0000 |
| Pseudo R2     | = | 0.8484 |

```
Log likelihood = -6.4874814
```

| foreign    | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|------------|-----------|-----------|-------|-------|----------------------|----------|
| mpg        | -.4944907 | .2655508  | -1.86 | 0.063 | -1.014961            | .0259792 |
| weight     | -.0060919 | .003101   | -1.96 | 0.049 | -.0121698            | -.000014 |
| gear_ratio | 15.70509  | 8.166234  | 1.92  | 0.054 | -.3004359            | 31.71061 |
| _cons      | -21.39527 | 25.41486  | -0.84 | 0.400 | -71.20747            | 28.41694 |

Note: 4 failures and 0 successes completely determined.

There are no missing standard errors in the output. If you receive the “completely determined” message and have one or more missing standard errors in your output, see the second case discussed below.

Note `gear_ratio`'s large coefficient. `logit` thought that the 4 observations with the smallest predicted probabilities were essentially predicted perfectly.

```
. predict p
(option pr assumed; Pr(foreign))

. sort p

. list p in 1/4
```

|    | P        |
|----|----------|
| 1. | 1.34e-10 |
| 2. | 6.26e-09 |
| 3. | 7.84e-09 |
| 4. | 1.49e-08 |

If this happens to you, you do not have to do anything. Computationally, the model is sound. The second case discussed below requires careful examination.

The second case occurs when the independent terms are all dummy variables or continuous ones with repeated values (e.g., age). Here one or more of the estimated coefficients will have missing standard errors. For example, consider this dataset consisting of 5 observations.

```
. use http://www.stata-press.com/data/r11/logitxmpl, clear
. list, separator(0)
```

|    | y | x1 | x2 |
|----|---|----|----|
| 1. | 0 | 0  | 0  |
| 2. | 0 | 0  | 0  |
| 3. | 0 | 1  | 0  |
| 4. | 1 | 1  | 0  |
| 5. | 0 | 0  | 1  |
| 6. | 1 | 0  | 1  |

```
. logit y x1 x2

Iteration 0: log likelihood = -3.819085
Iteration 1: log likelihood = -2.9527336
Iteration 2: log likelihood = -2.8110282
Iteration 3: log likelihood = -2.7811973
Iteration 4: log likelihood = -2.7746107
Iteration 5: log likelihood = -2.7730128
(output omitted)
Iteration 15996: log likelihood = -2.7725887 (not concave)
Iteration 15997: log likelihood = -2.7725887 (not concave)
Iteration 15998: log likelihood = -2.7725887 (not concave)
Iteration 15999: log likelihood = -2.7725887 (not concave)
Iteration 16000: log likelihood = -2.7725887 (not concave)
convergence not achieved

Logistic regression Number of obs = 6
 LR chi2(2) = 2.09
 Prob > chi2 = 0.1480
Log likelihood = -2.7725887 Pseudo R2 = 0.2740
```

| y     | Coef.    | Std. Err. | z      | P> z  | [95% Conf. Interval] |          |
|-------|----------|-----------|--------|-------|----------------------|----------|
| x1    | 18.3704  | 2         | 9.19   | 0.000 | 14.45047             | 22.29033 |
| x2    | 18.3704  | .         | .      | .     | .                    | .        |
| _cons | -18.3704 | 1.414214  | -12.99 | 0.000 | -21.14221            | -15.5986 |

```
Note: 2 failures and 0 successes completely determined.
convergence not achieved
r(430);
```

Three things are happening here. First, `logit` iterates almost forever and then declares nonconvergence. Second, `logit` can fit the outcome ( $y = 0$ ) for the covariate pattern  $x1 = 0$  and  $x2 = 0$  (i.e., the first two observations) perfectly. This observation is the “2 failures and 0 successes completely determined”. Third, if this observation is dropped, then  $x1$ ,  $x2$ , and the constant are collinear.

This is the cause of the nonconvergence, the message “completely determined”, and the missing standard errors. It happens when you have a covariate pattern (or patterns) with only one outcome and there is collinearity when the observations corresponding to this covariate pattern are dropped.

If this happens to you, confirm the causes. First, identify the covariate pattern with only one outcome. (For your data, replace  $x1$  and  $x2$  with the independent variables of your model.)

```
. egen pattern = group(x1 x2)
. quietly logit y x1 x2, iterate(100)
. predict p
(option pr assumed; Pr(y))
```

```
. summarize p
```

| Variable | Obs | Mean     | Std. Dev. | Min      | Max |
|----------|-----|----------|-----------|----------|-----|
| p        | 6   | .3333333 | .2581989  | 1.05e-08 | .5  |

If successes were completely determined, that means that there are predicted probabilities that are almost 1. If failures were completely determined, that means that there are predicted probabilities that are almost 0. The latter is the case here, so we locate the corresponding value of `pattern`:

```
. tabulate pattern if p < 1e-7
```

| group(x1<br>x2) | Freq. | Percent | Cum.   |
|-----------------|-------|---------|--------|
| 1               | 2     | 100.00  | 100.00 |
| Total           | 2     | 100.00  |        |

Once we omit this covariate `pattern` from the estimation sample, `logit` can deal with the collinearity:

```
. logit y x1 x2 if pattern !=1, nolog
note: x2 omitted because of collinearity
```

Logistic regression

|               |   |        |
|---------------|---|--------|
| Number of obs | = | 4      |
| LR chi2(1)    | = | 0.00   |
| Prob > chi2   | = | 1.0000 |
| Pseudo R2     | = | 0.0000 |

Log likelihood = -2.7725887

| y     | Coef.     | Std. Err. | z    | P> z  | [95% Conf. Interval] |
|-------|-----------|-----------|------|-------|----------------------|
| x1    | 0         | 2         | 0.00 | 1.000 | -3.919928 3.919928   |
| x2    | (omitted) |           |      |       |                      |
| _cons | 0         | 1.414214  | 0.00 | 1.000 | -2.771808 2.771808   |

We omit the collinear variable. Then we must decide whether to include or omit the observations with `pattern = 1`. We could include them,

```
. logit y x1, nolog
```

Logistic regression

|               |   |        |
|---------------|---|--------|
| Number of obs | = | 6      |
| LR chi2(1)    | = | 0.37   |
| Prob > chi2   | = | 0.5447 |
| Pseudo R2     | = | 0.0480 |

Log likelihood = -3.6356349

| y     | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |
|-------|-----------|-----------|-------|-------|----------------------|
| x1    | 1.098612  | 1.825742  | 0.60  | 0.547 | -2.479776 4.677001   |
| _cons | -1.098612 | 1.154701  | -0.95 | 0.341 | -3.361784 1.164559   |

(Continued on next page)

or exclude them,

```
. logit y x1 if pattern !=1, nolog
Logistic regression Number of obs = 4
 LR chi2(1) = 0.00
 Prob > chi2 = 1.0000
Log likelihood = -2.7725887 Pseudo R2 = 0.0000
```

| y     | Coef. | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|-------|-------|-----------|------|-------|----------------------|----------|
| x1    | 0     | 2         | 0.00 | 1.000 | -3.919928            | 3.919928 |
| _cons | 0     | 1.414214  | 0.00 | 1.000 | -2.771808            | 2.771808 |

If the covariate pattern that predicts outcome perfectly is meaningful, you may want to exclude these observations from the model. Here you would report that covariate pattern such and such predicted outcome perfectly and that the best model for the rest of the data is . . . . But, more likely, the perfect prediction was simply the result of having too many predictors in the model. Then you would omit the extraneous variables from further consideration and report the best model for all the data.



## Saved results

logit saves the following in e():

|               |                                                |
|---------------|------------------------------------------------|
| Scalars       |                                                |
| e(N)          | number of observations                         |
| e(N_cds)      | number of completely determined successes      |
| e(N_cdf)      | number of completely determined failures       |
| e(k)          | number of parameters                           |
| e(k_eq)       | number of equations                            |
| e(k_eq_model) | number of equations in model Wald test         |
| e(k_dv)       | number of dependent variables                  |
| e(k_autoCns)  | number of base, empty, and omitted constraints |
| e(df_m)       | model degrees of freedom                       |
| e(r2_p)       | pseudo- <i>R</i> -squared                      |
| e(l1)         | log likelihood                                 |
| e(l1_0)       | log likelihood, constant-only model            |
| e(N_clust)    | number of clusters                             |
| e(chi2)       | $\chi^2$                                       |
| e(p)          | significance of model test                     |
| e(rank)       | rank of e(V)                                   |
| e(ic)         | number of iterations                           |
| e(rc)         | return code                                    |
| e(converged)  | 1 if converged, 0 otherwise                    |

## Macros

|                                 |                                                                          |
|---------------------------------|--------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>logit</code>                                                       |
| <code>e(cmdline)</code>         | command as typed                                                         |
| <code>e(depvar)</code>          | name of dependent variable                                               |
| <code>e(wtype)</code>           | weight type                                                              |
| <code>e(wexp)</code>            | weight expression                                                        |
| <code>e(title)</code>           | title in estimation output                                               |
| <code>e(clustvar)</code>        | name of cluster variable                                                 |
| <code>e(offset)</code>          | offset                                                                   |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                  |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                           |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                            |
| <code>e(opt)</code>             | type of optimization                                                     |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>       | type of ml method                                                        |
| <code>e(user)</code>            | name of likelihood-evaluator program                                     |
| <code>e(technique)</code>       | maximization technique                                                   |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular              |
| <code>e(crittype)</code>        | optimization criterion                                                   |
| <code>e(properties)</code>      | b V                                                                      |
| <code>e(estat_cmd)</code>       | program used to implement <code>estat</code>                             |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                           |
| <code>e(marginsnotok)</code>    | predictions disallowed by <code>margins</code>                           |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>           |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>           |

## Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(gradient)</code>     | gradient vector                              |
| <code>e(mns)</code>          | vector of means of the independent variables |
| <code>e(rules)</code>        | information about perfect predictors         |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`logit` is implemented as an ado-file.

Cramer (2003, chap. 9) surveys the prehistory and history of the logit model. The word “logit” was coined by Berkson (1944) and is analogous to the word “probit”. For an introduction to probit and logit, see, for example, Aldrich and Nelson (1984), Cameron and Trivedi (2009), Greene (2008), Jones (2007), Long (1997), Long and Freese (2006), Pampel (2000), or Powers and Xie (2008).

The likelihood function for logit is

$$\ln L = \sum_{j \in S} w_j \ln F(\mathbf{x}_j \mathbf{b}) + \sum_{j \notin S} w_j \ln \{1 - F(\mathbf{x}_j \mathbf{b})\}$$

where  $S$  is the set of all observations  $j$ , such that  $y_j \neq 0$ ,  $F(z) = e^z / (1 + e^z)$ , and  $w_j$  denotes the optional weights.  $\ln L$  is maximized as described in [R] [maximize](#).

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#). The scores are calculated as  $\mathbf{u}_j = \{1 - F(\mathbf{x}_j\mathbf{b})\}\mathbf{x}_j$  for the positive outcomes and  $-F(\mathbf{x}_j\mathbf{b})\mathbf{x}_j$  for the negative outcomes.

`logit` also supports estimation with survey data. For details on VCEs with survey data, see [SVY] [variance estimation](#).

Joseph Berkson (1899–1982) was born in New York City and studied at the College of the City of New York, Columbia, and Johns Hopkins, earning both an MD and a doctorate in statistics. He then worked at Johns Hopkins before moving to the Mayo Clinic in 1931 as a biostatistician. Among many other contributions, his most influential drew upon a long-sustained interest in the logistic function, especially his 1944 paper on bioassay, in which he introduced the term “logit”. Berkson was a frequent participant in controversy, sometimes humorous, sometimes bitter, on subjects such as the evidence for links between smoking and various diseases and the relative merits of probit and logit methods and of different calculation methods.

## References

- Aldrich, J. H., and F. D. Nelson. 1984. *Linear Probability, Logit, and Probit Models*. Newbury Park, CA: Sage.
- Archer, K. J., and S. Lemeshow. 2006. [Goodness-of-fit test for a logistic regression model fitted using survey sample data](#). *Stata Journal* 6: 97–105.
- Berkson, J. 1944. Application of the logistic function to bio-assay. *Journal of the American Statistical Association* 39: 357–365.
- Cameron, A. C., and P. K. Trivedi. 2009. *Microeconometrics Using Stata*. College Station, TX: Stata Press.
- Cleves, M. A., and A. Tosetto. 2000. [sg139: Logistic regression when binary outcome is measured with uncertainty](#). *Stata Technical Bulletin* 55: 20–23. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 152–156. College Station, TX: Stata Press.
- Cramer, J. S. 2003. *Logit Models from Economics and Other Fields*. Cambridge: Cambridge University Press.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- Hilbe, J. M. 2009. *Logistic Regression Models*. Boca Raton, FL: Chapman & Hill/CRC.
- Hosmer Jr., D. W., and S. Lemeshow. 2000. *Applied Logistic Regression*. 2nd ed. New York: Wiley.
- Jones, A. 2007. *Applied Econometrics for Health Economists: A Practical Guide*. 2nd ed. Abingdon, UK: Radcliffe.
- Judge, G. G., W. E. Griffiths, R. C. Hill, H. Lütkepohl, and T.-C. Lee. 1985. *The Theory and Practice of Econometrics*. 2nd ed. New York: Wiley.
- Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.
- Long, J. S., and J. Freese. 2006. *Regression Models for Categorical Dependent Variables Using Stata*. 2nd ed. College Station, TX: Stata Press.
- Miranda, A., and S. Rabe-Hesketh. 2006. [Maximum likelihood estimation of endogenous switching and sample selection models for binary, ordinal, and count variables](#). *Stata Journal* 6: 285–308.
- Mitchell, M. N., and X. Chen. 2005. [Visualizing main effects and interactions for binary logit models](#). *Stata Journal* 5: 64–82.
- O’Fallon, W. M. 1998. Berkson, Joseph. In Vol. 1 of *Encyclopedia of Biostatistics*, ed. P. Armitage and T. Colton, 290–295. Chichester, UK: Wiley.
- Pampel, F. C. 2000. *Logistic Regression: A Primer*. Thousand Oaks, CA: Sage.

- Powers, D. A., and Y. Xie. 2008. *Statistical Methods for Categorical Data Analysis*. 2nd ed. Bingley, UK: Emerald.
- Pregibon, D. 1981. Logistic regression diagnostics. *Annals of Statistics* 9: 705–724.
- Schonlau, M. 2005. [Boosted regression \(boosting\): An introductory tutorial and a Stata plugin](#). *Stata Journal* 5: 330–354.
- Xu, J., and J. S. Long. 2005. [Confidence intervals for predicted outcomes in regression models for categorical outcomes](#). *Stata Journal* 5: 537–559.

## Also see

- [R] [logit postestimation](#) — Postestimation tools for logit
- [R] [roc](#) — Receiver operating characteristic (ROC) analysis
- [R] [brier](#) — Brier score decomposition
- [R] [exlogistic](#) — Exact logistic regression
- [R] [glogit](#) — Logit and probit regression for grouped data
- [R] [logistic](#) — Logistic regression, reporting odds ratios
- [R] [probit](#) — Probit regression
- [SVY] [svy estimation](#) — Estimation commands for survey data
- [XT] [xtlogit](#) — Fixed-effects, random-effects, and population-averaged logit models
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are of special interest after `logit`:

| command                 | description                                                                                              |
|-------------------------|----------------------------------------------------------------------------------------------------------|
| <code>estat clas</code> | <code>estat classification</code> reports various summary statistics, including the classification table |
| <code>estat gof</code>  | Pearson or Hosmer–Lemeshow goodness-of-fit test                                                          |
| <code>lroc</code>       | graphs the ROC curve and calculates the area under the curve                                             |
| <code>lsens</code>      | graphs sensitivity and specificity versus probability cutoff                                             |

These commands are not appropriate after the `svy` prefix.

For information about these commands, see [\[R\] logistic postestimation](#).

The following standard postestimation commands are also available:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>            | link test for model specification                                                                   |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.



## Syntax for predict

```
predict [type] newvar [if] [in] [, statistic nooffset rules asif]
```

| statistic                | description                                                                             |
|--------------------------|-----------------------------------------------------------------------------------------|
| Main                     |                                                                                         |
| <code>pr</code>          | probability of a positive outcome; the default                                          |
| <code>xb</code>          | $\mathbf{x}_j\mathbf{b}$ , fitted values                                                |
| <code>stdp</code>        | standard error of the prediction                                                        |
| * <code>dbeta</code>     | <a href="#">Pregibon (1981)</a> $\Delta\hat{\beta}$ influence statistic                 |
| * <code>deviance</code>  | deviance residual                                                                       |
| * <code>dx2</code>       | <a href="#">Hosmer Jr. and Lemeshow (2000)</a> $\Delta\chi^2$ influence statistic       |
| * <code>ddeviance</code> | <a href="#">Hosmer Jr. and Lemeshow (2000)</a> $\Delta D$ influence statistic           |
| * <code>hat</code>       | <a href="#">Pregibon (1981)</a> leverage                                                |
| * <code>number</code>    | sequential number of the covariate pattern                                              |
| * <code>residuals</code> | Pearson residuals; adjusted for number sharing covariate pattern                        |
| * <code>rstandard</code> | standardized Pearson residuals; adjusted for number sharing covariate pattern           |
| <code>score</code>       | first derivative of the log likelihood with respect to $\mathbf{x}_j\boldsymbol{\beta}$ |

Unstarred statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample. Starred statistics are calculated only for the estimation sample, even when `if e(sample)` is not specified.

`pr`, `xb`, `stdp`, and `score` are the only options allowed with `svy` estimation results.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

### Main

`pr`, the default, calculates the probability of a positive outcome.

`xb` calculates the linear prediction.

`stdp` calculates the standard error of the linear prediction.

`dbeta` calculates the [Pregibon \(1981\)](#)  $\Delta\hat{\beta}$  influence statistic, a standardized measure of the difference in the coefficient vector that is due to deletion of the observation along with all others that share the same covariate pattern. In [Hosmer Jr. and Lemeshow \(2000\)](#) jargon, this statistic is  $M$ -asymptotic; that is, it is adjusted for the number of observations that share the same covariate pattern.

`deviance` calculates the deviance residual.

`dx2` calculates the [Hosmer Jr. and Lemeshow \(2000\)](#)  $\Delta\chi^2$  influence statistic, reflecting the decrease in the Pearson  $\chi^2$  that is due to the deletion of the observation and all others that share the same covariate pattern.

`ddeviance` calculates the [Hosmer Jr. and Lemeshow \(2000\)](#)  $\Delta D$  influence statistic, which is the change in the deviance residual that is due to deletion of the observation and all others that share the same covariate pattern.

**hat** calculates the [Pregibon \(1981\)](#) leverage or the diagonal elements of the hat matrix adjusted for the number of observations that share the same covariate pattern.

**number** numbers the covariate patterns—observations with the same covariate pattern have the same **number**. Observations not used in estimation have **number** set to missing. The first covariate pattern is numbered 1, the second 2, and so on.

**residuals** calculates the Pearson residual as given by [Hosmer Jr. and Lemeshow \(2000\)](#) and adjusted for the number of observations that share the same covariate pattern.

**rstandard** calculates the standardized Pearson residual as given by [Hosmer Jr. and Lemeshow \(2000\)](#) and adjusted for the number of observations that share the same covariate pattern.

**score** calculates the equation-level score,  $\partial \ln L / \partial (\mathbf{x}_j \boldsymbol{\beta})$ .

#### Options

**nooffset** is relevant only if you specified **offset**(*varname*) for **logit**. It modifies the calculations made by **predict** so that they ignore the offset variable; the linear prediction is treated as  $\mathbf{x}_j \mathbf{b}$  rather than as  $\mathbf{x}_j \mathbf{b} + \text{offset}_j$ .

**rules** requests that Stata use any rules that were used to identify the model when making the prediction. By default, Stata calculates missing for excluded observations.

**asif** requests that Stata ignore the rules and exclusion criteria and calculate predictions for all observations possible using the estimated parameter from the model.

## Remarks

Once you have fit a logit model, you can obtain the predicted probabilities by using the **predict** command for both the estimation sample and other samples; see [\[U\] 20 Estimation and postestimation commands](#) and [\[R\] predict](#). Here we will make only a few more comments.

**predict** without arguments calculates the predicted probability of a positive outcome, i.e.,  $\Pr(y_j = 1) = F(\mathbf{x}_j \mathbf{b})$ . With the **xb** option, **predict** calculates the linear combination  $\mathbf{x}_j \mathbf{b}$ , where  $\mathbf{x}_j$  are the independent variables in the  $j$ th observation and  $\mathbf{b}$  is the estimated parameter vector. This is sometimes known as the index function because the cumulative distribution function indexed at this value is the probability of a positive outcome.

In both cases, Stata remembers any rules used to identify the model and calculates missing for excluded observations, unless **rules** or **asif** is specified. For information about the other statistics available after **predict**, see [\[R\] logistic postestimation](#).

## ► Example 1

In [example 2](#) of [\[R\] logit](#), we fit the logit model **logit foreign b3.repair**. To obtain predicted probabilities, type

```
. use http://www.stata-press.com/data/r11/repair
(1978 Automobile Data)
. logit foreign b3.repair
note: 1.repair != 0 predicts failure perfectly
 1.repair dropped and 10 obs not used
 (output omitted)
. predict p
(option pr assumed; Pr(foreign))
(10 missing values generated)
```

```
. summarize foreign p
```

| Variable | Obs | Mean     | Std. Dev. | Min | Max |
|----------|-----|----------|-----------|-----|-----|
| foreign  | 58  | .2068966 | .4086186  | 0   | 1   |
| p        | 48  | .25      | .1956984  | .1  | .5  |

Stata remembers any rules used to identify the model and sets predictions to missing for any excluded observations. `logit` dropped the variable `1.repair` from our model and excluded 10 observations. Thus when we typed `predict p`, those same 10 observations were again excluded, and their predictions were set to missing.

`predict`'s `rules` option uses the rules in the prediction. During estimation, we were told “`1.repair` != 0 predicts failure perfectly”, so the rule is that when `1.repair` is not zero, we should predict 0 probability of success or a positive outcome:

```
. predict p2, rules
(option pr assumed; Pr(foreign))
. summarize foreign p p2
```

| Variable | Obs | Mean     | Std. Dev. | Min | Max |
|----------|-----|----------|-----------|-----|-----|
| foreign  | 58  | .2068966 | .4086186  | 0   | 1   |
| p        | 48  | .25      | .1956984  | .1  | .5  |
| p2       | 58  | .2068966 | .2016268  | 0   | .5  |

`predict`'s `asif` option ignores the rules and exclusion criteria and calculates predictions for all observations possible by using the estimated parameters from the model:

```
. predict p3, asif
(option pr assumed; Pr(foreign))
. summarize foreign p p2 p3
```

| Variable | Obs | Mean     | Std. Dev. | Min | Max |
|----------|-----|----------|-----------|-----|-----|
| foreign  | 58  | .2068966 | .4086186  | 0   | 1   |
| p        | 48  | .25      | .1956984  | .1  | .5  |
| p2       | 58  | .2068966 | .2016268  | 0   | .5  |
| p3       | 58  | .2931035 | .2016268  | .1  | .5  |

Which is right? What `predict` does by default is the most conservative approach. If many observations had been excluded because of a simple rule, we could be reasonably certain that the `rules` prediction is correct. The `asif` prediction is correct only if the exclusion is a fluke, and we would be willing to exclude the variable from the analysis anyway. Then, however, we would refit the model to include the excluded observations.

◀

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

See [Methods and formulas](#) of [R] [logistic postestimation](#) for details.

## References

- Archer, K. J., and S. Lemeshow. 2006. Goodness-of-fit test for a logistic regression model fitted using survey sample data. *Stata Journal* 6: 97–105.
- Hosmer Jr., D. W., and S. Lemeshow. 2000. *Applied Logistic Regression*. 2nd ed. New York: Wiley.
- Pregibon, D. 1981. Logistic regression diagnostics. *Annals of Statistics* 9: 705–724.

## Also see

[R] [logit](#) — Logistic regression, reporting coefficients

[R] [logistic postestimation](#) — Postestimation tools for logistic

[U] [20 Estimation and postestimation commands](#)

Syntax

```
loneway response_var group_var [if] [in] [weight] [, options]
```

| options               | description                                                       |
|-----------------------|-------------------------------------------------------------------|
| <hr/>                 |                                                                   |
| Main                  |                                                                   |
| <code>mean</code>     | expected value of $F$ distribution; default is 1                  |
| <code>median</code>   | median of $F$ distribution; default is 1                          |
| <code>exact</code>    | exact confidence intervals (groups must be equal with no weights) |
| <code>level(#)</code> | set confidence level; default is <code>level(95)</code>           |

`by` is allowed; see [D] [by](#).  
`aweight`s are allowed; see [U] [11.1.6 weight](#).

Menu

Statistics > Linear models and related > ANOVA/MANOVA > Large one-way ANOVA

Description

`loneway` fits one-way analysis-of-variance (ANOVA) models on datasets with many levels of `group_var` and presents different ancillary statistics from `oneway` (see [R] [oneway](#)):

| Feature                                    | oneway | loneway |
|--------------------------------------------|--------|---------|
| Fit one-way model                          | x      | x       |
| on fewer than 376 levels                   | x      | x       |
| on more than 376 levels                    |        | x       |
| Bartlett's test for equal variance         | x      |         |
| Multiple-comparison tests                  | x      |         |
| Intragroup correlation and SE              |        | x       |
| Intragroup correlation confidence interval |        | x       |
| Est. reliability of group-averaged score   |        | x       |
| Est. SD of group effect                    |        | x       |
| Est. SD within group                       |        | x       |

Options

Main

- `mean` specifies that the expected value of the  $F_{k-1,N-k}$  distribution be used as the reference point  $F_m$  in the estimation of  $\rho$  instead of the default value of 1.
- `median` specifies that the median of the  $F_{k-1,N-k}$  distribution be used as the reference point  $F_m$  in the estimation of  $\rho$  instead of the default value of 1.

`exact` requests that exact confidence intervals be computed, as opposed to the default asymptotic confidence intervals. This option is allowed only if the groups are equal in size and weights are not used.

`level(#)` specifies the confidence level, as a percentage, for confidence intervals of the coefficients. The default is `level(95)` or as set by `set level`; see [U] [20.7 Specifying the width of confidence intervals](#).

## Remarks

Remarks are presented under the following headings:

- The one-way ANOVA model*
- R-squared*
- The random-effects ANOVA model*
- Intraclass correlation*
- Estimated reliability of the group-averaged score*

## The one-way ANOVA model

### ► Example 1

`loneaway`'s output looks like that of `oneway`, except that `loneaway` presents more information at the end. Using our automobile dataset, we have created a (numeric) variable called `manufacturer_grp` identifying the manufacturer of each car, and within each manufacturer we have retained a maximum of four models, selecting those with the lowest mpg. We can compute the intraclass correlation of mpg for all manufacturers with at least four models as follows:

```
. use http://www.stata-press.com/data/r11/auto7
(1978 Automobile Data)
. loneaway mpg manufacturer_grp if nummake == 4
```

One-way Analysis of Variance for mpg: Mileage (mpg)

|                      |           |    | Number of obs = | 36     |          |
|----------------------|-----------|----|-----------------|--------|----------|
|                      |           |    | R-squared =     | 0.5228 |          |
| Source               | SS        | df | MS              | F      | Prob > F |
| Between manufactur~p | 621.88889 | 8  | 77.736111       | 3.70   | 0.0049   |
| Within manufactur~p  | 567.75    | 27 | 21.027778       |        |          |
| Total                | 1189.6389 | 35 | 33.989683       |        |          |

|                        |           |                      |
|------------------------|-----------|----------------------|
| Intraclass correlation | Asy. S.E. | [95% Conf. Interval] |
| 0.40270                | 0.18770   | 0.03481 0.77060      |

|                                         |          |
|-----------------------------------------|----------|
| Estimated SD of manufactur~p effect     | 3.765247 |
| Estimated SD within manufactur~p        | 4.585605 |
| Est. reliability of a manufactur~p mean | .72950   |
| (evaluated at n=4.00)                   |          |

In addition to the standard one-way ANOVA output, `loneaway` produces the *R*-squared, the estimated standard deviation of the group effect, the estimated standard deviation within group, the intragroup correlation, the estimated reliability of the group-averaged mean, and, for unweighted data, the asymptotic standard error and confidence interval for the intragroup correlation.

## R-squared

The  $R$ -squared is, of course, simply the underlying  $R^2$  for a regression of *response\_var* on the levels of *group\_var*, or *mpg* on the various manufacturers here.

## The random-effects ANOVA model

loneway assumes that we observe a variable,  $y_{ij}$ , measured for  $n_i$  elements within  $k$  groups or classes such that

$$y_{ij} = \mu + \alpha_i + \epsilon_{ij}, \quad i = 1, 2, \dots, k, \quad j = 1, 2, \dots, n_i$$

and  $\alpha_i$  and  $\epsilon_{ij}$  are independent zero-mean random variables with variance  $\sigma_\alpha^2$  and  $\sigma_\epsilon^2$ , respectively. This is the random-effects ANOVA model, also known as the components-of-variance model, in which it is typically assumed that the  $y_{ij}$  are normally distributed.

The interpretation with respect to our example is that the observed value of our response variable, *mpg*, is created in two steps. First, the  $i$ th manufacturer is chosen, and a value,  $\alpha_i$ , is determined—the typical *mpg* for that manufacturer less the overall *mpg*  $\mu$ . Then a deviation,  $\epsilon_{ij}$ , is chosen for the  $j$ th model within this manufacturer. This is how much that particular automobile differs from the typical *mpg* value for models from this manufacturer.

For our sample of 36 car models, the estimated standard deviations are  $\sigma_\alpha = 3.8$  and  $\sigma_\epsilon = 4.6$ . Thus a little more than half of the variation in *mpg* between cars is attributable to the car model, with the rest attributable to differences between manufacturers. These standard deviations differ from those that would be produced by a (standard) fixed-effects regression in that the regression would require the sum within each manufacturer of the  $\epsilon_{ij}$ ,  $\epsilon_i$ , for the  $i$ th manufacturer, to be zero, whereas these estimates merely impose the constraint that the sum is *expected* to be zero.

## Intraclass correlation

There are various estimators of the intraclass correlation, such as the pairwise estimator, which is defined as the Pearson product-moment correlation computed over all possible pairs of observations that can be constructed within groups. For a discussion of various estimators, see [Donner \(1986\)](#). loneway computes what is termed the analysis of variance, or ANOVA, estimator. This intraclass correlation is the theoretical upper bound on the variation in *response\_var* that is explainable by *group\_var*, of which  $R$ -squared is an overestimate because of the serendipity of fitting. This correlation is comparable to an  $R$ -squared—you do not have to square it.

In our example, the intra-manu correlation, the correlation of *mpg* within manufacturer, is 0.40. Because *aweights* were not used and the default correlation was computed (i.e., the *mean* and *median* options were not specified), loneway also provided the asymptotic confidence interval and standard error of the intraclass correlation estimate.

## Estimated reliability of the group-averaged score

The estimated reliability of the group-averaged score or mean has an interpretation similar to that of the intragroup correlation; it is a comparable number if we average *response\_var* by *group\_var*, or *mpg* by *manu* in our example. It is the theoretical upper bound of a regression of manufacturer-averaged *mpg* on characteristics of manufacturers. Why would we want to collapse our 36-observation dataset into a 9-observation dataset of manufacturer averages? Because the 36 observations might be a mirage. When General Motors builds cars, do they sometimes put a Pontiac label and sometimes a Chevrolet label on them, so that it appears in our data as if we have two cars when we really have

only one, replicated? If that were the case, and if it were the case for many other manufacturers, then we would be forced to admit that we do not have data on 36 cars; we instead have data on nine manufacturer-averaged characteristics.

## Saved results

`loneway` saves the following in `r()`:

|                     |                               |                       |                                      |
|---------------------|-------------------------------|-----------------------|--------------------------------------|
| Scalars             |                               |                       |                                      |
| <code>r(N)</code>   | number of observations        | <code>r(rho_t)</code> | estimated reliability                |
| <code>r(rho)</code> | intraclass correlation        | <code>r(se)</code>    | asympt. SE of intraclass correlation |
| <code>r(lb)</code>  | lower bound of 95% CI for rho | <code>r(sd_w)</code>  | estimated SD within group            |
| <code>r(ub)</code>  | upper bound of 95% CI for rho | <code>r(sd_b)</code>  | estimated SD of group effect         |

## Methods and formulas

`loneway` is implemented as an ado-file.

The mean squares in the `loneway`'s ANOVA table are computed as

$$MS_{\alpha} = \sum_i w_{i.} (\bar{y}_{i.} - \bar{y}_{..})^2 / (k - 1)$$

and

$$MS_{\epsilon} = \sum_i \sum_j w_{ij} (y_{ij} - \bar{y}_{i.})^2 / (N - k)$$

in which

$$w_{i.} = \sum_j w_{ij} \quad w_{..} = \sum_i w_{i.} \quad \bar{y}_{i.} = \sum_j w_{ij} y_{ij} / w_{i.} \quad \text{and} \quad \bar{y}_{..} = \sum_i w_{i.} \bar{y}_{i.} / w_{..}$$

The corresponding expected values of these mean squares are

$$E(MS_{\alpha}) = \sigma_{\epsilon}^2 + g \sigma_{\alpha}^2 \quad \text{and} \quad E(MS_{\epsilon}) = \sigma_{\epsilon}^2$$

in which

$$g = \frac{w_{..} - \sum_i w_{i.}^2 / w_{..}}{k - 1}$$

In the unweighted case, we get

$$g = \frac{N - \sum_i n_i^2 / N}{k - 1}$$

As expected,  $g = m$  for the case of no weights and equal group sizes in the data, i.e.,  $n_i = m$  for all  $i$ . Replacing the expected values with the observed values and solving yields the ANOVA estimates of  $\sigma_{\alpha}^2$  and  $\sigma_{\epsilon}^2$ . Substituting these into the definition of the intraclass correlation

$$\rho = \frac{\sigma_{\alpha}^2}{\sigma_{\alpha}^2 + \sigma_{\epsilon}^2}$$

yields the ANOVA estimator of the intraclass correlation:

$$\rho_A = \frac{F_{\text{obs}} - 1}{F_{\text{obs}} - 1 + g}$$



$F_{\text{obs}}$  is the observed value of the  $F$  statistic from the ANOVA table. For no weights and equal  $n_i$ ,  $\rho_A = \text{roh}$ , which is the intragroup correlation defined by Kish (1965). Two slightly different estimators are available through the `mean` and `median` options (Gleason 1997). If either of these options is specified, the estimate of  $\rho$  becomes

$$\rho = \frac{F_{\text{obs}} - F_m}{F_{\text{obs}} + (g - 1)F_m}$$

For the `mean` option,  $F_m = E(F_{k-1, N-K}) = (N - k)/(N - k - 2)$ , i.e., the expected value of the ANOVA table's  $F$  statistic. For the `median` option,  $F_m$  is simply the median of the  $F$  statistic. Setting  $F_m$  to 1 gives  $\rho_A$ , so for large samples, these different point estimators are essentially the same. Also, because the intraclass correlation of the random-effects model is by definition nonnegative, for any of the three possible point estimators,  $\rho$  is truncated to zero if  $F_{\text{obs}}$  is less than  $F_m$ .

For no weighting, interval estimators for  $\rho_A$  are computed. If the groups are equal sized (all  $n_i$  equal) and the `exact` option is specified, the following exact (assuming that the  $y_{ij}$  are normally distributed)  $100(1 - \alpha)\%$  confidence interval is computed:

$$\left\{ \frac{F_{\text{obs}} - F_m F_u}{F_{\text{obs}} + (g - 1)F_m F_u}, \frac{F_{\text{obs}} - F_m F_l}{F_{\text{obs}} + (g - 1)F_m F_l} \right\}$$

with  $F_m = 1$ ,  $F_l = F_{\alpha/2, k-1, N-k}$ , and  $F_u = F_{1-\alpha/2, k-1, N-k}$ ,  $F_{\cdot, k-1, N-k}$  being the cumulative distribution function for the  $F$  distribution with  $k - 1$  and  $N - k$  degrees of freedom. If `mean` or `median` is specified,  $F_m$  is defined as above. If the groups are equal sized and `exact` is not specified, the following asymptotic  $100(1 - \alpha)\%$  confidence interval for  $\rho_A$  is computed,

$$\left[ \rho_A - z_{\alpha/2} \sqrt{V(\rho_A)}, \rho_A + z_{\alpha/2} \sqrt{V(\rho_A)} \right]$$

where  $z_{\alpha/2}$  is the  $100(1 - \alpha/2)$  percentile of the standard normal distribution and  $\sqrt{V(\rho_A)}$  is the asymptotic standard error of  $\rho$  defined below. This confidence interval is also available for unequal groups. It is not applicable and, therefore, not computed for the estimates of  $\rho$  provided by the `mean` and `median` options. Again, because the intraclass coefficient is nonnegative, if the lower bound is negative for either confidence interval, it is truncated to zero. As might be expected, the coverage probability of a truncated interval is higher than its nominal value.

The asymptotic standard error of  $\rho_A$ , assuming that the  $y_{ij}$  are normally distributed, is also computed when appropriate, namely, for unweighted data and when  $\rho_A$  is computed (neither the `mean` option nor the `median` option is specified):

$$V(\rho_A) = \frac{2(1 - \rho)^2}{g^2} (A + B + C)$$

with

$$\begin{aligned} A &= \frac{\{1 + \rho(g - 1)\}^2}{N - k} \\ B &= \frac{(1 - \rho)\{1 + \rho(2g - 1)\}}{k - 1} \\ C &= \frac{\rho^2 \{\sum n_i^2 - 2N^{-1} \sum n_i^3 + N^{-2} (\sum n_i^2)^2\}}{(k - 1)^2} \end{aligned}$$

and  $\rho_A$  is substituted for  $\rho$  (Donner 1986).

The estimated reliability of the group-averaged score, known as the Spearman–Brown prediction formula in the psychometric literature (Winer, Brown, and Michels 1991, 1014), is

$$\rho_t = \frac{t\rho}{1 + (t-1)\rho}$$

for group size  $t$ . `lone way` computes  $\rho_t$  for  $t = g$ .

The estimated standard deviation of the group effect is  $\sigma_\alpha = \sqrt{(\text{MS}_\alpha - \text{MS}_\epsilon)/g}$ . This deviation comes from the assumption that an observation is derived by adding a group effect to a within-group effect.

The estimated standard deviation within group is the square root of the mean square due to error, or  $\sqrt{\text{MS}_\epsilon}$ .

## Acknowledgment

We thank John Gleason of Syracuse University for his contributions to improving `lone way`.

## References

- Donner, A. 1986. A review of inference procedures for the intraclass correlation coefficient in the one-way random effects model. *International Statistical Review* 54: 67–82.
- Gleason, J. R. 1997. [sg65: Computing intraclass correlations and large ANOVAs](#). *Stata Technical Bulletin* 35: 25–31. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 167–176. College Station, TX: Stata Press.
- Kish, L. 1965. *Survey Sampling*. New York: Wiley.
- Marchenko, Y. 2006. [Estimating variance components in Stata](#). *Stata Journal* 6: 1–21.
- Winer, B. J., D. R. Brown, and K. M. Michels. 1991. *Statistical Principles in Experimental Design*. 3rd ed. New York: McGraw–Hill.

## Also see

- [\[R\] anova](#) — Analysis of variance and covariance
- [\[R\] oneway](#) — One-way analysis of variance

Syntax

```
lowess yvar xvar [if] [in] [, options]
```

| options                                                                                                              | description                                                          |
|----------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------|
| Main                                                                                                                 |                                                                      |
| <u>mean</u>                                                                                                          | running-mean smooth; default is running-line least squares           |
| <u>noweight</u>                                                                                                      | suppress weighted regressions; default is tricube weighting function |
| <u>bwidth</u> (#)                                                                                                    | use # for the bandwidth; default is <code>bwidth(0.8)</code>         |
| <u>logit</u>                                                                                                         | transform dependent variable to logits                               |
| <u>adjust</u>                                                                                                        | adjust smoothed mean to equal mean of dependent variable             |
| <u>nograph</u>                                                                                                       | suppress graph                                                       |
| <u>generate</u> ( <i>newvar</i> )                                                                                    | create <i>newvar</i> containing smoothed values of <i>yvar</i>       |
| Plot                                                                                                                 |                                                                      |
| <i>marker_options</i>                                                                                                | change look of markers (color, size, etc.)                           |
| <i>marker_label_options</i>                                                                                          | add marker labels; change look or position                           |
| Smoothed line                                                                                                        |                                                                      |
| <u>lineopts</u> ( <i>cline_options</i> )                                                                             | affect rendition of the smoothed line                                |
| Add plots                                                                                                            |                                                                      |
| <u>addplot</u> ( <i>plot</i> )                                                                                       | add other plots to generated graph                                   |
| Y axis, X axis, Titles, Legend, Overall, By                                                                          |                                                                      |
| <i>twoway_options</i>                                                                                                | any of the options documented in <a href="#">[G] twoway_options</a>  |
| <i>yvar</i> and <i>xvar</i> may contain time-series operators; see <a href="#">[U] 11.4.4 Time-series varlists</a> . |                                                                      |

Menu

Statistics > Nonparametric analysis > Lowess smoothing

Description

`lowess` carries out a locally weighted regression of *yvar* on *xvar*, displays the graph, and optionally saves the smoothed variable.

Warning: `lowess` is computationally intensive and may therefore take a long time to run on a slow computer. Lowess calculations on 1,000 observations, for instance, require performing 1,000 regressions.

## Options

### Main

**mean** specifies running-mean smoothing; the default is running-line least-squares smoothing.

**noweight** prevents the use of Cleveland's (1979) tricube weighting function; the default is to use this weighting function.

**bwidth(#)** specifies the bandwidth. Centered subsets of  $\text{bwidth}() \times N$  observations are used for calculating smoothed values for each point in the data except for the end points, where smaller, uncentered subsets are used. The greater the **bwidth()**, the greater the smoothing. The default is 0.8.

**logit** transforms the smoothed *yvar* into logits. Predicted values less than 0.0001 or greater than 0.9999 are set to  $1/N$  and  $1 - 1/N$ , respectively, before taking logits.

**adjust** adjusts the mean of the smoothed *yvar* to equal the mean of *yvar* by multiplying by an appropriate factor. This option is useful when smoothing binary (0/1) data.

**nograph** suppresses displaying the graph.

**generate(newvar)** creates *newvar* containing the smoothed values of *yvar*.

### Plot

**marker\_options** affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [G] [marker\\_options](#).

**marker\_label\_options** specify if and how the markers are to be labeled; see [G] [marker\\_label\\_options](#).

### Smoothed line

**lineopts(cline\_options)** affects the rendition of the lowess-smoothed line; see [G] [cline\\_options](#).

### Add plots

**addplot(plot)** provides a way to add other plots to the generated graph; see [G] [addplot\\_option](#).

### Y axis, X axis, Titles, Legend, Overall, By

**twoway\_options** are any of the options documented in [G] [twoway\\_options](#). These include options for titling the graph (see [G] [title\\_options](#)), options for saving the graph to disk (see [G] [saving\\_option](#)), and the **by()** option (see [G] [by\\_option](#)).

## Remarks

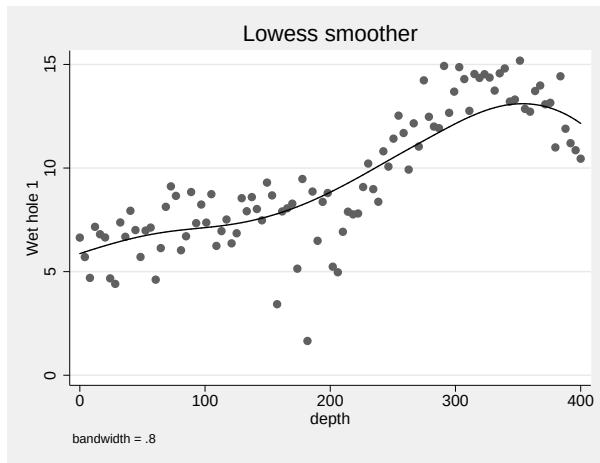
By default, **lowess** provides locally weighted scatterplot smoothing. The basic idea is to create a new variable (*newvar*) that, for each *yvar*  $y_i$ , contains the corresponding smoothed value. The smoothed values are obtained by running a regression of *yvar* on *xvar* by using only the data  $(x_i, y_i)$  and a few of the data near this point. In **lowess**, the regression is weighted so that the central point  $(x_i, y_i)$  gets the highest weight and points that are farther away (based on the distance  $|x_j - x_i|$ ) receive less weight. The estimated regression line is then used to predict the smoothed value  $\hat{y}_i$  for  $y_i$  only. The procedure is repeated to obtain the remaining smoothed values, which means that a separate weighted regression is performed for every point in the data.

Lowess is a desirable smoother because of its locality—it tends to follow the data. Polynomial smoothing methods, for instance, are global in that what happens on the extreme left of a scatterplot can affect the fitted values on the extreme right.

## ► Example 1

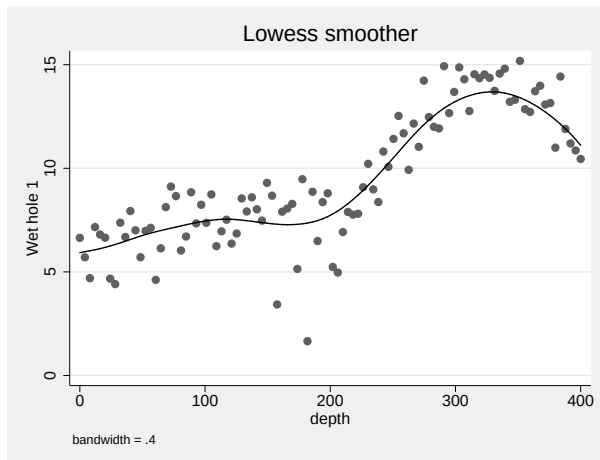
The amount of smoothing is affected by `bwidth(#)`. You are warned to experiment with different values. For instance,

```
. use http://www.stata-press.com/data/r11/lowess1
(example data for lowess)
. lowess h1 depth
```



Now compare that with

```
. lowess h1 depth, bwidth(.4)
```



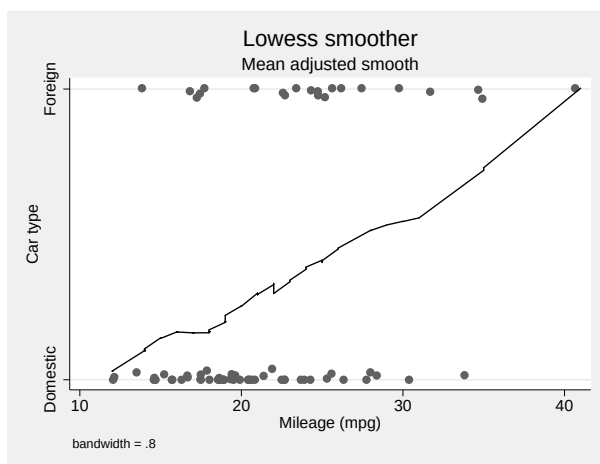
In the first case, the default bandwidth of 0.8 is used, meaning that 80% of the data are used in smoothing each point. In the second case, we explicitly specified a bandwidth of 0.4. Smaller bandwidths follow the original data more closely.

## ► Example 2

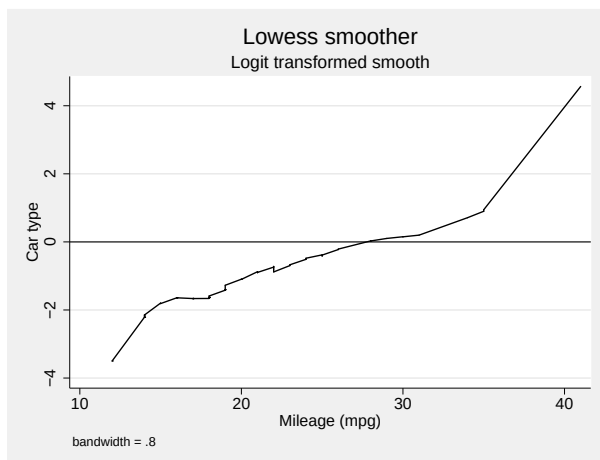
Two lowess options are especially useful with binary (0/1) data: `adjust` and `logit`. `adjust` adjusts the resulting curve (by multiplication) so that the mean of the smoothed values is equal to the mean of the unsmoothed values. `logit` specifies that the smoothed curve be in terms of the log of the odds ratio:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. lowess foreign mpg, ylabel(0 "Domestic" 1 "Foreign") jitter(5) adjust
```



```
. lowess foreign mpg, logit yline(0)
```



With binary data, if you do not use the `logit` option, it is a good idea to specify `graph's jitter()` option; see [G] [graph twoway scatter](#). Because the underlying data (whether the car was manufactured outside the United States here) take on only two values, raw data points are more likely to be on top of each other, thus making it impossible to tell how many points there are. `graph's jitter()` option adds some noise to the data to shift the points around. This noise affects only the location of points on the graph, not the lowess curve.

When you specify the `logit` option, the display of the raw data is suppressed.



## □ Technical note

`lowess` can be used for more than just lowess smoothing. Lowess can be usefully thought of as a combination of two smoothing concepts: the use of predicted values from regression (rather than means) for imputing a smoothed value and the use of the tricube weighting function (rather than a constant weighting function). `lowess` allows you to combine these concepts freely. You can use line smoothing without weighting (specify `noweight`), mean smoothing with tricube weighting (specify `mean`), or mean smoothing without weighting (specify `mean` and `noweight`).



## Methods and formulas

`lowess` is implemented as an ado-file.

Let  $y_i$  and  $x_i$  be the two variables, and assume that the data are ordered so that  $x_i \leq x_{i+1}$  for  $i = 1, \dots, N - 1$ . For each  $y_i$ , a smoothed value  $y_i^s$  is calculated.

The subset used in calculating  $y_i^s$  is indices  $i_- = \max(1, i - k)$  through  $i_+ = \min(i + k, N)$ , where  $k = \lfloor (N \times \text{bwidth} - 0.5)/2 \rfloor$ . The weights for each of the observations between  $j = i_-, \dots, i_+$  are either 1 (`noweight`) or the tricube (default),

$$w_j = \left\{ 1 - \left( \frac{|x_j - x_i|}{\Delta} \right)^3 \right\}^3$$

where  $\Delta = 1.0001 \max(x_{i_+} - x_i, x_i - x_{i_-})$ . The smoothed value  $y_i^s$  is then the (weighted) mean or the (weighted) regression prediction at  $x_i$ .

William S. Cleveland (1943– ) studied mathematics and statistics at Princeton and Yale. He worked for several years at Bell Labs in New Jersey and now teaches statistics and computer science at Purdue. He has made key contributions in many areas of statistics, including graphics and data visualization, time series, environmental applications, and analysis of Internet traffic data.

## Acknowledgment

`lowess` is a modified version of a command originally written by Patrick Royston of the MRC Clinical Trials Unit, London.

## References

- Chambers, J. M., W. S. Cleveland, B. Kleiner, and P. A. Tukey. 1983. *Graphical Methods for Data Analysis*. Belmont, CA: Wadsworth.
- Cleveland, W. S. 1979. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association* 74: 829–836.

———. 1993. *Visualizing Data*. Summit, NJ: Hobart.

———. 1994. *The Elements of Graphing Data*. Rev. ed. Summit, NJ: Hobart.

Cox, N. J. 2005. [Speaking Stata: Smoothing in various directions](#). *Stata Journal* 5: 574–593.

Goodall, C. 1990. A survey of smoothing techniques. In *Modern Methods of Data Analysis*, ed. J. Fox and J. S. Long, 126–176. Newbury Park, CA: Sage.

Royston, P. 1991. [gr6: Lowess smoothing](#). *Stata Technical Bulletin* 3: 7–9. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 41–44. College Station, TX: Stata Press.

Royston, P., and N. J. Cox. 2005. [A multivariable scatterplot smoother](#). *Stata Journal* 5: 405–412.

Salgado-Ugarte, I. H., and M. Shimizu. 1995. [snp8: Robust scatterplot smoothing: Enhancements to Stata's ksm](#). *Stata Technical Bulletin* 25: 23–26. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 190–194. College Station, TX: Stata Press.

Sasieni, P. 1994. [snp7: Natural cubic splines](#). *Stata Technical Bulletin* 22: 19–22. Reprinted in *Stata Technical Bulletin Reprints*, vol. 4, pp. 171–174. College Station, TX: Stata Press.

## Also see

[\[D\] ipolate](#) — Linearly interpolate (extrapolate) values

[\[R\] smooth](#) — Robust nonlinear smoother

[\[R\] lpoly](#) — Kernel-weighted local polynomial smoothing



Syntax

lpoly yvar xvar [if] [in] [weight] [, options]

| options                                                                                    | description                                                                                                        |
|--------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| Main                                                                                       |                                                                                                                    |
| <u>k</u> ernel( <i>kernel</i> )                                                            | specify kernel function; default is <code>kernel(epanechnikov)</code>                                              |
| <u>b</u> width( <i>#</i>   <i>varname</i> )                                                | specify kernel bandwidth                                                                                           |
| <u>d</u> egree( <i>#</i> )                                                                 | specify degree of the polynomial smooth; default is <code>degree(0)</code>                                         |
| <u>g</u> enerate([ <i>newvar</i> <sub><i>x</i></sub> ] <i>newvar</i> <sub><i>s</i></sub> ) | store smoothing grid in <i>newvar</i> <sub><i>x</i></sub> and smoothed points in <i>newvar</i> <sub><i>s</i></sub> |
| <u>n</u> ( <i>#</i> )                                                                      | obtain the smooth at <i>#</i> points; default is <code>min(<i>N</i>, 50)</code>                                    |
| <u>a</u> t( <i>varname</i> )                                                               | obtain the smooth at the values specified by <i>varname</i>                                                        |
| <u>n</u> ograph                                                                            | suppress graph                                                                                                     |
| <u>n</u> oscatte <u>r</u>                                                                  | suppress scatterplot only                                                                                          |
| SE/CI                                                                                      |                                                                                                                    |
| <u>c</u> i                                                                                 | plot confidence bands                                                                                              |
| <u>l</u> evel( <i>#</i> )                                                                  | set confidence level; default is <code>level(95)</code>                                                            |
| <u>s</u> e( <i>newvar</i> )                                                                | store standard errors in <i>newvar</i>                                                                             |
| <u>p</u> width( <i>#</i> )                                                                 | specify pilot bandwidth for standard error calculation                                                             |
| <u>v</u> ar( <i>#</i>   <i>varname</i> )                                                   | specify estimates of residual variance                                                                             |
| Scatterplot                                                                                |                                                                                                                    |
| <i>marker_options</i>                                                                      | change look of markers (color, size, etc.)                                                                         |
| <i>marker_label_options</i>                                                                | add marker labels; change look or position                                                                         |
| Smoothed line                                                                              |                                                                                                                    |
| <u>l</u> ine <u>o</u> pts( <i>cline_options</i> )                                          | affect rendition of the smoothed line                                                                              |
| CI plot                                                                                    |                                                                                                                    |
| <u>c</u> i <u>o</u> pts( <i>cline_options</i> )                                            | affect rendition of the confidence bands                                                                           |
| Add plots                                                                                  |                                                                                                                    |
| <u>a</u> ddplot( <i>plot</i> )                                                             | add other plots to the generated graph                                                                             |
| Y axis, X axis, Titles, Legend, Overall                                                    |                                                                                                                    |
| <i>twoway_options</i>                                                                      | any options other than <code>by()</code> documented in [G] <i>twoway_options</i>                                   |

| <i>kernel</i>             | description                               |
|---------------------------|-------------------------------------------|
| <code>epanechnikov</code> | Epanechnikov kernel function; the default |
| <code>epan2</code>        | alternative Epanechnikov kernel function  |
| <code>biweight</code>     | biweight kernel function                  |
| <code>cosine</code>       | cosine trace kernel function              |
| <code>gaussian</code>     | Gaussian kernel function                  |
| <code>parzen</code>       | Parzen kernel function                    |
| <code>rectangle</code>    | rectangle kernel function                 |
| <code>triangle</code>     | triangle kernel function                  |

weights and aweights are allowed; see [U] 11.1.6 [weight](#).

## Menu

Statistics > Nonparametric analysis > Local polynomial smoothing

## Description

`lpoly` performs a kernel-weighted local polynomial regression of `yvar` on `xvar` and displays a graph of the smoothed values with (optional) confidence bands.

## Options

Main

- `kernel(kernel)` specifies the kernel function for use in calculating the weighted local polynomial estimate. The default is `kernel(epanechnikov)`.
- `bwidth(#|varname)` specifies the half-width of the kernel—the width of the smoothing window around each point. If `bwidth()` is not specified, a rule-of-thumb (ROT) bandwidth estimator is calculated and used. A local variable bandwidth may be specified in *varname*, in conjunction with an explicit smoothing grid using the `at()` option.
- `degree(#)` specifies the degree of the polynomial to be used in the smoothing. The default is `degree(0)`, meaning local-mean smoothing.
- `generate([newvarx newvars])` stores the smoothing grid in *newvar<sub>x</sub>* and the smoothed values in *newvar<sub>s</sub>*. If `at()` is not specified, then both *newvar<sub>x</sub>* and *newvar<sub>s</sub>* must be specified. Otherwise, only *newvar<sub>s</sub>* is to be specified.
- `n(#)` specifies the number of points at which the smooth is to be calculated. The default is  $\min(N, 50)$ , where  $N$  is the number of observations.
- `at(varname)` specifies a variable that contains the values at which the smooth should be calculated. By default, the smoothing is done on an equally spaced grid, but you can use `at()` to instead perform the smoothing at the observed  $x$ 's, for example. This option also allows you to more easily obtain smooths for different variables or different subsamples of a variable and then overlay the estimates for comparison.
- `nograph` suppresses drawing the graph of the estimated smooth. This option is often used with the `generate()` option.
- `noscatter` suppresses superimposing a scatterplot of the observed data over the smooth. This option is useful when the number of resulting points would be so large as to clutter the graph.

## SE/CI

`ci` plots confidence bands, using the confidence level specified in `level()`.

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see [U] [20.7 Specifying the width of confidence intervals](#).

`se(newvar)` stores the estimates of the standard errors in *newvar*. This option requires specifying `generate()` or `at()`.

`pwidth(#)` specifies the pilot bandwidth to be used for standard-error computations. The default is chosen to be 1.5 times the value of the ROT bandwidth selector. If you specify `pwidth()` without specifying `se()` or `ci`, then the `ci` option is assumed.

`var(# | varname)` specifies an estimate of a constant residual variance or a variable containing estimates of the residual variances at each grid point required for standard-error computation. By default, the residual variance at each smoothing point is estimated by the normalized weighted residual sum of squares obtained from locally fitting a polynomial of order  $p + 2$ , where  $p$  is the degree specified in `degree()`. `var(varname)` is allowed only if `at()` is specified. If you specify `var()` without specifying `se()` or `ci`, then the `ci` option is assumed.

## Scatterplot

*marker\_options* affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [G] [marker\\_options](#).

*marker\_label\_options* specify if and how the markers are to be labeled; see [G] [marker\\_label\\_options](#).

## Smoothed line

`lineopts(cline_options)` affects the rendition of the smoothed line; see [G] [cline\\_options](#).

## CI plot

`ciopts(cline_options)` affects the rendition of the confidence bands; see [G] [cline\\_options](#).

## Add plots

`addplot(plot)` provides a way to add other plots to the generated graph; see [G] [addplot\\_option](#).

## Y axis, X axis, Titles, Legend, Overall

*twoway\_options* are any of the options documented in [G] [twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

## Remarks

Remarks are presented under the following headings:

[Introduction](#)  
[Local polynomial smoothing](#)  
[Choice of a bandwidth](#)  
[Confidence bands](#)

## Introduction

The last 25 years or so has seen a significant outgrowth in the literature on scatterplot smoothing, otherwise known as univariate nonparametric regression. Of most appeal is the idea of making no assumptions about the functional form for the expected value of a response given a regressor, but instead allowing the data to “speak for themselves”. Various methods and estimators fall into the category of nonparametric regression, including local mean smoothing as described independently by [Nadaraya \(1964\)](#) and [Watson \(1964\)](#), the [Gasser and Müller \(1979\)](#) estimator, locally weighted scatterplot smoothing (LOWESS) as described by [Cleveland \(1979\)](#), wavelets (e.g., [Donoho \[1995\]](#)), and splines ([Eubank 1999](#)), to name a few. Much of the vast literature focuses on automating the amount of smoothing to be performed and dealing with the bias/variance tradeoff inherent to this type of estimation. For example, for Nadaraya–Watson the amount of smoothing is controlled by choosing a *bandwidth*.

Smoothing via local polynomials is by no means a new idea but instead one that has been rediscovered in recent years in articles such as [Fan \(1992\)](#). A natural extension of the local mean smoothing of Nadaraya–Watson, local polynomial regression involves fitting the response to a polynomial form of the regressor via locally weighted least squares. Higher-order polynomials have better bias properties than the zero-degree local polynomials of the Nadaraya–Watson estimator; in general, higher-order polynomials do not require bias adjustment at the boundary of the regression space. For a definitive reference on local polynomial smoothing, see [Fan and Gijbels \(1996\)](#).

## Local polynomial smoothing

Consider a set of scatterplot data  $\{(x_1, y_1), \dots, (x_n, y_n)\}$  from the model

$$y_i = m(x_i) + \sigma(x_i)\epsilon_i \quad (1)$$

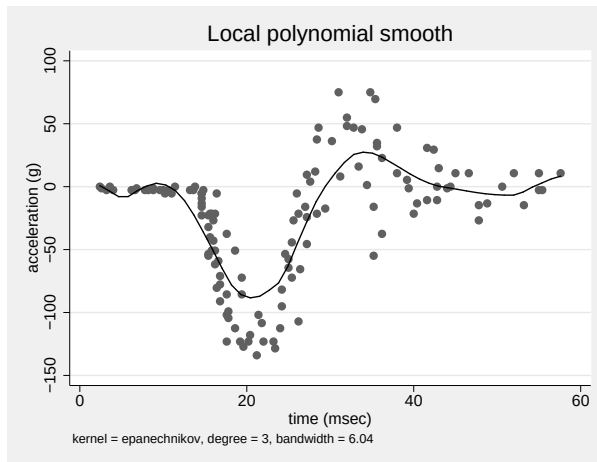
for some unknown mean and variance functions  $m(\cdot)$  and  $\sigma^2(\cdot)$ , and symmetric errors  $\epsilon_i$  with  $E(\epsilon_i) = 0$  and  $\text{Var}(\epsilon_i) = 1$ . The goal is to estimate  $m(x_0) = E[Y|X = x_0]$ , making no assumption about the functional form of  $m(\cdot)$ .

`lpoly` estimates  $m(x_0)$  as the constant term (intercept) of a regression, weighted by the kernel function specified in `kernel()`, of  $yvar$  on the polynomial terms  $(xvar - x_0), (xvar - x_0)^2, \dots, (xvar - x_0)^p$  for each smoothing point  $x_0$ . The degree of the polynomial,  $p$ , is specified in `degree()`, the amount of smoothing is controlled by the bandwidth specified in `bwidth()`, and the chosen kernel function is specified in `kernel()`.

### ► Example 1

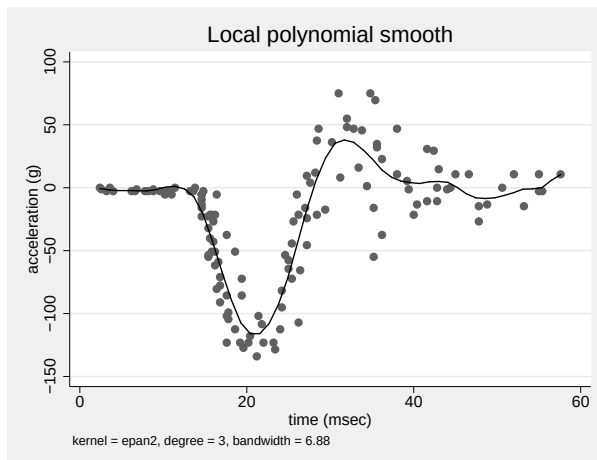
Consider the motorcycle data as examined (among other places) in [Fan and Gijbels \(1996\)](#). The data consist of 133 observations and measure the acceleration (`accel` measured in grams [g]) of a dummy’s head during impact over time (`time` measured in milliseconds). For these data, we use `lpoly` to fit a local cubic polynomial with the default bandwidth (obtained using the ROT method) and the default Epanechnikov kernel.

```
. use http://www.stata-press.com/data/r11/motorcycle
(Motorcycle data from Fan & Gijbels (1996))
. lpoly accel time, degree(3)
```



The default bandwidth and kernel settings do not provide a satisfactory fit in this example. To improve the fit, we can either supply a different bandwidth by using the `bwidth()` option or specify a different kernel by using the `kernel()` option. For example, using the alternative Epanechnikov kernel, `kernel(epan2)`, below provides a better fit for these data.

```
. lpoly accel time, degree(3) kernel(epan2)
```



◀

## □ Technical note

`lpoly` allows specifying in `degree()` both odd and even orders of the polynomial to be used for the smoothing. However, the odd-order,  $2k + 1$ , polynomial approximations are preferable. They have

an extra parameter compared with the even-order,  $2k$ , approximations, which leads to a significant bias reduction and there is no increase of variability associated with adding this extra parameter. Using an odd order when estimating the regression function is therefore usually sufficient. For a more thorough discussion, see [Fan and Gijbels \(1996\)](#).



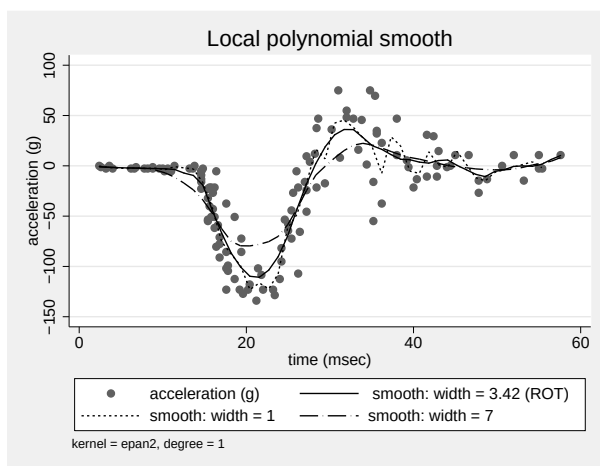
## Choice of a bandwidth

The choice of a bandwidth is crucial for many smoothing techniques, including local polynomial smoothing. In general, using a large bandwidth gives smooths with a large bias, whereas a small bandwidth may result in highly variable smoothed values. Various techniques exist for optimal bandwidth selection. By default, `lpoly` uses the ROT method to estimate the bandwidth used for the smoothing; see [Methods and formulas](#) for details.

### ► Example 2

Using the motorcycle data, we demonstrate how a local linear polynomial fit changes using different bandwidths.

```
. lpoly accel time, degree(1) kernel(epan2) bwidth(1) generate(at smooth1)
> nograph
. lpoly accel time, degree(1) kernel(epan2) bwidth(7) at(at) generate(smooth2)
> nograph
. label variable smooth1 "smooth: width = 1"
. label variable smooth2 "smooth: width = 7"
. lpoly accel time, degree(1) kernel(epan2) at(at) addplot(line smooth* at)
> legend(label(2 "smooth: width = 3.42 (ROT)") note("kernel = epan2, degree = 1"))
```

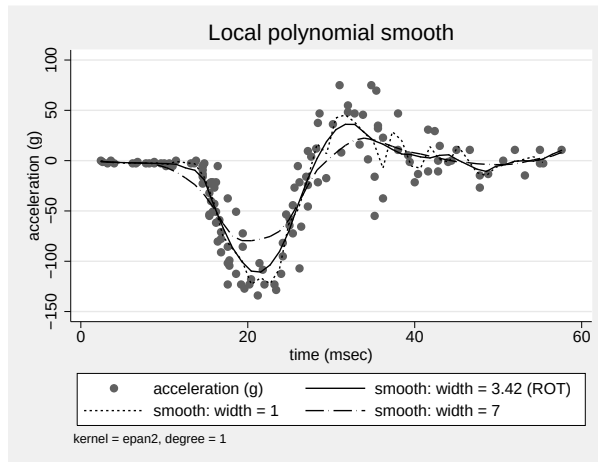


From this graph, we can see that the local linear polynomial fit with larger bandwidth (`width = 7`) corresponds to a smoother line but fails to fit the curvature of the scatterplot data. The smooth obtained using the width equal to one seems to fit most data points, but the corresponding line has several spikes indicating larger variability. The smooth obtained using the ROT bandwidth estimator seems to have a good tradeoff between the fit and variability in this example.

In the above, we also demonstrated how the `generate()` and `addplot()` options may be used to produce overlaid plots obtained from `lpoly` with different options. The `nograph` option saves time when you need to save only results with `generate()`.

However, to avoid generating variables manually, one can use `twoway lpoly` instead; see [G] [graph twoway lpoly](#) for more details.

```
. twoway scatter accel time ||
> lpoly accel time, degree(1) kernel(epan2) lpattern(solid) ||
> lpoly accel time, degree(1) kernel(epan2) bwidth(1) ||
> lpoly accel time, degree(1) kernel(epan2) bwidth(7) ||
> , legend(label(2 "smooth: width = 3.42 (ROT)") label(3 "smooth: width = 1")
> label(4 "smooth: width = 7"))
> title("Local polynomial smooth") note("kernel = epan2, degree = 1")
> xtitle("time (msec)") ytitle("acceleration (g)")
```



4

The ROT estimate is commonly used as an initial guess for the amount of smoothing; this approach may be sufficient when the choice of a bandwidth is less important. In other cases, you can pick your own bandwidth.

When the shape of the regression function has a combination of peaked and flat regions, a variable bandwidth may be preferable over the constant bandwidth to allow for different degrees of smoothness in different regions. The `bwidth()` option allows you to specify the values of the local variable bandwidths as those stored in a variable in your data.

Similar issues with bias and variability arise when choosing a pilot bandwidth (the `pwidth()` option) used to compute standard errors of the local polynomial smoother. The default value is chosen to be  $1.5 \times \text{ROT}$ . For a review of methods for pilot bandwidth selection, see [Fan and Gijbels \(1996\)](#).

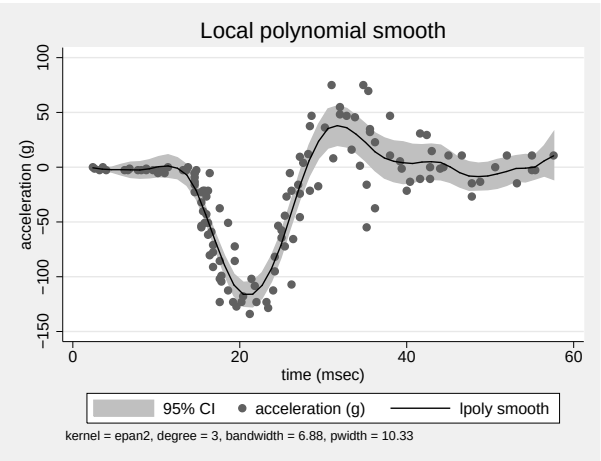
## Confidence bands

The established asymptotic normality of the local polynomial estimators under certain conditions allows the construction of approximate confidence bands. `lpoly` offers the `ci` option to plot these bands.

➤ Example 3

Let us plot the confidence bands for the local polynomial fit from example 1.

```
. lpoly accel time, degree(3) kernel(epan2) ci
```



You can obtain graphs with overlaid confidence bands by using `twoway lpolyci`; see [\[G\] graph twoway lpolyci](#) for examples.

Constructing the confidence intervals involves computing standard errors obtained by taking a square root of the estimate of the conditional variance of the local polynomial estimator at each grid point  $x_0$ . Estimating the conditional variance requires fitting a polynomial of a higher order locally by using a different bandwidth, the pilot bandwidth. The value of the pilot bandwidth may be supplied by using `pwidth()`. By default, the value of  $1.5 \times \text{ROT}$  is used. Also, estimates of the residual variance  $\sigma^2(x_0)$  at each grid point,  $x_0$ , are required to obtain the estimates of the conditional variances. These estimates may be supplied by using the `var()` option. By default, they are computed using the normalized weighted residual sum of squares from a local polynomial fit of a higher order. See [Methods and formulas](#) for details. The standard errors may be saved by using `se()`.

Saved results

`lpoly` saves the following in `r()`:

|                        |                                  |                        |                         |
|------------------------|----------------------------------|------------------------|-------------------------|
| Scalars                |                                  |                        |                         |
| <code>r(degree)</code> | smoothing polynomial degree      | <code>r(bwidth)</code> | bandwidth of the smooth |
| <code>r(ngrid)</code>  | number of successful regressions | <code>r(pwidth)</code> | pilot bandwidth         |
| <code>r(N)</code>      | sample size                      |                        |                         |
| Macros                 |                                  |                        |                         |
| <code>r(kernel)</code> | name of kernel                   |                        |                         |



## Methods and formulas

lpoly is implemented as an ado-file.

Consider model (1), written in matrix notation,

$$\mathbf{y} = m(\mathbf{x}) + \epsilon$$

where  $\mathbf{y}$  and  $\mathbf{x}$  are the  $n \times 1$  vectors of scatterplot values,  $\epsilon$  is the  $n \times 1$  vector of errors with zero mean and covariance matrix  $\Sigma = \text{diag}\{\sigma(x_i)\}\mathbf{I}_n$ , and  $m(\cdot)$  and  $\sigma(\cdot)$  are some unknown functions. Define  $m(x_0) = E[Y|X = x_0]$  and  $\sigma^2(x_0) = \text{Var}[Y|X = x_0]$  to be the conditional mean and conditional variance of random variable  $Y$  (residual variance), respectively, for some realization  $x_0$  of random variable  $X$ .

The method of local polynomial smoothing is based on the approximation of  $m(x)$  locally by a  $p$ th order polynomial in  $(x - x_0)$  for some  $x$  in the neighborhood of  $x_0$ . For the scatterplot data  $\{(x_1, y_1), \dots, (x_n, y_n)\}$ , the  $p$ th-order local polynomial smooth  $\hat{m}(x_0)$  is equal to  $\hat{\beta}_0$ , an estimate of the intercept of the weighted linear regression,

$$\hat{\beta} = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{y} \quad (2)$$

where  $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p)^T$  is the vector of estimated regression coefficients (with  $\{\hat{\beta}_j = (j!)^{-1} \hat{m}^{(j)}(x)|_{x=x_0}, j = 0, \dots, p\}$  also representing estimated coefficients from a corresponding Taylor expansion);  $\mathbf{X} = \{(x_i - x_0)^j\}_{i,j=1,0}^{n,p}$  is a design matrix; and  $\mathbf{W} = \text{diag}\{K_h(x_i - x_0)\}_{n \times n}$  is a weighting matrix with weights  $K_h(\cdot)$  defined as  $K_h(x) = h^{-1}K(x/h)$ , with  $K(\cdot)$  being a kernel function and  $h$  defining a bandwidth. The kernels are defined in [Methods and formulas of \[R\] kdensity](#).

The default bandwidth is obtained using the ROT method of bandwidth selection. The ROT bandwidth is the plugin estimator of the asymptotically optimal constant bandwidth. This is the bandwidth that minimizes the conditional weighted mean integrated squared error. The ROT plugin bandwidth selector for the smoothing bandwidth  $h$  is defined as follows; assuming constant residual variance  $\sigma^2(x_0) = \sigma^2$  and odd degree  $p$ :

$$\hat{h} = C_{0,p}(K) \left[ \frac{\hat{\sigma}^2 \int w_0(x) dx}{n \int \{\hat{m}^{(p+1)}(x)\}^2 w_0(x) f(x) dx} \right]^{1/(2p+3)} \quad (3)$$

where  $C_{0,p}(K)$  is a constant, as defined in [Fan and Gijbels \(1996\)](#), that depends on the kernel function  $K(\cdot)$ , and the degree of a polynomial  $p$  and  $w_0$  is chosen to be an indicator function on the interval  $[\min_{\mathbf{x}} + 0.05 \times \text{range}_{\mathbf{x}}, \max_{\mathbf{x}} - 0.05 \times \text{range}_{\mathbf{x}}]$  with  $\min_{\mathbf{x}}$ ,  $\max_{\mathbf{x}}$ , and  $\text{range}_{\mathbf{x}}$  being, respectively, the minimum, maximum, and the range of  $\mathbf{x}$ . To obtain the estimates of a constant residual variance,  $\hat{\sigma}^2$ , and  $(p+1)$ th order derivative of  $m(x)$ , denoted as  $\hat{m}^{(p+1)}(x)$ , a polynomial in  $\mathbf{x}$  of order  $(p+3)$  is fit globally to  $\mathbf{y}$ .  $\hat{\sigma}^2$  is estimated as a standardized residual sum of squares from this fit.

The expression for the asymptotically optimal constant bandwidth used in constructing the ROT bandwidth estimator is derived for the odd-order polynomial approximations. For even-order polynomial fits the expression would depend not only on  $m^{(p+1)}(x)$  but also on  $m^{(p+2)}(x)$  and the design density and its derivative,  $f(x)$  and  $f'(x)$ . Therefore, the ROT bandwidth selector would require estimation of these additional quantities. Instead, for an even-degree  $p$  of the local polynomial, lpoly uses the value of the ROT estimator (3) computed using degree  $p+1$ . As such, for even degrees this is not a plugin estimator of the asymptotically optimal constant bandwidth.

The estimates of the conditional variance of local polynomial estimators are obtained using

$$\widehat{\text{Var}}\{\widehat{m}(x_0)|X = x_0\} = \widehat{\sigma}_m^2(x_0) = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{W}^2 \mathbf{X}) (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \widehat{\sigma}^2(x_0) \quad (4)$$

where  $\widehat{\sigma}^2(x_0)$  is estimated by the normalized weighted residual sum of squares from the  $(p+2)$ th order polynomial fit using pilot bandwidth  $h^*$ .

When the bias is negligible the normal-approximation method yields a  $(1-\alpha) \times 100\%$  confidence interval for  $m(x_0)$ ,

$$\{\widehat{m}(x_0) - z_{(1-\alpha/2)} \widehat{\sigma}_m(x_0), \widehat{m}(x_0) + z_{(1-\alpha/2)} \widehat{\sigma}_m(x_0)\}$$

where  $z_{(1-\alpha/2)}$  is the  $(1-\alpha/2)$ th quantile of the standard Gaussian distribution, and  $\widehat{m}(x_0)$  and  $\widehat{\sigma}_m(x_0)$  are as defined in (2) and (4), respectively.

## References

- Cleveland, W. S. 1979. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association* 74: 829–836.
- Cox, N. J. 2005. Speaking Stata: Smoothing in various directions. *Stata Journal* 5: 574–593.
- Donoho, D. L. 1995. Nonlinear solution of linear inverse problems by wavelet-vaguelette decomposition. *Applied and Computational Harmonic Analysis* 2: 101–126.
- Eubank, R. L. 1999. *Nonparametric Regression and Spline Smoothing*. 2nd ed. New York: Dekker.
- Fan, J. 1992. Design-adaptive nonparametric regression. *Journal of the American Statistical Association* 87: 998–1004.
- Fan, J., and I. Gijbels. 1996. *Local Polynomial Modelling and Its Applications*. London: Chapman & Hall.
- Gasser, T., and H. G. Müller. 1979. Kernel estimation of regression functions. In *Smoothing Techniques for Curve Estimation, Lecture Notes in Mathematics*, ed. T. Gasser and M. Rosenblatt, 23–68. New York: Springer.
- Gutierrez, R. G., J. M. Linhart, and J. S. Pitblado. 2003. From the help desk: Local polynomial regression and Stata plugins. *Stata Journal* 3: 412–419.
- Nadaraya, E. A. 1964. On estimating regression. *Theory of Probability and Its Application* 9: 141–142.
- Sheather, S. J., and M. C. Jones. 1991. A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society, Series B* 53: 683–690.
- Watson, G. S. 1964. Smooth regression analysis. *Sankhyā Series A* 26: 359–372.

## Also see

- [R] [kdensity](#) — Univariate kernel density estimation
- [R] [lowess](#) — Lowess smoothing
- [R] [regress](#) — Linear regression
- [G] [graph twoway lpoly](#) — Local polynomial smooth plots
- [G] [graph twoway lpolyci](#) — Local polynomial smooth plots with CIs

Syntax

```
lrtest modelspec1 [modelspec2] [, options]
```

where *modelspec* is *name* | . | (*namelist*)

where *name* is the name under which estimation results were saved using `estimates store`, and “.” refers to the last estimation results, whether or not these were already stored.

| <i>options</i>      | description                                                        |
|---------------------|--------------------------------------------------------------------|
| <code>stats</code>  | display statistical information about the two models               |
| <code>dir</code>    | display descriptive information about the two models               |
| <code>df (#)</code> | override the automatic degrees-of-freedom calculation; seldom used |
| <code>force</code>  | force testing even when apparently invalid                         |

Menu

Statistics > Postestimation > Tests > Likelihood-ratio test

Description

`lrtest` performs a likelihood-ratio test of the null hypothesis that the parameter vector of a statistical model satisfies some smooth constraint. To conduct the test, both the unrestricted and the restricted models must be fit using the maximum likelihood method (or some equivalent method), and the results of at least one must be stored using `estimates store`.

*modelspec<sub>1</sub>* and *modelspec<sub>2</sub>* specify the restricted and unrestricted model in any order. *modelspec<sub>1</sub>* and *modelspec<sub>2</sub>* cannot have names in common; for example, `lrtest (A B C) (C D E)` is not allowed because both model specifications include *C*. If *modelspec<sub>2</sub>* is not specified, the last estimation result is used; this is equivalent to specifying *modelspec<sub>2</sub>* as a period (.).

`lrtest` supports composite models specified by a parenthesized list of model names. In a composite model, we assume that the log likelihood and dimension (number of free parameters) of the full model are obtained as the sum of the log-likelihood values and dimensions of the constituting models.

`lrtest` provides an important alternative to `test` for models fit via maximum likelihood or equivalent methods.

Options

`stats` displays statistical information about the unrestricted and restricted models, including the information indices of Akaike and Schwarz.

`dir` displays descriptive information about the unrestricted and restricted models; see `estimates dir` in [R] [estimates store](#).

`df(#)` is seldom specified; it overrides the automatic degrees-of-freedom calculation.

`force` forces the likelihood-ratio test calculations to take place in situations where `lrtest` would normally refuse to do so and issue an error. Such situations arise when one or more assumptions of the test are violated, for example, if the models were fit with `vce(robust)`, `vce(cluster clustvar)`, or `pweights`; when the dependent variables in the two models differ; when the null log likelihoods differ; when the samples differ; or when the estimation commands differ. If you use the `force` option, there is no guarantee as to the validity or interpretability of the resulting test.

## Remarks

The standard way to use `lrtest` is to do the following:

1. Fit either the restricted model or the unrestricted model by using one of Stata's estimation commands and then store the results using `estimates store name`.
2. Fit the alternative model (the unrestricted or restricted model) and then type `'lrtest name .'`. `lrtest` determines for itself which of the two models is the restricted model by comparing the degrees of freedom.

Often you may want to store the alternative model with `estimates store name2`, for instance, if you plan additional tests against models yet to be fit. The likelihood-ratio test is then obtained as `lrtest name name2`.

Remarks are presented under the following headings:

*Nested models*

*Composite models*

## Nested models

`lrtest` may be used with any estimation command that reports a log likelihood, including `heckman`, `logit`, `poisson`, `stcox`, and `streg`. You must check that one of the model specifications implies a statistical model that is *nested within* the model implied by the other specification. Usually, this means that both models are fit with the same estimation command (e.g., both are fit by `logit`, with the same dependent variables) and that the set of covariates of one model is a subset of the covariates of the other model. Second, `lrtest` is valid only for models that are fit by maximum likelihood or by some equivalent method, so it does not apply to models that were fit with probability weights or clusters. Specifying the `vce(robust)` option similarly would indicate that you are worried about the valid specification of the model, so you would not use `lrtest`. Third, `lrtest` assumes that under the null hypothesis, the test statistic is (approximately) distributed as chi-squared. This assumption is not true for likelihood-ratio tests of “boundary conditions”, such as tests for the presence of overdispersion or random effects (Gutierrez, Carter, and Drukker 2001).

### ► Example 1

We have data on infants born with low birthweights along with the characteristics of the mother (Hosmer Jr. and Lemeshow 2000; see also [R] [logistic](#)). We fit the following model:

```
. use http://www.stata-press.com/data/r11/lbw
(Hosmer & Lemeshow data)
```

```
. logistic low age lwt i.race smoke ptl ht ui
```

```
Logistic regression
```

```
Number of obs = 189
LR chi2(8) = 33.22
Prob > chi2 = 0.0001
Pseudo R2 = 0.1416
```

```
Log likelihood = -100.724
```

| low   | Odds Ratio | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-------|------------|-----------|-------|-------|----------------------|----------|
| age   | .9732636   | .0354759  | -0.74 | 0.457 | .9061578             | 1.045339 |
| lwt   | .9849634   | .0068217  | -2.19 | 0.029 | .9716834             | .9984249 |
| race  |            |           |       |       |                      |          |
| 2     | 3.534767   | 1.860737  | 2.40  | 0.016 | 1.259736             | 9.918406 |
| 3     | 2.368079   | 1.039949  | 1.96  | 0.050 | 1.001356             | 5.600207 |
| smoke | 2.517698   | 1.00916   | 2.30  | 0.021 | 1.147676             | 5.523162 |
| ptl   | 1.719161   | .5952579  | 1.56  | 0.118 | .8721455             | 3.388787 |
| ht    | 6.249602   | 4.322408  | 2.65  | 0.008 | 1.611152             | 24.24199 |
| ui    | 2.1351     | .9808153  | 1.65  | 0.099 | .8677528             | 5.2534   |

We now wish to test the constraint that the coefficients on `age`, `lwt`, `ptl`, and `ht` are all zero or, equivalently here, that the odds ratios are all 1. One solution is to type

```
. test age lwt ptl ht
(1) [low]age = 0
(2) [low]lwt = 0
(3) [low]ptl = 0
(4) [low]ht = 0

 chi2(4) = 12.38
Prob > chi2 = 0.0147
```

This test is based on the inverse of the information matrix and is therefore based on a quadratic approximation to the likelihood function; see [\[R\] test](#). A more precise test would be to refit the model, applying the proposed constraints, and then calculate the likelihood-ratio test.

We first save the current model:

```
. estimates store full
```

We then fit the constrained model, which here is the model omitting `age`, `lwt`, `ptl`, and `ht`:

```
. logistic low i.race smoke ui
```

```
Logistic regression
```

```
Number of obs = 189
LR chi2(4) = 18.80
Prob > chi2 = 0.0009
Pseudo R2 = 0.0801
```

```
Log likelihood = -107.93404
```

| low   | Odds Ratio | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|-------|------------|-----------|------|-------|----------------------|----------|
| race  |            |           |      |       |                      |          |
| 2     | 3.052746   | 1.498087  | 2.27 | 0.023 | 1.166747             | 7.987382 |
| 3     | 2.922593   | 1.189229  | 2.64 | 0.008 | 1.316457             | 6.488285 |
| smoke | 2.945742   | 1.101838  | 2.89 | 0.004 | 1.415167             | 6.131715 |
| ui    | 2.419131   | 1.047359  | 2.04 | 0.041 | 1.035459             | 5.651788 |

That done, `lrtest` compares this model with the model we previously saved:

```
. lrttest full .
Likelihood-ratio test LR chi2(4) = 14.42
(Assumption: . nested in full) Prob > chi2 = 0.0061
```

Let's compare results. `test` reported that `age`, `lwt`, `ptl`, and `ht` were jointly significant at the 1.5% level; `lrttest` reports that they are significant at the 0.6% level. Given the quadratic approximation made by `test`, we could argue that `lrttest`'s results are more accurate.

`lrttest` explicates the assumption that, from a comparison of the degrees of freedom, it has assessed that the last fit model (`.`) is nested within the model stored as `full`. In other words, `full` is the unconstrained model and `.` is the constrained model.

The names in “(Assumption: . nested in full)” are actually links. Click on a name, and the results for that model are replayed.



Aside: The `nestreg` command provides a simple syntax for performing likelihood-ratio tests for nested model specifications; see [R] [nestreg](#). In the previous example, we fit a full `logistic` model, used `estimates store` to store the `full` model, fit a constrained `logistic` model, and used `lrttest` to report a likelihood-ratio test between two models. To do this with one call to `nestreg`, use the `lrttable` option.

## □ Technical note

`lrttest` determines the degrees of freedom of a model as the rank of the (co)variance matrix  $e(V)$ . There are two issues here. First, the *numerical* determination of the rank of a matrix is a subtle problem that can, for instance, be affected by the scaling of the variables in the model. The rank of a matrix depends on the number of (independent) linear combinations of coefficients that sum exactly to zero. In the world of numerical mathematics, it is hard to tell whether a very small number is really nonzero or is a real zero that happens to be slightly off because of roundoff error from the finite precision with which computers make floating-point calculations. Whether a small number is being classified as one or the other, typically on the basis of a threshold, affects the determined degrees of freedom. Although Stata generally makes sensible choices, it is bound to make mistakes occasionally. The moral of this story is to make sure that the calculated degrees of freedom are as you expect before interpreting the results.



## □ Technical note

A second issue involves `regress` and related commands such as `anova`. Mainly for historical reasons, `regress` does not treat the residual variance,  $\sigma^2$ , the same way that it treats the regression coefficients. Type `estat vce` after `regress`, and you will see the regression coefficients, not  $\hat{\sigma}^2$ . Most estimation commands for models with ancillary parameters (e.g., `streg` and `heckman`) treat all parameters as equals. There is nothing technically wrong with `regress` here; we are usually focused on the regression coefficients, and their estimators are uncorrelated with  $\hat{\sigma}^2$ . But, formally,  $\sigma^2$  adds a degree of freedom to the model, which does not matter if you are comparing two regression models by a likelihood-ratio test. This test depends on the difference in the degrees of freedom, and hence being “off by 1” in each does not matter. But, if you are comparing a regression model with a larger model—e.g., a heteroskedastic regression model fit by `arch`—the automatic determination of the degrees of freedom is incorrect, and you must specify the `df(#)` option.



## ► Example 2

Returning to the low-birthweight data in the first example, we now wish to test that the coefficient on `2.race` is equal to that on `3.race`. The base model is still stored under the name `full`, so we need only fit the constrained model and perform the test. With  $z$  as the index of the logit model, the base model is

$$z = \beta_0 + \beta_1 \text{age} + \beta_2 \text{lwt} + \beta_3 2.\text{race} + \beta_4 3.\text{race} + \dots$$

If  $\beta_3 = \beta_4$ , this can be written as

$$z = \beta_0 + \beta_1 \text{age} + \beta_2 \text{lwt} + \beta_3 (2.\text{race} + 3.\text{race}) + \dots$$

We can fit the constrained model as follows:

```
. constraint 1 2.race = 3.race
. logistic low age lwt i.race smoke ptl ht ui, constraints(1)
Logistic regression Number of obs = 189
 Wald chi2(7) = 25.17
Log likelihood = -100.9997 Prob > chi2 = 0.0007
(1) [low]2.race - [low]3.race = 0
```

|       | low | Odds Ratio | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-------|-----|------------|-----------|-------|-------|----------------------|----------|
| age   |     | .9716799   | .0352638  | -0.79 | 0.429 | .9049649             | 1.043313 |
| lwt   |     | .9864971   | .0064627  | -2.08 | 0.038 | .9739114             | .9992453 |
| race  |     |            |           |       |       |                      |          |
| 2     |     | 2.728186   | 1.080207  | 2.53  | 0.011 | 1.255586             | 5.927907 |
| 3     |     | 2.728186   | 1.080207  | 2.53  | 0.011 | 1.255586             | 5.927907 |
| smoke |     | 2.664498   | 1.052379  | 2.48  | 0.013 | 1.228633             | 5.778414 |
| ptl   |     | 1.709129   | .5924776  | 1.55  | 0.122 | .8663666             | 3.371691 |
| ht    |     | 6.116391   | 4.215585  | 2.63  | 0.009 | 1.58425              | 23.61385 |
| ui    |     | 2.09936    | .9699702  | 1.61  | 0.108 | .8487997             | 5.192407 |

Comparing this model with our original model, we obtain

```
. lrtest full .
Likelihood-ratio test LR chi2(1) = 0.55
(Assumption: . nested in full) Prob > chi2 = 0.4577
```

By comparison, typing `test 2.race=3.race` after fitting our base model results in a significance level of 0.4572. Alternatively, we can first store the restricted model, here using the name `equal`. Next, `lrtest` is invoked specifying the names of the restricted and unrestricted models (we don't care about the order). This time, we also add the option `stats` requesting a table of model statistics, including the model selection indices AIC and BIC.

```
. estimates store equal
. lrtest equal full, stats
Likelihood-ratio test LR chi2(1) = 0.55
(Assumption: equal nested in full) Prob > chi2 = 0.4577
```

| Model | Obs | ll(null) | ll(model) | df | AIC      | BIC      |
|-------|-----|----------|-----------|----|----------|----------|
| equal | 189 | .        | -100.9997 | 8  | 217.9994 | 243.9334 |
| full  | 189 | -117.336 | -100.724  | 9  | 219.448  | 248.6237 |

Note: N=Obs used in calculating BIC; see **[R] BIC note**

Composite models

lrtest supports composite models; that is, models that can be fit by fitting a series of simpler models or by fitting models on subsets of the data. Theoretically, a composite model is one in which the likelihood function,  $L(\theta)$ , of the parameter vector,  $\theta$ , can be written as the product

$$L(\theta) = L_1(\theta_1) \times L_2(\theta_2) \times \cdots \times L_k(\theta_k)$$

of likelihood terms with  $\theta = (\theta_1, \dots, \theta_k)$  a partitioning of the full parameter vector. In such a case, the full-model likelihood  $L(\theta)$  is maximized by maximizing the likelihood terms  $L_j(\theta_j)$  in turn. Obviously,  $\log L(\hat{\theta}) = \sum_{j=1}^k \log L_j(\hat{\theta}_j)$ . The degrees of freedom for the composite model is obtained as the sum of the degrees of freedom of the constituting models.

➤ Example 3

As an example of the application of composite models, we consider a test of the hypothesis that the coefficients of a statistical model do not differ between different portions (“regimes”) of the covariate space. Economists call a test for such a hypothesis a *Chow test*.

We continue the analysis of the data on children of low birthweight by using logistic regression modeling and study whether the regression coefficients are the same among the three races: white, black, and other. A likelihood-ratio Chow test can be obtained by fitting the logistic regression model for each of the races and then comparing the combined results with those of the model previously stored as full. Because the full model included dummies for the three races, this version of the Chow test allows the intercept of the logistic regression model to vary between the regimes (races).

```
. logistic low age lwt smoke ptl ht ui if 1.race, nolog
Logistic regression Number of obs = 96
 LR chi2(6) = 13.86
 Prob > chi2 = 0.0312
Log likelihood = -45.927061 Pseudo R2 = 0.1311
```

| low   | Odds Ratio | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-------|------------|-----------|-------|-------|----------------------|----------|
| age   | .9869674   | .0527757  | -0.25 | 0.806 | .8887649             | 1.096021 |
| lwt   | .9900874   | .0106101  | -0.93 | 0.353 | .9695089             | 1.011103 |
| smoke | 4.208697   | 2.680133  | 2.26  | 0.024 | 1.20808              | 14.66222 |
| ptl   | 1.592145   | .7474264  | 0.99  | 0.322 | .6344379             | 3.995544 |
| ht    | 2.900166   | 3.193537  | 0.97  | 0.334 | .3350554             | 25.1032  |
| ui    | 1.229523   | .9474768  | 0.27  | 0.789 | .2715165             | 5.567715 |

```
. estimates store white
. logistic low age lwt smoke ptl ht ui if 2.race, nolog
Logistic regression Number of obs = 26
 LR chi2(6) = 10.12
 Prob > chi2 = 0.1198
Log likelihood = -12.654157 Pseudo R2 = 0.2856
```

| low   | Odds Ratio | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-------|------------|-----------|-------|-------|----------------------|----------|
| age   | .8735313   | .1377846  | -0.86 | 0.391 | .6412332             | 1.189983 |
| lwt   | .9747736   | .016689   | -1.49 | 0.136 | .9426065             | 1.008038 |
| smoke | 16.50373   | 24.37044  | 1.90  | 0.058 | .9133647             | 298.2083 |
| ptl   | 4.866916   | 9.33151   | 0.83  | 0.409 | .1135573             | 208.5895 |
| ht    | 85.05605   | 214.6382  | 1.76  | 0.078 | .6049308             | 11959.27 |
| ui    | 67.61338   | 133.3313  | 2.14  | 0.033 | 1.417399             | 3225.322 |

```
. estimates store black
```



```
. logistic low age lwt smoke ptl ht ui if 3.race, nolog
Logistic regression Number of obs = 67
 LR chi2(6) = 14.06
 Prob > chi2 = 0.0289
Log likelihood = -37.228444 Pseudo R2 = 0.1589
```

|       | low | Odds Ratio | Std. Err. | z     | P> z  | [95% Conf. Interval] |
|-------|-----|------------|-----------|-------|-------|----------------------|
| age   |     | .9263905   | .0665386  | -1.06 | 0.287 | .8047407 1.06643     |
| lwt   |     | .9724499   | .015762   | -1.72 | 0.085 | .9420424 1.003839    |
| smoke |     | .7979034   | .6340585  | -0.28 | 0.776 | .1680885 3.787586    |
| ptl   |     | 2.845675   | 1.777944  | 1.67  | 0.094 | .8363053 9.682908    |
| ht    |     | 7.767503   | 10.00537  | 1.59  | 0.112 | .6220764 96.98826    |
| ui    |     | 2.925006   | 2.046473  | 1.53  | 0.125 | .7423107 11.52571    |

```
. est store other
```

We are now ready to perform the likelihood-ratio Chow test:

```
. lrtest (full) (white black other), stats
Likelihood-ratio test LR chi2(12) = 9.83
 Prob > chi2 = 0.6310
```

Assumption: (full) nested in (white, black, other)

| Model | Obs | ll(null)  | ll(model) | df | AIC      | BIC      |
|-------|-----|-----------|-----------|----|----------|----------|
| full  | 189 | -117.336  | -100.724  | 9  | 219.448  | 248.6237 |
| white | 96  | -52.85752 | -45.92706 | 7  | 105.8541 | 123.8046 |
| black | 26  | -17.71291 | -12.65416 | 7  | 39.30831 | 48.11499 |
| other | 67  | -44.26039 | -37.22844 | 7  | 88.45689 | 103.8897 |

Note: N=Obs used in calculating BIC; see **[R] BIC note**

We cannot reject the hypothesis that the logistic regression model applies to each of the races at any reasonable significance level. By specifying the `stats` option, we can verify the degrees of freedom of the test:  $12 = 7 + 7 + 7 - 9$ . We can obtain the same test by fitting an expanded model with interactions between all covariates and `race`.

(Continued on next page)

```
. logistic low race##c.(age lwt smoke ptl ht ui)
Logistic regression
Log likelihood = -95.809661
Number of obs = 189
LR chi2(20) = 43.05
Prob > chi2 = 0.0020
Pseudo R2 = 0.1835
```

| low          | Odds Ratio | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|--------------|------------|-----------|-------|-------|----------------------|----------|
| race         |            |           |       |       |                      |          |
| 2            | 99.62137   | 402.0829  | 1.14  | 0.254 | .0365434             | 271578.9 |
| 3            | 100.3769   | 309.586   | 1.49  | 0.135 | .2378638             | 42358.38 |
| age          | .9869674   | .0527757  | -0.25 | 0.806 | .8887649             | 1.096021 |
| lwt          | .9900874   | .0106101  | -0.93 | 0.353 | .9695089             | 1.011103 |
| smoke        | 4.208697   | 2.680133  | 2.26  | 0.024 | 1.20808              | 14.66222 |
| ptl          | 1.592145   | .7474264  | 0.99  | 0.322 | .6344379             | 3.995544 |
| ht           | 2.900166   | 3.193537  | 0.97  | 0.334 | .3350554             | 25.1032  |
| ui           | 1.229523   | .9474768  | 0.27  | 0.789 | .2715165             | 5.567715 |
| race#c.age   |            |           |       |       |                      |          |
| 2            | .885066    | .1474079  | -0.73 | 0.464 | .638569              | 1.226714 |
| 3            | .9386232   | .0840486  | -0.71 | 0.479 | .7875366             | 1.118695 |
| race#c.lwt   |            |           |       |       |                      |          |
| 2            | .9845329   | .0198857  | -0.77 | 0.440 | .9463191             | 1.02429  |
| 3            | .9821859   | .0190847  | -0.93 | 0.355 | .9454839             | 1.020313 |
| race#c.smoke |            |           |       |       |                      |          |
| 2            | 3.921338   | 6.305992  | 0.85  | 0.395 | .167725              | 91.67917 |
| 3            | .1895844   | .1930601  | -1.63 | 0.102 | .025763              | 1.395113 |
| race#c.ptl   |            |           |       |       |                      |          |
| 2            | 3.05683    | 6.034089  | 0.57  | 0.571 | .0638301             | 146.3918 |
| 3            | 1.787322   | 1.396789  | 0.74  | 0.457 | .3863582             | 8.268285 |
| race#c.ht    |            |           |       |       |                      |          |
| 2            | 29.328     | 80.7482   | 1.23  | 0.220 | .1329492             | 6469.623 |
| 3            | 2.678295   | 4.538712  | 0.58  | 0.561 | .0966916             | 74.18702 |
| race#c.ui    |            |           |       |       |                      |          |
| 2            | 54.99155   | 116.4274  | 1.89  | 0.058 | .8672471             | 3486.977 |
| 3            | 2.378976   | 2.476124  | 0.83  | 0.405 | .309335              | 18.29579 |

```
. lrtest full .
Likelihood-ratio test
(Assumption: full nested in .)
LR chi2(12) = 9.83
Prob > chi2 = 0.6310
```

Applying `lrtest` for the full model against the model with all interactions yields the same test statistic and *p*-value as for the full model against the composite model for the three regimes. Here the specification of the model with interactions was convenient, and `logistic` had no problem computing the estimates for the expanded model. In models with more complicated likelihoods, such as Heckman’s selection model (see [\[R\] heckman](#)) or complicated survival-time models (see [\[ST\] streg](#)), fitting the models with all interactions may be numerically demanding and may be much more time consuming than fitting a series of models separately for each regime.

Given the model with all interactions, we could also test the hypothesis of no differences among the regions (races) by a Wald version of the Chow test by using the `testparm` command.

```
. testparm race#c.(age lwt smoke ptl ht ui)
(1) [low]2.race#c.age = 0
(2) [low]3.race#c.age = 0
(3) [low]2.race#c.lwt = 0
(4) [low]3.race#c.lwt = 0
(5) [low]2.race#c.smoke = 0
(6) [low]3.race#c.smoke = 0
(7) [low]2.race#c.ptl = 0
(8) [low]3.race#c.ptl = 0
(9) [low]2.race#c.ht = 0
(10) [low]3.race#c.ht = 0
(11) [low]2.race#c.ui = 0
(12) [low]3.race#c.ui = 0

 chi2(12) = 8.24
Prob > chi2 = 0.7663
```

We conclude that, here, the Wald version of the Chow test is similar to the likelihood-ratio version of the Chow test.

◀

## Saved results

`lrtest` saves the following in `r()`:

| Scalars              |                       |
|----------------------|-----------------------|
| <code>r(p)</code>    | level of significance |
| <code>r(df)</code>   | degrees of freedom    |
| <code>r(chi2)</code> | LR test statistic     |

Programmers wishing their estimation commands to be compatible with `lrtest` should note that `lrtest` requires that the following results be returned:

|                     |                            |
|---------------------|----------------------------|
| <code>e(cmd)</code> | name of estimation command |
| <code>e(ll)</code>  | log-likelihood value       |
| <code>e(V)</code>   | the (co)variance matrix    |
| <code>e(N)</code>   | number of observations     |

`lrtest` also verifies that `e(N)`, `e(ll_0)`, and `e(depvar)` are consistent between two noncomposite models.

## Methods and formulas

`lrtest` is implemented as an ado-file.

Let  $L_0$  and  $L_1$  be the log-likelihood values associated with the full and constrained models, respectively. The test statistic of the likelihood-ratio test is  $LR = -2(L_1 - L_0)$ . If the constrained model is true, LR is approximately  $\chi^2$  distributed with  $d_0 - d_1$  degrees of freedom, where  $d_0$  and  $d_1$  are the model degrees of freedom associated with the full and constrained models, respectively (Greene 2008, 498–500).

`lrtest` determines the degrees of freedom of a model as the rank of `e(V)`, computed as the number of nonzero diagonal elements of `invsym(e(V))`.

## References

- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- Gutierrez, R. G., S. Carter, and D. M. Drukker. 2001. [sg160: On boundary-value likelihood-ratio tests](#). *Stata Technical Bulletin* 60: 15–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 269–273. College Station, TX: Stata Press.
- Hosmer Jr., D. W., and S. Lemeshow. 2000. *Applied Logistic Regression*. 2nd ed. New York: Wiley.
- Kleinbaum, D. G., and M. Klein. 2002. *Logistic Regression: A Self-Learning Text*. 2nd ed. New York: Springer.
- Pérez-Hoyos, S., and A. Tobías. 1999. [sg111: A modified likelihood-ratio test command](#). *Stata Technical Bulletin* 49: 24–25. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 171–173. College Station, TX: Stata Press.
- Wang, Z. 2000. [sg133: Sequential and drop one term likelihood-ratio tests](#). *Stata Technical Bulletin* 54: 46–47. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 332–334. College Station, TX: Stata Press.

## Also see

- [R] [test](#) — Test linear hypotheses after estimation
- [R] [testnl](#) — Test nonlinear hypotheses after estimation
- [R] [nestreg](#) — Nested model statistics

Syntax

```
lv [varlist] [if] [in] [, generate tail(#)]

by is allowed; see [D] by.
```

Menu

Statistics > Summaries, tables, and tests > Distributional plots and tests > Letter-value display

Description

lv shows a letter-value display (Tukey 1977, 44–49; Hoaglin 1983) for each variable in *varlist*. If no variables are specified, letter-value displays are shown for each numeric variable in the data.

Options

Main

**generate** adds four new variables to the data: `_mid`, containing the midsummaries; `_spread`, containing the spreads; `_psigma`, containing the pseudosigmas; and `_z2`, containing the squared values from a standard normal distribution corresponding to the particular letter value. If the variables `_mid`, `_spread`, `_psigma`, and `_z2` already exist, their contents are replaced. At most, only the first 11 observations of each variable are used; the remaining observations contain missing. If *varlist* specifies more than one variable, the newly created variables contain results for the last variable specified. The **generate** option may not be used with the **by** prefix.

**tail(#)** indicates the inverse of the tail density through which letter values are to be displayed: 2 corresponds to the median (meaning half in each tail), 4 to the fourths (roughly the 25th and 75th percentiles), 8 to the eighths, and so on. # may be specified as 4, 8, 16, 32, 64, 128, 256, 512, or 1,024 and defaults to a value of # that has corresponding depth just greater than 1. The default is taken as 1,024 if the calculation results in a number larger than 1,024. Given the intelligent default, this option is rarely specified.

Remarks

Letter-value displays are a collection of observations drawn systematically from the data, focusing especially on the tails rather than the middle of the distribution. The displays are called letter-value displays because letters have been (almost arbitrarily) assigned to tail densities:

| Letter | Tail area | Letter | Tail area |
|--------|-----------|--------|-----------|
| M      | 1/2       | B      | 1/64      |
| F      | 1/4       | A      | 1/128     |
| E      | 1/8       | Z      | 1/256     |
| D      | 1/16      | Y      | 1/512     |
| C      | 1/32      | X      | 1/1024    |

➤ Example 1

We have data on the mileage ratings of 74 automobiles. To obtain a letter-value display, we type

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. lv mpg
```

| #           | 74   | Mileage (mpg) |       |      |         |             |
|-------------|------|---------------|-------|------|---------|-------------|
| M           | 37.5 |               | 20    |      | spread  | pseudosigma |
| F           | 19   | 18            | 21.5  | 25   | 7       | 5.216359    |
| E           | 10   | 15            | 21.5  | 28   | 13      | 5.771728    |
| D           | 5.5  | 14            | 22.25 | 30.5 | 16.5    | 5.576303    |
| C           | 3    | 14            | 24.5  | 35   | 21      | 5.831039    |
| B           | 2    | 12            | 23.5  | 35   | 23      | 5.732448    |
| A           | 1.5  | 12            | 25    | 38   | 26      | 6.040635    |
|             | 1    | 12            | 26.5  | 41   | 29      | 6.16562     |
| inner fence |      | 7.5           |       | 35.5 | # below | # above     |
| outer fence |      | -3            |       | 46   | 0       | 1           |
|             |      |               |       |      | 0       | 0           |

The decimal points can be made to line up and thus the output made more readable by specifying a display format for the variable; see [\[U\] 12.5 Formats: Controlling how data are displayed](#).

```
. format mpg %9.2f

. lv mpg
```

| #           | 74   | Mileage (mpg) |       |       |         |             |
|-------------|------|---------------|-------|-------|---------|-------------|
| M           | 37.5 |               | 20.00 |       | spread  | pseudosigma |
| F           | 19   | 18.00         | 21.50 | 25.00 | 7.00    | 5.22        |
| E           | 10   | 15.00         | 21.50 | 28.00 | 13.00   | 5.77        |
| D           | 5.5  | 14.00         | 22.25 | 30.50 | 16.50   | 5.58        |
| C           | 3    | 14.00         | 24.50 | 35.00 | 21.00   | 5.83        |
| B           | 2    | 12.00         | 23.50 | 35.00 | 23.00   | 5.73        |
| A           | 1.5  | 12.00         | 25.00 | 38.00 | 26.00   | 6.04        |
|             | 1    | 12.00         | 26.50 | 41.00 | 29.00   | 6.17        |
| inner fence |      | 7.50          |       | 35.50 | # below | # above     |
| outer fence |      | -3.00         |       | 46.00 | 0       | 1           |
|             |      |               |       |       | 0       | 0           |

At the top, the number of observations is indicated as 74. The first line shows the statistics associated with M, the letter value that puts half the density in each tail, or the median. The median has *depth* 37.5 (that is, in the ordered data, M is 37.5 observations in from the extremes) and has value 20. The next line shows the statistics associated with F or the fourths. The fourths have depth 19 (that is, in the ordered data, the lower fourth is observation 19, and the upper fourth is observation 74 – 19 + 1), and the values of the lower and upper fourths are 18 and 25. The number in the middle is the point halfway between the fourths—called a midsummary. If the distribution were perfectly symmetric, the midsummary would equal the median. The spread is the difference between the lower and upper summaries (25 – 18 = 7). For fourths, half the data lie within a 7-mpg band. The pseudosigma is a calculation of the standard deviation using only the lower and upper summaries and assuming that the variable is normally distributed. If the data really were normally distributed, all the pseudosigmas would be roughly equal.

After the letter values, the line labeled with depth 1 reports the minimum and maximum values. Here the halfway point between the extremes is 26.5, which is greater than the median, indicating that 41 is more extreme than 12, at least relative to the median. And with each letter value, the

midsummaries are increasing—our data are skewed. The pseudosigmas are also increasing, indicating that the data are spreading out relative to a normal distribution, although, given the evident skewness, this elongation may be an artifact of the skewness.

At the end is an attempt to identify outliers, although the points so identified are merely outside some predetermined cutoff. Points outside the inner fence are called *outside values* or *mild outliers*. Points outside the outer fence are called *severe outliers*. The inner fence is defined as  $(3/2)\text{IQR}$  and the outer fence as  $3\text{IQR}$  above and below the F summaries, where the interquartile range (IQR) is the spread of the fourths.



□ Technical note

The form of the letter-value display has varied slightly with different authors.  $1v$  displays appear as described by [Hoaglin \(1983\)](#) but as modified by [Emerson and Stoto \(1983\)](#), where they included the midpoint of each of the spreads. This format was later adopted by [Hoaglin \(1985\)](#). If the distribution is symmetric, the midpoints will all be roughly equal. On the other hand, if the midpoints vary systematically, the distribution is skewed.

The pseudosigmas are obtained from the lower and upper summaries for each letter value. For each letter value, they are the standard deviation a normal distribution would have if its spread for the given letter value were to equal the observed spread. If the pseudosigmas are all roughly equal, the data are said to have *neutral elongation*. If the pseudosigmas increase systematically, the data are said to be more elongated than a normal, i.e., have thicker tails. If the pseudosigmas decrease systematically, the data are said to be less elongated than a normal, i.e., have thinner tails.

Interpretation of the number of mild and severe outliers is more problematic. The following discussion is drawn from [Hamilton \(1991\)](#):

Obviously, the presence of any such outliers does not rule out that the data have been drawn from a normal distribution; in large datasets, there will most certainly be observations outside  $(3/2)\text{IQR}$  and  $3\text{IQR}$ . Severe outliers, however, make up about two per million (0.0002%) of a normal population. In samples, they lie far enough out to have substantial effects on means, standard deviations, and other classical statistics. The 0.0002%, however, should be interpreted carefully; outliers appear more often in small samples than one might expect from population proportions because of sampling variation in estimated quartiles. Monte Carlo simulation by [Hoaglin, Iglewicz, and Tukey \(1986\)](#) obtained these results on the percentages and numbers of outliers in random samples from a normal population:

| n   | percentage |        | number |        |
|-----|------------|--------|--------|--------|
|     | any        | severe | any    | severe |
| 10  | 2.83       | .362   | .283   | .0362  |
| 20  | 1.66       | .074   | .332   | .0148  |
| 50  | 1.15       | .011   | .575   | .0055  |
| 100 | .95        | .002   | .95    | .002   |
| 200 | .79        | .001   | 1.58   | .002   |
| 300 | .75        | .001   | 2.25   | .003   |
| ∞   | .70        | .0002  | ∞      | ∞      |

Thus the presence of any severe outliers in samples of less than 300 is sufficient to reject normality. [Hoaglin, Iglewicz, and Tukey \(1981\)](#) suggested the approximation  $0.00698 + 0.4/n$  for the fraction of mild outliers in a sample of size  $n$  or, equivalently,  $0.00698n + 0.4$  for the number of outliers.



➤ Example 2

The `generate` option adds the `_mid`, `_spread`, `_psigma`, and `_z2` variables to our data, making possible many of the diagnostic graphs suggested by [Hoaglin \(1985\)](#).

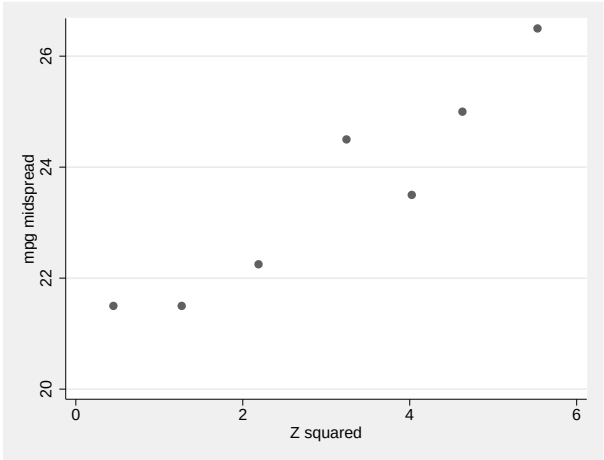
```
. lv mpg, generate
(output omitted)
. list _mid _spread _psigma _z2 in 1/12
```

|     | _mid  | _spread | _psigma  | _z2      |
|-----|-------|---------|----------|----------|
| 1.  | 20    | .       | .        | .        |
| 2.  | 21.5  | 7       | 5.216359 | .4501955 |
| 3.  | 21.5  | 13      | 5.771728 | 1.26828  |
| 4.  | 22.25 | 16.5    | 5.576303 | 2.188846 |
| 5.  | 24.5  | 21      | 5.831039 | 3.24255  |
| 6.  | 23.5  | 23      | 5.732448 | 4.024532 |
| 7.  | 25    | 26      | 6.040635 | 4.631499 |
| 8.  | .     | .       | .        | .        |
| 9.  | .     | .       | .        | .        |
| 10. | .     | .       | .        | .        |
| 11. | 26.5  | 29      | 6.16562  | 5.53073  |
| 12. | .     | .       | .        | .        |

Observations 12 through the end are missing for these new variables. The definition of the observations is always the same. The first observation contains the M summary; the second, the F; the third, the E; and so on. Observation 11 always contains the summary for depth 1. Observations 8–10—corresponding to letter values Z, Y, and X—contain missing because these statistics were not calculated. We have only 74 observations, and their depth would be 1.

[Hoaglin \(1985\)](#) suggests graphing the midsummary against  $z^2$ . If the distribution is not skewed, the points in the resulting graph will be along a horizontal line:

```
. scatter _mid _z2
```



The graph clearly indicates the skewness of the distribution. We might also graph `_psigma` against `_z2` to examine elongation.



## Saved results

lv saves the following in `r()`:

Scalars

|                        |                        |                     |              |
|------------------------|------------------------|---------------------|--------------|
| <code>r(N)</code>      | number of observations | <code>r(u_C)</code> | upper 32nd   |
| <code>r(min)</code>    | minimum                | <code>r(l_B)</code> | lower 64th   |
| <code>r(max)</code>    | maximum                | <code>r(u_B)</code> | upper 64th   |
| <code>r(median)</code> | median                 | <code>r(l_A)</code> | lower 128th  |
| <code>r(l_F)</code>    | lower 4th              | <code>r(u_A)</code> | upper 128th  |
| <code>r(u_F)</code>    | upper 4th              | <code>r(l_Z)</code> | lower 256th  |
| <code>r(l_E)</code>    | lower 8th              | <code>r(u_Z)</code> | upper 256th  |
| <code>r(u_E)</code>    | upper 8th              | <code>r(l_Y)</code> | lower 512th  |
| <code>r(l_D)</code>    | lower 16th             | <code>r(u_Y)</code> | upper 512th  |
| <code>r(u_D)</code>    | upper 16th             | <code>r(l_X)</code> | lower 1024th |
| <code>r(l_C)</code>    | lower 32nd             | <code>r(u_X)</code> | upper 1024th |

The lower/upper 8ths, 16ths, . . . , 1024ths will be defined only if there are sufficient data.

## Methods and formulas

lv is implemented as an ado-file.

Let  $N$  be the number of (nonmissing) observations on  $x$ , and let  $x_{(i)}$  refer to the ordered data when  $i$  is an integer. Define  $x_{(i+0.5)} = (x_{(i)} + x_{(i+1)})/2$ ; the median is defined as  $x_{\{(N+1)/2\}}$ .

Define  $x_{[d]}$  as the pair of numbers  $x_{(d)}$  and  $x_{(N+1-d)}$ , where  $d$  is called the *depth*. Thus  $x_{[1]}$  refers to the minimum and maximum of the data. Define  $m = (N + 1)/2$  as the depth of the median,  $f = (\lfloor m \rfloor + 1)/2$  as the depth of the fourths,  $e = (\lfloor f \rfloor + 1)/2$  as the depth of the eighths, and so on. Depths are reported on the far left of the letter-value display. The corresponding fourths of the data are  $x_{[f]}$ , the eighths are  $x_{[e]}$ , and so on. These values are reported inside the display. The middle value is defined as the corresponding midpoint of  $x_{[.]}$ . The spreads are defined as the difference in  $x_{[.]}$ .

The corresponding point  $z_i$  on a standard normal distribution is obtained as (Hoaglin 1985, 456–457)

$$z_i = \begin{cases} F^{-1}\{(d_i - 1/3)/(N + 1/3)\} & \text{if } d_i > 1 \\ F^{-1}\{0.695/(N + 0.390)\} & \text{otherwise} \end{cases}$$

where  $d_i$  is the depth of the letter value. The corresponding pseudosigma is obtained as the ratio of the spread to  $-2z_i$  (Hoaglin 1985, 431).

Define  $(F_l, F_u) = x_{[f]}$ . The inner fence has cutoffs  $F_l - \frac{3}{2}(F_u - F_l)$  and  $F_u + \frac{3}{2}(F_u - F_l)$ . The outer fence has cutoffs  $F_l - 3(F_u - F_l)$  and  $F_u + 3(F_u - F_l)$ .

The inner-fence values reported by lv are almost equal to those used by `graph`, `box` to identify outside points. The only difference is that `graph` uses a slightly different definition of fourths, namely, the 25th and 75th percentiles as defined by `summarize`; see [R] [summarize](#).

## References

- Emerson, J. D., and M. A. Stoto. 1983. Transforming data. In *Understanding Robust and Exploratory Data Analysis*, ed. D. C. Hoaglin, F. Mosteller, and J. W. Tukey, 97–128. New York: Wiley.
- Fox, J. 1990. Describing univariate distributions. In *Modern Methods of Data Analysis*, ed. J. Fox and J. S. Long, 58–125. Newbury Park, CA: Sage.
- Hamilton, L. C. 1991. [sed4: Resistant normality check and outlier identification](#). *Stata Technical Bulletin* 3: 15–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 86–90. College Station, TX: Stata Press.
- Hoaglin, D. C. 1983. Letter values: A set of selected order statistics. In *Understanding Robust and Exploratory Data Analysis*, ed. D. C. Hoaglin, F. Mosteller, and J. W. Tukey, 33–57. New York: Wiley.
- . 1985. Using quantiles to study shape. In *Exploring Data Tables, Trends, and Shapes*, ed. D. C. Hoaglin, F. Mosteller, and J. W. Tukey, 417–460. New York: Wiley.
- Hoaglin, D. C., B. Iglewicz, and J. W. Tukey. 1981. Small-sample performance of a resistant rule for outlier detection. In *1980 Proceedings of the Statistical Computing Section*. Washington, DC: American Statistical Association.
- . 1986. Performance of some resistant rules for outlier labeling. *Journal of the American Statistical Association* 81: 991–999.
- Tukey, J. W. 1977. *Exploratory Data Analysis*. Reading, MA: Addison–Wesley.

## Also see

- [\[R\] diagnostic plots](#) — Distributional diagnostic plots
- [\[R\] stem](#) — Stem-and-leaf displays
- [\[R\] summarize](#) — Summary statistics

Syntax

```
margins [marginlist] [if] [in] [weight] [, response_options options]
```

where *marginlist* is a list of factor variables or interactions that appear in the current estimation results. The variables may be typed with or without the `i.` prefix and you may use any factor-variable syntax:

```
. margins i.sex i.group i.sex#i.group
. margins sex group sex#i.group
. margins sex##group
```

| <i>response_options</i>              | description                                                 |
|--------------------------------------|-------------------------------------------------------------|
| Main                                 |                                                             |
| <u>predict</u> ( <i>pred_opt</i> )   | estimate margins for <code>predict</code> , <i>pred_opt</i> |
| <u>expression</u> ( <i>pnl_exp</i> ) | estimate margins for <i>pnl_exp</i>                         |
| <u>dydx</u> ( <i>varlist</i> )       | estimate marginal effect of variables in <i>varlist</i>     |
| <u>eyex</u> ( <i>varlist</i> )       | estimate elasticities of variables in <i>varlist</i>        |
| <u>eydx</u> ( <i>varlist</i> )       | estimate semielasticity— $d(y)/d(\ln x)$                    |
| <u>eydx</u> ( <i>varlist</i> )       | estimate semielasticity— $d(\ln y)/d(x)$                    |
| <u>continuous</u>                    | treat factor-level indicators as continuous                 |

| <i>options</i>              | description                                             |
|-----------------------------|---------------------------------------------------------|
| Main                        |                                                         |
| <u>grand</u>                | add the overall margin; default if no <i>marginlist</i> |
| At                          |                                                         |
| <u>at</u> ( <i>atspec</i> ) | estimate margins at specified values of covariates      |
| <u>atmeans</u>              | estimate margins at the means of covariates             |
| <u>asbalanced</u>           | treat all factor variables as balanced                  |

|                                  |                                                     |
|----------------------------------|-----------------------------------------------------|
| if/in/over                       |                                                     |
| <u>over</u> ( <i>varlist</i> )   | estimate margins at unique values of <i>varlist</i> |
| <u>subpop</u> ( <i>subspec</i> ) | estimate margins for subpopulation                  |

|                                  |                                                                            |
|----------------------------------|----------------------------------------------------------------------------|
| Within                           |                                                                            |
| <u>within</u> ( <i>varlist</i> ) | estimate margins at unique values of the nesting factors in <i>varlist</i> |

|                                     |                                                  |
|-------------------------------------|--------------------------------------------------|
| SE                                  |                                                  |
| <u>vce</u> ( <i>delta</i> )         | estimate SEs using delta method; the default     |
| <u>vce</u> ( <i>unconditional</i> ) | estimate SEs allowing for sampling of covariates |
| <u>nose</u>                         | do not estimate SEs                              |

## Advanced

|                                         |                                                  |
|-----------------------------------------|--------------------------------------------------|
| <code>noweights</code>                  | ignore weights specified in estimation           |
| <code>noesample</code>                  | do not restrict margins to the estimation sample |
| <code>emptycells(<i>empspec</i>)</code> | treatment of empty cells for balanced factors    |
| <code>noestimcheck</code>               | suppress estimability checks                     |
| <code>force</code>                      | estimate margins despite potential problems      |
| <code>chainrule</code>                  | use the chain rule when computing derivatives    |
| <code>nochainrule</code>                | do not use the chain rule                        |

## Reporting

|                         |                                                     |
|-------------------------|-----------------------------------------------------|
| <code>level(#)</code>   | confidence level; default is <code>level(95)</code> |
| <code>vsquish</code>    | suppress blank lines between terms                  |
| <code>noatlegend</code> | suppress legend of fixed covariate values           |
| <code>post</code>       | post margins and their VCE as estimation results    |

---

See `at()` under *Options* for a description of *atspec*.

`fweights`, `awweights`, `iweights`, and `pweights` are allowed; see [U] 11.1.6 [weight](#).

## Menu

Statistics > Postestimation > Marginal means and predictive margins

Statistics > Postestimation > Marginal effects

## Description

Margins are statistics calculated from predictions of a previously fit model at fixed values of some covariates and averaging or otherwise integrating over the remaining covariates.

The `margins` command estimates margins of responses for specified values of covariates and presents the results as a table.

Capabilities include estimated marginal means, least-squares means, average and conditional marginal and partial effects (which may be reported as derivatives or as elasticities), average and conditional adjusted predictions, and predictive margins.

## Options

*Warning: The option descriptions are brief and use jargon. Skip to [Remarks](#) if you are reading about margins for the first time.*

---

Main

---

`predict(pred_opt)` and `expression(pnl_exp)` are mutually exclusive; they specify the response. If neither is specified, the response will be the default prediction that would be produced by `predict` after the underlying estimation command.

`predict(pred_opt)` specifies the option(s) to be specified with the `predict` command to produce the variable that will be used as the response. After estimation by `logistic`, you could specify `predict(xb)` to obtain linear predictions rather than the `predict` command's default, the probabilities.

`expression(pnl_exp)` specifies the response as an expression. See [R] [predictnl](#) for a full description of `pnl_exp`. After estimation by `logistic`, you might specify `expression(exp(predict(xb)))` to use relative odds rather than probabilities as the response. For examples, see [Example 12: Margins of a specified expression](#).

`dydx(varlist)`, `eyex(varlist)`, `dyex(varlist)`, and `eydx(varlist)` request that `margins` report derivatives of the response with respect to `varlist` rather than on the response itself. `eyex()`, `dyex()`, and `eydx()` report derivatives as elasticities; see [Expressing derivatives as elasticities](#).

`continuous` is relevant only when one of `dydx()` or `eydx()` is also specified. It specifies that the levels of factor variables be treated as continuous; see [Derivatives versus discrete differences](#). This option is implied if there is a single-level factor variable specified in `dydx()` or `eydx()`.

`grand` specifies that the overall margin be reported. `grand` is assumed when `marginlist` is empty.

At

`at(atspec)` specifies values for covariates to be treated as fixed.

`at(age=20)` fixes covariate `age` to the value specified. `at()` may be used to fix continuous or factor covariates.

`at(age=20 sex=1)` simultaneously fixes covariates `age` and `sex` at the values specified.

`at(age=(20 30 40 50))` fixes age first at 20, then at 30, . . . `margins` produces separate results for each specified value.

`at(age=(20(10)50))` does the same as `at(age=(20 30 40 50))`; that is, you may specify a `numlist`.

`at((mean) age (median) distance)` fixes the covariates at the summary statistics specified. `at((p25) _all)` fixes all covariates at their 25th percentile values. See [Syntax of at\(\)](#) for the full list of summary-statistic modifiers.

`at((mean) _all (median) x x2=1.2 z=(1 2 3))` is read from left to right, with latter specifiers overriding former ones. Thus all covariates are fixed at their means except for `x` (fixed at its median), `x2` (fixed at 1.2), and `z` (fixed first at 1, then at 2, and finally at 3).

`at((means) _all (asobserved) x2)` is a convenient way to set all covariates except `x2` to the mean.

Multiple `at()` options can be specified, and each will produce a different set of margins.

See [Syntax of at\(\)](#) for more information.

`atmeans` specifies that covariates be fixed at their means and is shorthand for `at((mean) _all)`. `atmeans` differs from `at((mean) _all)` in that `atmeans` will affect subsequent `at()` options. For instance,

```
. margins ..., atmeans at((p25) x) at((p75) x)
```

produces two sets of margins with both sets evaluated at the means of all covariates except `x`.

`asbalanced` is shorthand for `at((asbalanced) _factor)` and specifies that factor covariates be evaluated as though there were an equal number of observations in each level; see [Obtaining margins as though the data were balanced](#). `asbalanced` differs from `at((asbalanced) _factor)` in that `asbalanced` will affect subsequent `at()` options in the same way as `atmeans` does.

## if/in/over

**over**(*varlist*) specifies that separate sets of margins be estimated for the groups defined by *varlist*. The variables in *varlist* must contain nonnegative integer values. The variables need not be covariates in your model. When **over**() is combined with the **vce(unconditional)** option, each group is treated as a subpopulation; see [\[SVY\] subpopulation estimation](#).

**subpop**( [*varname*] [*if exp*] ) is intended for use with the **vce(unconditional)** option. It specifies that margins be estimated for the single subpopulation identified by the indicator variable or by the *if* expression or by both. Zero indicates that the observation be excluded; nonzero, that it be included; and missing value, that it be treated as outside of the population (and so ignored). See [\[SVY\] subpopulation estimation](#) for why **subpop**() is preferred to *if* expressions and *in* ranges when also using **vce(unconditional)**. If **subpop**() is used without **vce(unconditional)**, it is treated merely as an additional *if* qualifier.

## Within

**within**(*varlist*) allows for nested designs. *varlist* contains the nesting variable(s) over which margins are to be estimated. See [Obtaining margins with nested designs](#). As with **over**(*varlist*), when **within**(*varlist*) is combined with **vce(unconditional)**, each level of the variables in *varlist* is treated as a subpopulation.

## SE

**vce(delta)** and **vce(unconditional)** specify how the VCE and, correspondingly, standard errors are calculated.

**vce(delta)** is the default. The delta method is applied to the formula for the response and the VCE of the estimation command. This method assumes that values of the covariates used to calculate the response are given or, if not all covariates are fixed using **at**(), that the data are given.

**vce(unconditional)** specifies that the covariates that are not fixed be treated in a way that accounts for their having been sampled. The VCE is estimated using the linearization method. This method allows for heteroskedasticity or other violations of distributional assumptions and allows for correlation among the observations in the same manner as **vce(robust)** and **vce(cluster ...)**, which may have been specified with the estimation command. This method also accounts for complex survey designs if the data are **svyset**. See [Obtaining margins with survey data and representative samples](#).

**nose** suppresses calculation of the VCE and standard errors. See [Requirements for model specification](#) for an example of the use of this option.

## Advanced

**noweights** specifies that any weights specified on the previous estimation command be ignored by **margins**. By default, **margins** uses the weights specified on the estimator to average responses and to compute summary statistics. If weights are specified on the **margins** command, they override previously specified weights, making it unnecessary to specify **noweights**. The **noweights** option is not allowed after **svy**: estimation when the **vce(unconditional)** option is specified.

**noesample** specifies that **margins** not restrict its computations to the estimation sample used by the previous estimation command. See [Example 15: Margins evaluated out of sample](#).

With the default delta-method VCE, **noesample** margins may be estimated on samples other than the estimation sample; such results are valid under the assumption that the data used are treated as being given.

You can specify `noesample` and `vce(unconditional)` together, but if you do, you should be sure that the data in memory correspond to the original `e(sample)`. To show that you understand that, you must also specify the `force` option. Be aware that making the `vce(unconditional)` calculation on a sample different from the estimation sample would be equivalent to estimating the coefficients on one set of data and computing the scores used by the linearization on another set; see [P] [\\_robust](#).

`emptycells(strict)` and `emptycells(reweight)` are relevant only when the `asbalanced` option is also specified. `emptycells()` specifies how empty cells are handled in interactions involving factor variables that are being treated as balanced; see [Obtaining margins as though the data were balanced](#).

`emptycells(strict)` is the default; it specifies that margins involving empty cells be treated as not estimable.

`emptycells(reweight)` specifies that the effects of the observed cells be increased to accommodate any missing cells. This makes the margin estimable but changes its interpretation. `emptycells(reweight)` is implied when the `within()` option is specified.

`noestimcheck` specifies that `margins` not check for estimability. By default, the requested margins are checked and those found not estimable are reported as such. Nonestimability is usually caused by empty cells. If `noestimcheck` is specified, estimates are computed in the usual way and reported even though the resulting estimates are manipulable, which is to say they can differ across equivalent models having different parameterizations. See [Estimability of margins](#).

`force` instructs `margins` to proceed in some situations where it would otherwise issue an error message because of apparent violations of assumptions. Do not be casual about specifying `force`. You need to understand and fully evaluate the statistical issues. For an example of the use of `force`, see [Using margins after the estimates use command](#).

`chainrule` and `nochainrule` specify whether `margins` uses the chain rule when numerically computing derivatives. You need not specify these options when using `margins` after any official Stata estimator; `margins` will choose the appropriate method automatically.

Specify `nochainrule` after estimation by a user-written command. We recommend using `nochainrule`, even though `chainrule` is usually safe and is always faster. `nochainrule` is safer because it makes no assumptions about how the parameters and covariates join to form the response.

`nochainrule` is implied when the `expression()` option is specified.

---

#### Reporting

`level(#)`; see [R] [estimation options](#).

`vsquish` specifies that the blank space separating factor-variable terms or time-series-operated variables from other variables in the model be suppressed.

`noatlegend` specifies that the legend showing the fixed values of covariates be suppressed.

`post` causes `margins` to behave like a Stata estimation (e-class) command. `margins` posts the vector of estimated margins along with the estimated variance–covariance matrix to `e()`, so you can treat the estimated margins just as you would results from any other estimation command. You could use `test` to perform simultaneous tests of hypotheses on the margins, or you could use `lincom` to create linear combinations. See [Example 10: Testing margins](#).

## Remarks

Remarks are presented under the following headings:

### *Introduction*

#### *Obtaining margins of responses*

*Example 1: A simple case after regress*

*Example 2: A simple case after logistic*

*Example 3: Average response versus response at average*

*Example 4: Multiple margins from one command*

*Example 5: Margins with interaction terms*

*Example 6: Margins with continuous variables*

*Example 7: Margins of continuous variables*

*Example 8: Margins of interactions*

*Example 9: Decomposing margins*

*Example 10: Testing margins*

*Example 11: Margins of a specified prediction*

*Example 12: Margins of a specified expression*

*Example 13: Margins with multiple outcomes (responses)*

*Example 14: Margins with multiple equations*

*Example 15: Margins evaluated out of sample*

#### *Obtaining margins of derivatives of responses (a.k.a. marginal effects)*

*Do not specify marginlist when you mean over()*

*Use at() freely, especially with continuous variables*

*Expressing derivatives as elasticities*

*Derivatives versus discrete differences*

*Example 16: Average marginal effect (partial effects)*

*Example 17: Average marginal effect of all covariates*

*Example 18: Evaluating marginal effects over the response surface*

#### *Obtaining margins with survey data and representative samples*

*Example 19: Inferences for populations, margins of response*

*Example 20: Inferences for populations, marginal effects*

*Example 21: Inferences for populations with svyset data*

### *Standardizing margins*

#### *Obtaining margins as though the data were balanced*

*Balancing using asbalanced*

*Balancing by standardization*

*Balancing nonlinear responses*

*Treating a subset of covariates as balanced*

*Using fvset design*

*Balancing in the presence of empty cells*

#### *Obtaining margins with nested designs*

##### *Introduction*

*Margins with nested designs as though the data were balanced*

*Coding of nested designs*

### *Special topics*

*Requirements for model specification*

*Estimability of margins*

*Manipulability of tests*

*Using margins after the estimates use command*

*Syntax of at()*

*Estimation commands that may be used with margins*

### *Glossary*

## Introduction

**margins** is a postestimation command, a command for use after you have fit a model using an estimation command such as **regress** or **logistic**, or using almost any other estimation command.

**margins** estimates and reports margins of responses and margins of derivatives of responses, also known as marginal effects. A margin is a statistic based on a fitted model in which some of or all the



covariates are fixed. Marginal effects are changes in the response for change in a covariate, which can be reported as a derivative, elasticity, or semielasticity.

## Obtaining margins of responses

What we call margins of responses are also known as predictive margins, adjusted predictions, and recycled predictions. When applied to balanced data, margins of responses are also called estimated marginal means and least-squares means.

A margin is a statistic based on a fitted model calculated over a dataset in which some of or all the covariates are fixed at values different from what they really are. For instance, after a linear regression fit on males and females, the marginal mean (margin of mean) for males is the predicted mean of the dependent variable, where every observation is treated as if it represents a male; thus those observations that in fact do represent males are included, as well as those observations that represent females. The marginal mean for female would be similarly obtained by treating all observations as if they represented females.

In making the calculation, sex is treated as male or female everywhere it appears in the model. The model might be

```
. regress y age bp i.sex sex#c.age sex#c.bp
```

and then, in making the marginal calculation of the mean for males and females, `margins` not only accounts for the direct effect of `i.sex` but also for the indirect effects of `sex#c.age` and `sex#c.bp`.

The response being margined can be any statistic produced by `[R]` **predict**, or any expression of those statistics.

Standard errors are obtained by the delta method, at least by default. The delta method assumes that the values at which the covariates are evaluated to obtain the marginal responses are fixed. When your sample represents a population, whether you are using `svy` or not (see [\[SVY\] svy](#)), you can specify `margins' vce(unconditional)` option and `margins` will produce standard errors that account for the sampling variability of the covariates. Some researchers reserve the term predictive margins to describe this.

The best way to understand `margins` is to see some examples. You can run the following examples yourself if you type

```
. use http://www.stata-press.com/data/r11/margex
(Artificial data for margins)
```

### Example 1: A simple case after regress

```
. regress y i.sex i.group
(output omitted)
. margins sex
```

```
Predictive margins Number of obs = 3000
Model VCE : OLS
Expression : Linear prediction, predict()
```

|     | Delta-method |           | z      | P> z  | [95% Conf. Interval] |          |
|-----|--------------|-----------|--------|-------|----------------------|----------|
|     | Margin       | Std. Err. |        |       |                      |          |
| sex |              |           |        |       |                      |          |
| 0   | 60.56034     | .5781782  | 104.74 | 0.000 | 59.42713             | 61.69355 |
| 1   | 78.88236     | .5772578  | 136.65 | 0.000 | 77.75096             | 80.01377 |

The numbers reported in the “Margin” column are average values of *y*. Based on a linear regression of *y* on *sex* and *group*, 60.6 would be the average value of *y* if everyone in the data were treated as if they were male, and 78.9 would be the average value if everyone were treated as if they were female.

Example 2: A simple case after logistic

margins may be used after almost any estimation command.

```
. logistic outcome i.sex i.group
(output omitted)
. margins sex
Predictive margins Number of obs = 3000
Model VCE : OIM
Expression : Pr(outcome), predict()
```

|     | Delta-method |           | z     | P> z  | [95% Conf. Interval] |          |
|-----|--------------|-----------|-------|-------|----------------------|----------|
|     | Margin       | Std. Err. |       |       |                      |          |
| sex |              |           |       |       |                      |          |
| 0   | .1286796     | .0111424  | 11.55 | 0.000 | .106841              | .1505182 |
| 1   | .1905087     | .0089719  | 21.23 | 0.000 | .1729241             | .2080933 |

The numbers reported in the “Margin” column are average predicted probabilities. Based on a logistic regression of *outcome* on *sex* and *group*, 0.13 would be the average probability of *outcome* if everyone in the data were treated as if they were male, and 0.19 would be the average probability if everyone were treated as if they were female.

margins reports average values after regress and average probabilities after logistic. By default, margins makes tables of whatever it is that predict (see [\[R\] predict](#)) predicts by default. Alternatively, margins can make tables of anything that predict can produce if you use margins’ predict() option; see [Example 11: Margins of a specified prediction](#).

Example 3: Average response versus response at average

In [example 2](#), margins reported average probabilities of *outcome* for *sex* = 0 and *sex* = 1. If we instead wanted the predicted probabilities evaluated at the mean of the covariates, we would specify margins’ atmeans option. We previously typed

```
. logistic outcome i.sex i.group
(output omitted)
. margins sex
(output omitted)
```

and now we type

```
. margins sex, atmeans
Adjusted predictions Number of obs = 3000
Model VCE : OIM
Expression : Pr(outcome), predict()
at
 : 0.sex = .4993333 (mean)
 : 1.sex = .5006667 (mean)
 : 1.group = .3996667 (mean)
 : 2.group = .3726667 (mean)
 : 3.group = .2276667 (mean)
```

|     | Delta-method |           | z     | P> z  | [95% Conf. Interval] |          |
|-----|--------------|-----------|-------|-------|----------------------|----------|
|     | Margin       | Std. Err. |       |       |                      |          |
| sex |              |           |       |       |                      |          |
| 0   | .0966105     | .0089561  | 10.79 | 0.000 | .0790569             | .1141641 |
| 1   | .1508362     | .0118064  | 12.78 | 0.000 | .127696              | .1739764 |

The prediction at the average of the covariates is different from the average of the predictions. The first is the expected probability of a person with average characteristics, a person who, in another problem, might be 3/4 married and have 1.2 children. The second is the average of the probability among actual persons in the data.

When you specify `atmeans` or any other `at` option, `margins` reports the values used for the covariates in the legend above the table. `margins` lists the values for all the covariates, including values it may not use, in the results that follow. In this example, `margins` reported means for `sex` even though those means were not used. They were not used because we asked for the margins of `sex`, so `sex` was fixed first at 0 and then at 1.

If you wish to suppress this legend, specify the `nolegend` option.

#### Example 4: Multiple margins from one command

More than one margin can be reported by just one `margins` command. You can type

```
. margins sex group
```

and doing that is equivalent in terms of the output to typing

```
. margins sex
. margins group
```

When multiple margins are requested on the same command, each is estimated separately. There is, however, a difference when you also specify `margins' post` option. Then the variance–covariance matrix for all margins requested is posted, and that is what allows you to test equality of margins, etc. Testing equality of margins is covered in [Example 10: Testing margins](#).

(Continued on next page)

In any case, below we request margins for `sex` and for `group`.

```
. margins sex group
Predictive margins Number of obs = 3000
Model VCE : OIM
Expression : Pr(outcome), predict()
```

|       | Delta-method |           | z     | P> z  | [95% Conf. Interval] |          |
|-------|--------------|-----------|-------|-------|----------------------|----------|
|       | Margin       | Std. Err. |       |       |                      |          |
| sex   |              |           |       |       |                      |          |
| 0     | .1286796     | .0111424  | 11.55 | 0.000 | .106841              | .1505182 |
| 1     | .1905087     | .0089719  | 21.23 | 0.000 | .1729241             | .2080933 |
| group |              |           |       |       |                      |          |
| 1     | .2826207     | .0146234  | 19.33 | 0.000 | .2539593             | .311282  |
| 2     | .1074814     | .0094901  | 11.33 | 0.000 | .0888812             | .1260817 |
| 3     | .0291065     | .0073417  | 3.96  | 0.000 | .0147169             | .043496  |

Example 5: Margins with interaction terms

The estimation command on which `margins` bases its calculations may contain interaction terms, such as an interaction of `sex` and `group`:

```
. logistic outcome i.sex i.group sex#group
(output omitted)
. margins sex group
Predictive margins Number of obs = 3000
Model VCE : OIM
Expression : Pr(outcome), predict()
```

|       | Delta-method |           | z     | P> z  | [95% Conf. Interval] |          |
|-------|--------------|-----------|-------|-------|----------------------|----------|
|       | Margin       | Std. Err. |       |       |                      |          |
| sex   |              |           |       |       |                      |          |
| 0     | .1561738     | .0132774  | 11.76 | 0.000 | .1301506             | .182197  |
| 1     | .1983749     | .0101546  | 19.54 | 0.000 | .1784723             | .2182776 |
| group |              |           |       |       |                      |          |
| 1     | .3211001     | .0176403  | 18.20 | 0.000 | .2865257             | .3556744 |
| 2     | .1152127     | .0099854  | 11.54 | 0.000 | .0956417             | .1347838 |
| 3     | .0265018     | .0109802  | 2.41  | 0.016 | .0049811             | .0480226 |

We fit the model by typing `logistic outcome i.sex i.group sex#group`, but the meaning would have been the same had we typed `logistic outcome sex##group`.

As mentioned in [example 4](#), the results for `sex` and the results for `group` are calculated independently, and we would have obtained the same results had we typed `margins sex` followed by `margins group`.

The margin for male (`sex = 0`) is 0.16. The probability 0.16 is the average probability if everyone in the data were treated as if `sex = 0`, including `sex = 0` in the main effect and `sex = 0` in the interaction of `sex` with `group`.

Had we specified `margins sex, atmeans`, we would have obtained not average probabilities but the probabilities evaluated at the average. Rather than obtaining 0.16, we would have obtained 0.10

for `sex = 0`. The 0.10 is calculated by taking the fitted model, plugging in `sex = 0` everywhere, and plugging in the average value of the group indicator variables everywhere they are used. That is, rather than treating the group indicators as being (1, 0, 0), (0, 1, 0), or (0, 0, 1) depending on observation, the group indicators are treated as being (0.40, 0.37, 0.23), which are the average values of `group = 1`, `group = 2`, and `group = 3`.

### Example 6: Margins with continuous variables

To the above example, we will add the continuous covariate `age` to the model and then rerun `margins sex group`.

```
. logistic outcome i.sex i.group sex#group age
 (output omitted)
. margins sex group
Predictive margins Number of obs = 3000
Model VCE : OIM
Expression : Pr(outcome), predict()
```

|       | Delta-method |           | z     | P> z  | [95% Conf. Interval] |          |
|-------|--------------|-----------|-------|-------|----------------------|----------|
|       | Margin       | Std. Err. |       |       |                      |          |
| sex   |              |           |       |       |                      |          |
| 0     | .1600644     | .0125653  | 12.74 | 0.000 | .1354368             | .184692  |
| 1     | .1966902     | .0100043  | 19.66 | 0.000 | .1770821             | .2162983 |
| group |              |           |       |       |                      |          |
| 1     | .2251302     | .0123233  | 18.27 | 0.000 | .200977              | .2492834 |
| 2     | .150603      | .0116505  | 12.93 | 0.000 | .1277685             | .1734376 |
| 3     | .0736157     | .0337256  | 2.18  | 0.029 | .0075147             | .1397167 |

Compared with the results presented in [example 5](#), results for `sex` change little, but results for groups 1 and 3 change markedly. The tables differ because now we are adjusting for the continuous covariate `age`, as well as for `sex` and `group`.

We will continue examining interactions in [example 8](#). Because we have added a continuous variable, let's take a detour to explain how to obtain margins for continuous variables and to explain their interpretation.

### Example 7: Margins of continuous variables

Continuing with our example of

```
. logistic outcome i.sex i.group sex#group age
```

let's examine the continuous covariate `age`.

You are not allowed to type `margins age`; doing that will produce an error:

```
. margins age
factor age not found in e(b)
r(111);
```

The message “factor age not found in e(b)” is `margins`' way of saying, “Yes, age might be in the model, but if it is, it is not included as a factor variable; it is in as a continuous variable.” Sometimes, Stata is overly terse. `margins` might also say that because age is continuous there are an infinite number of values at which it could evaluate the margins. At what value(s) should age be fixed? `margins` requires more guidance with continuous covariates. We can provide that guidance by using the `at()` option and typing

```
. margins, at(age=40)
```

To understand why that yields the desired result, let us tell you that if you were to type

```
. margins
```

`margins` would report the overall margin—the margin that holds nothing constant. Because our model is logistic, the average value of the predicted probabilities would be reported. The `at()` option fixes one or more covariates to the value(s) specified and can be used with both factor and continuous variables. Thus, if you typed `margins, at(age=40)`, then `margins` would average over the data the responses for everybody, setting `age=40`. Here is what happens when you type that:

```
. margins, at(age=40)
Predictive margins Number of obs = 3000
Model VCE : OIM
Expression : Pr(outcome), predict()
at : age = 40
```

|       | Delta-method |           | z     | P> z  | [95% Conf. Interval] |          |
|-------|--------------|-----------|-------|-------|----------------------|----------|
|       | Margin       | Std. Err. |       |       |                      |          |
| _cons | .1133603     | .0070731  | 16.03 | 0.000 | .0994972             | .1272234 |

Reported is the margin for `age = 40`, adjusted for the other covariates in our model. If we wanted to obtain the margins for `age 30, 35, 40, 45, and 50`, we could type

```
. margins, at(age=(30 35 40 45 50))
```

or, equivalently,

```
. margins, at(age=(30(5)50))
```

Example 8: Margins of interactions

Our model is

```
. logistic outcome i.sex i.group sex#group age
```

We can obtain the margins of all possible combinations of the levels of `sex` and the levels of `group` by typing

```
. margins sex#group
Predictive margins Number of obs = 3000
Model VCE : OIM
Expression : Pr(outcome), predict()
```

|           | Delta-method |           | z     | P> z  | [95% Conf. Interval] |          |
|-----------|--------------|-----------|-------|-------|----------------------|----------|
|           | Margin       | Std. Err. |       |       |                      |          |
| sex#group |              |           |       |       |                      |          |
| 0 1       | .2379605     | .0237178  | 10.03 | 0.000 | .1914745             | .2844465 |
| 0 2       | .0658294     | .0105278  | 6.25  | 0.000 | .0451953             | .0864636 |
| 0 3       | .0538001     | .0136561  | 3.94  | 0.000 | .0270347             | .0805656 |
| 1 1       | .2158632     | .0112968  | 19.11 | 0.000 | .1937218             | .2380045 |
| 1 2       | .2054406     | .0183486  | 11.20 | 0.000 | .1694781             | .2414032 |
| 1 3       | .085448      | .0533914  | 1.60  | 0.110 | -.0191973            | .1900932 |

The first line in the table reports the marginal probability for `sex = 0` and `group = 1`. That is, it reports the estimated probability if everyone in the data were treated as if they were `sex = 0` and `group = 1`.

Also reported are all the other combinations of `sex` and `group`.

By the way, we could have typed `margins sex#group` even if our fitted model did not include `sex#group`. Estimation is one thing, and asking questions about the nature of the estimates is another. `margins` does, however, require that `i.sex` and `i.group` appear somewhere in the model, because fixing a value outside the model would just produce the grand margin, and you can separately ask for that if you want it by typing `margins` without arguments.

### Example 9: Decomposing margins

We have the model

```
. logistic outcome i.sex i.group sex#group age
```

In [example 6](#), we typed `margins sex` and obtained 0.160 for males and 0.197 for females. We are going to decompose each of those numbers. Let us explain:

1. The margin for males, 0.160, treats everyone as if they were male, and that amounts to simultaneously
  - 1a. treating males as males and
  - 1b. treating females as males.
2. The margin for females, 0.197, treats everyone as if they were female, and that amounts to simultaneously
  - 2a. treating males as females and
  - 2b. treating females as females.

The margins 1a and 1b are the decomposition of 1, and the margins 2a and 2b are the decomposition of 2.

We could obtain 1a and 2a by typing

```
. margins if sex==0, at(sex=(0 1))
```

because the qualifier `if sex==0` would restrict `margins` to running on only the males. Similarly, we could obtain 1b and 2b by typing

```
. margins if sex==1, at(sex=(0 1))
```

*(Continued on next page)*

We run these examples below:

```
. margins if sex==0, at(sex=(0 1))
Predictive margins Number of obs = 1498
Model VCE : OIM
Expression : Pr(outcome), predict()
1._at : sex = 0
2._at : sex = 1
```

|     | Delta-method |           |       |       | [95% Conf. Interval] |          |
|-----|--------------|-----------|-------|-------|----------------------|----------|
|     | Margin       | Std. Err. | z     | P> z  |                      |          |
| _at |              |           |       |       |                      |          |
| 1   | .0794393     | .0062147  | 12.78 | 0.000 | .0672586             | .0916199 |
| 2   | .1335584     | .0127351  | 10.49 | 0.000 | .1085981             | .1585187 |

```
. margins if sex==1, at(sex=(0 1))
Predictive margins Number of obs = 1502
Model VCE : OIM
Expression : Pr(outcome), predict()
1._at : sex = 0
2._at : sex = 1
```

|     | Delta-method |           |       |       | [95% Conf. Interval] |          |
|-----|--------------|-----------|-------|-------|----------------------|----------|
|     | Margin       | Std. Err. | z     | P> z  |                      |          |
| _at |              |           |       |       |                      |          |
| 1   | .2404749     | .0199709  | 12.04 | 0.000 | .2013326             | .2796171 |
| 2   | .2596538     | .0104756  | 24.79 | 0.000 | .2391219             | .2801857 |

Putting together the results from [example 6](#) and the results above, we have

|                                         |       |
|-----------------------------------------|-------|
| Margin treating everybody as themselves | 0.170 |
| Margin treating everybody as male       | 0.160 |
| Margin treating male as male            | 0.079 |
| Margin treating female as male          | 0.240 |
| Margin treating everybody as female     | 0.197 |
| Margin treating male as female          | 0.134 |
| Margin treating female as female        | 0.260 |

Example 10: Testing margins

Continuing with the previous example, it would be interesting to test the equality of 2b and 1b, to test whether the average probability of a positive outcome for females treated as females is equal to that for females treated as males. That test would be different from testing the overall significance of `sex` in our model. The test performed on our model would be a test of whether the probability of a positive outcome differs between males and females when they have equal values of the other covariates. The test of equality of margins is a test of whether the average probabilities differ given the different pattern of values of the other covariates that the two sexes have in our data.

There are three steps required to perform tests on margins. First, you must arrange it so that all the margins of interest are reported by just one `margins` command. Second, you must specify `margins'` `post` option. Third, you perform the test with the `test` command.



In the previous example, we used two commands to obtain our results, namely,

```
. margins if sex==0, at(sex=(0 1))
. margins if sex==1, at(sex=(0 1))
```

We could, however, have obtained the same results by typing just one command:

```
. margins, over(sex) at(sex=(0 1))
```

Performing `margins, over(sex)` first restricts the sample to `sex==0` and then restricts it to `sex==1`, and that is equivalent to the two different `if` conditions that we specified before.

To test whether females treated as females is equal to females treated as males, we will need to type

```
. margins, over(sex) at(sex=(0 1)) post
. test _b[2._at#1.sex] = _b[1._at#1.sex]
```

We admit that the second command may seem to have come out of nowhere. When we specify `post` on the `margins` command, `margins` behaves as if it were an estimation command, which means that 1) it posts its estimates and full VCE to `e()`, 2) it gains the ability to replay results just as any estimation command can, and 3) it gains access to the standard postestimation commands. Item 3 explains why we could use `test`. We learned that we wanted to test `_b[2._at#1.sex]` and `_b[1._at#1.sex]` by replaying the estimation results, but this time with the standard estimation command `coeflegend` option. So what we typed was

```
. margins, over(sex) at(sex=(0 1)) post
. margins, coeflegend
. test _b[2._at#1.sex] = _b[1._at#1.sex]
```

We will let you try `margins, coeflegend` for yourself. The results of running the other two commands are

```
. margins, over(sex) at(sex=(0 1)) post
Predictive margins Number of obs = 3000
Model VCE : OIM
Expression : Pr(outcome), predict()
over : sex
1._at : 0.sex
 sex = 0
 1.sex = 0
 sex = 0
2._at : 0.sex
 sex = 1
 1.sex = 1
 sex = 1
```

|         | Delta-method |           |       |       |                      |          |
|---------|--------------|-----------|-------|-------|----------------------|----------|
|         | Margin       | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
| _at#sex |              |           |       |       |                      |          |
| 1 0     | .0794393     | .0062147  | 12.78 | 0.000 | .0672586             | .0916199 |
| 1 1     | .2404749     | .0199709  | 12.04 | 0.000 | .2013326             | .2796171 |
| 2 0     | .1335584     | .0127351  | 10.49 | 0.000 | .1085981             | .1585187 |
| 2 1     | .2596538     | .0104756  | 24.79 | 0.000 | .2391219             | .2801857 |

```
. test _b[2._at#1.sex] = _b[1._at#1.sex]
(1) - 1bn._at#1.sex + 2._at#1.sex = 0
 chi2(1) = 0.72
 Prob > chi2 = 0.3951
```

Example 11: Margins of a specified prediction

We will fit the model

```
. use http://www.stata-press.com/data/r11/margex
. tobit ycn i.sex i.group sex#group age, ul(90)
```

and we will tell the following story about the variables: We run a peach orchard where we allow people to pick their own peaches. A person receives one empty basket in exchange for \$20, along with the right to enter the orchard. There is no official limit on how many peaches a person can pick, but only 90 peaches will fit into a basket. The dependent variable in the above tobit model, ycn, is the number of peaches picked. We use tobit, a special case of censored-normal regression, because ycn is censored at 90.

After fitting this model, if we type

```
. margins sex
```

we would obtain the margins for males and for females of the uncensored number of peaches picked. We would obtain that because predict after tobit produces the uncensored number by default. To obtain the truncated prediction, we would have to specify predict’s ystar(.,90) option. If we want the margins based on that response, we type

```
. margins sex, predict(ystar(.,90))
```

The results of typing that are

```
. tobit ycn i.sex i.group sex#group age, ul(90)
(output omitted)

. margins sex, predict(ystar(.,90))
Predictive margins Number of obs = 3000
Model VCE : OIM
Expression : E(ycn*|ycn<90), predict(ystar(.,90))
```

|     | Delta-method |           |        |       |                      |          |
|-----|--------------|-----------|--------|-------|----------------------|----------|
|     | Margin       | Std. Err. | z      | P> z  | [95% Conf. Interval] |          |
| sex |              |           |        |       |                      |          |
| 0   | 62.21804     | .5996928  | 103.75 | 0.000 | 61.04266             | 63.39342 |
| 1   | 78.34272     | .455526   | 171.98 | 0.000 | 77.4499              | 79.23553 |

In our previous examples, sex = 1 has designated females, so evidently the females visiting our orchard are better at filling baskets than the men.

Example 12: Margins of a specified expression

Continuing with our peach orchard example and the previously fit model

```
. use http://www.stata-press.com/data/r11/margex
. tobit ycn i.sex i.group sex#group age, ul(90)
```

let’s examine how well our baskets are working for us. What is the proportion of the number of peaches actually picked to the number that would have been picked were the baskets larger? As mentioned in [example 11](#), `predict, ystar(.,90)` produces the expected number picked given the limit of basket size. `predict, xb` would predict the expected number without a limit. We want the ratio of those two predictions. That ratio will measure as a proportion how well the baskets work. Thus we could type

```
. margins sex, expression(predict(ystar(.,90))/predict(xb))
```

That would give us the proportion for everyone treated as male and everyone treated as female, but what we want to know is how well baskets work for true males and true females, so we will type

```
. margins, over(sex) expression(predict(ystar(.,90))/predict(xb))
```

```
. margins, over(sex) expression(predict(ystar(0,90))/predict(xb))
```

```
Predictive margins Number of obs = 3000
Model VCE : OIM
Expression : predict(ystar(0,90))/predict(xb)
over : sex
```

|     | Delta-method |           | z      | P> z  | [95% Conf. Interval] |          |
|-----|--------------|-----------|--------|-------|----------------------|----------|
|     | Margin       | Std. Err. |        |       |                      |          |
| sex |              |           |        |       |                      |          |
| 0   | .9811785     | .0013037  | 752.60 | 0.000 | .9786233             | .9837337 |
| 1   | .9419962     | .0026175  | 359.88 | 0.000 | .936866              | .9471265 |

By the way, we could count the number of peaches saved by the limited basket size during the period of data collection by typing

```
. count
3000
. margins, expression(3000*(predict(xb)-predict(ystar(.,90))))
(output omitted)
```

The number of peaches saved turns outs to be 9,183.

Example 13: Margins with multiple outcomes (responses)

Estimation commands such as `mlogit` and `mprobit` (see [\[R\] mlogit](#) and [\[R\] mprobit](#)) calculate multiple responses, and those multiple responses are reflected in the options available with `predict` after estimation. Obtaining margins for such estimators is thus the same as obtaining margins of a specified prediction, which was demonstrated in [example 11](#). The solution is to include the `predict_opt` that selects the desired response in margins’ `predict(predict_opt)` option.

If we fit the multinomial logistic model

```
. mlogit group i.sex age
```

then to obtain the margins for the probability that `group = 1`, we would type

```
. margins sex, predict(outcome(1))
```

and to obtain the margins for the probability that `group = 3`, we would type

```
. margins sex, predict(outcome(3))
```

We learned about the `outcome(1)` and `outcome(3)` options by looking in [R] [mlogit postestimation](#). For an example using `margins` with a multiple-outcome estimator, see [example 4](#) in [R] [mlogit postestimation](#).

#### Example 14: Margins with multiple equations

Estimation commands such as `mvreg`, `manova`, `sureg`, and `reg3` (see [R] [mvreg](#), [MV] [manova](#), [R] [sureg](#), and [R] [reg3](#)) fit multiple equations. Obtaining margins for such estimators is the same as obtaining margins with multiple outcomes (see [example 13](#)), which in turn is the same as obtaining margins of a specified prediction (see [example 11](#)). You place the relevant option from the estimator's `predict` command into `margins`' `predict(predict_opt)` option.

If we fit the seemingly unrelated regression model

```
. sureg (y = i.sex age) (distance = i.sex i.group)
```

we can obtain the marginal means of `y` for males and females by typing

```
. margins sex, predict(equation(y))
```

and we can obtain the marginal means of `distance` by typing

```
. margins sex, predict(equation(distance))
```

We could obtain the difference between the margins of `y` and `distance` by typing

```
. margins sex, expression(predict(equation(y)) -
> predict(equation(distance)))
```

More examples can be found in [MV] [manova](#) and [MV] [manova postestimation](#).

#### Example 15: Margins evaluated out of sample

You can fit your model on one dataset and use `margins` on another if you specify `margins' noesample` option. Remember that `margins` reports estimated average responses, and, unless you lock all the covariates at fixed values by using the `at()` option, the remaining variables are allowed to vary as they are observed to vary in the data. That is indeed the point of using `margins`. The fitted model provides the basis for adjusting for the remaining variables, and the data provide their values. The predictions produced by `margins` are of interest assuming the data used by `margins` are in some sense interesting or representative. In some cases, you might need to fit your model on one set of data and perform `margins` on another.

In [example 11](#), we fit the model

```
. tobit ycn i.sex i.group sex#group age, ul(90)
```

and we told a story about our peach orchard in which we charged people \$20 to collect a basket of peaches, where baskets could hold at most 90 peaches. Let us now tell you that we believe the data on which we estimated those margins were unrepresentative, or at least, we have a more representative sample stored in another `.dta` file. That dataset includes the demographics of our customers but does not include counts of peaches picked. It is a lot of work counting those peaches.

Thus we will fit our model just as we did previously using the detailed data, but we will bring the other, more representative dataset into memory before issuing the `margins sex, predict(ystar(.,90))` command, and we will add `noesample` to it.

```
. use http://www.stata-press.com/data/r11/margex
(Artificial data for margins)

. tobit ycn i.sex i.group sex#group age, ul(90)
(output omitted)

. use http://www.stata-press.com/data/r11/peach, clear
. margins sex, predict(ystar(.,90)) noesample

Predictive margins Number of obs = 2727
Model VCE : OIM
Expression : E(ycn*|ycn<90), predict(ystar(.,90))
```

|     | Delta-method |           | z      | P> z  | [95% Conf. Interval] |          |
|-----|--------------|-----------|--------|-------|----------------------|----------|
|     | Margin       | Std. Err. |        |       |                      |          |
| sex |              |           |        |       |                      |          |
| 0   | 56.79774     | 1.003727  | 56.59  | 0.000 | 54.83047             | 58.76501 |
| 1   | 75.02146     | .643742   | 116.54 | 0.000 | 73.75975             | 76.28317 |

In [example 12](#), we produced an estimate of the number of peaches saved by the limited-size baskets. We can update that estimate using the new demographic data by typing

```
. count
2727

. margins, exp(2727*(predict(xb)-predict(ystar(.,90)))) noesample
(output omitted)
```

By running the above, we find that the updated number of peaches saved is 6,408.

## Obtaining margins of derivatives of responses (a.k.a. marginal effects)

Derivatives of responses are themselves responses, so everything said above in [Obtaining margins of responses](#) is equally true of derivatives of responses, and every example above could be repeated here substituting the derivative of the response for the response.

Derivatives are of interest because they are an informative way of summarizing fitted results. The change in a response for a change in the covariate is easy to understand and to explain. In simple models, one hardly needs `margins` to assist in obtaining such margins. Consider the simple linear regression

$$y = \beta_0 + \beta_1 \times \text{sex} + \beta_2 \times \text{age} + \epsilon$$

The derivatives of the responses are

$$dy/d(\text{sex}) = \beta_1$$

$$dy/d(\text{age}) = \beta_2$$

The derivatives are the fitted coefficients. How does  $y$  change between males and females? It changes by  $\beta_1$ . How does  $y$  change with age? It changes by  $\beta_2$  per year.

If you make the model a little more complicated, however, the need for margins arises. Consider the model

$$y = \beta_0 + \beta_1 \times \text{sex} + \beta_2 \times \text{age} + \beta_3 \times \text{age}^2 + \epsilon$$

Now the derivative with respect to age is

$$dy/d(\text{age}) = \beta_2 + 2 \times \beta_3 \times \text{age}$$

The change in  $y$  for a change in age itself changes with age, and so to better understand the fitted results, you might want to make a table of the change in  $y$  for a change in age for  $\text{age} = 30$ ,  $\text{age} = 40$ , and  $\text{age} = 50$ . `margins` can do that.

Consider an even more complicated model, such as

$$y = \beta_0 + \beta_1 \times \text{sex} + \beta_2 \times \text{age} + \beta_3 \times \text{age}^2 + \beta_4 \times \text{bp} + \beta_5 \times \text{sex} \times \text{bp} + \beta_6 \times \text{tmt} + \beta_7 \times \text{tmt} \times \text{age} + \beta_8 \times \text{tmt} \times \text{age}^2 + \epsilon \quad (1)$$

The derivatives are

$$dy/d(\text{sex}) = \beta_1 + \beta_5 \times \text{bp}$$

$$dy/d(\text{age}) = \beta_2 + 2 \times \beta_3 \times \text{age} + \beta_7 \times \text{tmt} + 2 \times \beta_8 \times \text{tmt} \times \text{age}$$

$$dy/d(\text{bp}) = \beta_4 + \beta_5 \times \text{sex}$$

$$dy/d(\text{tmt}) = \beta_6 + \beta_7 \times \text{age} + \beta_8 \times \text{age}^2$$

At this point, `margins` becomes indispensable.

### Do not specify `marginlist` when you mean `over()`

`margins` has the same syntax when used with derivatives of responses as when used with responses. To obtain derivatives, one specifies the `dydx()` option. If we wanted to examine the response variable  $dy/d(\text{tmt})$ , we would specify `margins' dydx(tmt)` option. The rest of the `margins` command has the same syntax as ordinarily, although one tends to specify different syntactical elements. For instance, one usually does not specify a *marginlist*. If we type

```
. margins sex, dydx(tmt)
```

we would obtain  $dy/d(\text{tmt})$  calculated first as if everyone were male and then as if everyone were female. At the least, we would probably want to specify

```
. margins sex, dydx(tmt) grand
```

so as also to obtain  $dy/d(\text{tmt})$ , the overall margin, the margin with everyone having their own value of sex. Usually, however, all we want is the overall margin, and because `grand` is the default when the *marginlist* is not specified, we would just type

```
. margins, dydx(tmt)
```

Alternatively, if we were interested in the decomposition by sex, then rather than type `margins sex, dydx(tmt)`, we probably want to type

```
. margins, over(sex) dydx(tmt)
```

This command gives us the average effect of `tmt` for males and again for females rather than the average effect with everyone treated as male and then again with everyone treated as female.

## Use at() freely, especially with continuous variables

Another option one tends to use more often with derivatives of responses than one does with responses is `at()`. Such use is often to better understand or to communicate how the response varies, or, in technical jargon, to explore the nature of the response surface.

For instance, the effect  $dy/d(\text{tmt})$  in (1) is equal to  $\beta_6 + \beta_7 \times \text{age} + \beta_8 \times \text{age}^2$ , and so simply to understand how treatment varies with age, we may want to fix age at various values. We might type

```
. margins, dydx(tmt) over(sex) at(age=(30 40 50))
```

## Expressing derivatives as elasticities

You specify the `dydx(varname)` option on the `margins` command to use  $dy/d(\text{varname})$  as the response variable. If you want that derivative expressed as an elasticity, you can specify `eyex(varname)`, `eydx(varname)`, or `dyex(varname)`. You substitute `e` for `d` where you want an elasticity. The formulas are

$$\text{dydx}() = dy/dx$$

$$\text{eyex}() = dy/dx \times (x/y)$$

$$\text{eydx}() = dy/dx \times (1/y)$$

$$\text{dyex}() = dy/dx \times (x)$$

and the interpretations are

|                       |                                  |                            |
|-----------------------|----------------------------------|----------------------------|
| <code>dydx()</code> : | change in $y$ for a              | change in $x$              |
| <code>eyex()</code> : | proportional change in $y$ for a | proportional change in $x$ |
| <code>eydx()</code> : | proportional change in $y$ for a | change in $x$              |
| <code>dyex()</code> : | change in $y$ for a              | proportional change in $x$ |

As `margins` always does with response functions, calculations are made at the observational level and are then averaged. Let's assume that in observation 5,  $dy/dx = 0.5$ ,  $y = 15$ , and  $x = 30$ ; then

$$\text{dydx}() = 0.5$$

$$\text{eyex}() = 1.0$$

$$\text{eydx}() = 0.03$$

$$\text{dyex}() = 15.0$$

Many social scientists would informally explain the meaning of `eyex() = 1` as “ $y$  increases 100% when  $x$  increases 100%” or as “ $y$  doubles when  $x$  doubles”, although neither statement is literally true. `eyex()`, `eydx()`, and `dyex()` are rates evaluated at a point, just as `dydx()` is a rate, and all such interpretations are valid only for small (infinitesimal) changes in  $x$ . It is true that `eyex() = 1` means  $y$  increases with  $x$  at a rate such that, if the rate were constant,  $y$  would double if  $x$  doubled. This issue of casual interpretation is no different from casually interpreting `dydx()` as if it represents the response to a unit change. It is not necessarily true that `dydx() = 0.5` means that “ $y$  increases by 0.5 if  $x$  increases by 1”. It is true that “ $y$  increases with  $x$  at a rate such that, if the rate were constant,  $y$  would increase by 0.5 if  $x$  increased by 1”.

`dydx()`, `eyex()`, `eydx()`, and `dyex()` may be used with continuous  $x$  variables. `dydx()` and `eydx()` may also be used with factor variables.

## Derivatives versus discrete differences

In (1),

$$y = \beta_0 + \beta_1 \times \text{sex} + \beta_2 \times \text{age} + \beta_3 \times \text{age}^2 + \beta_4 \times \text{bp} + \beta_5 \times \text{sex} \times \text{bp} + \beta_6 \times \text{tmt} \\ + \beta_7 \times \text{tmt} \times \text{age} + \beta_8 \times \text{tmt} \times \text{age}^2 + \epsilon$$

Let us call your attention to the derivatives of  $y$  with respect to age and sex:

$$dy/d(\text{age}) = \beta_2 + 2 \times \beta_3 \times \text{age} + \beta_7 \times \text{tmt} + 2 \times \beta_8 \times \text{tmt} \times \text{age} \quad (2)$$

$$dy/d(\text{sex}) = \beta_1 + \beta_5 \times \text{bp} \quad (3)$$

`age` is presumably a continuous variable and (2) is precisely how `margins` calculates its derivatives when you type `margins, dydx(age)`. `sex`, however, is presumably a factor variable, and `margins` does not necessarily make the calculation using (3) were you to type `margins, dydx(sex)`. We will explain, but let us first clarify what we mean by a continuous and a factor variable. Say that you fit (1) by typing

```
. regress y i.sex age c.age#c.age i.bp bp#sex
> i.tmt tmt#c.age tmt#c.age#c.age
```

It is important that `sex` entered the model as a factor variable. It would not do to type `regress y sex ...` because then `sex` would be a continuous variable, or at least it would be a continuous variable from Stata's point of view. The model estimates would be the same, but `margins`' understanding of those estimates would be a little different. With the model estimated using `i.sex`, `margins` understands that either `sex` is 0 or `sex` is 1. With the model estimated using `sex`, `margins` thinks `sex` is continuous and, for instance, `sex = 1.5` is a possibility.

`margins` calculates `dydx()` differently for continuous and for factor variables. For continuous variables, `margins` calculates  $dy/dx$ . For factor variables, `margins` calculates the discrete first-difference from the base category. To obtain that for `sex`, write down the model and then subtract from it the model evaluated at the base category for `sex`, which is `sex = 0`. If you do that, you will get the same formula as we obtained for the derivative, namely,

$$\text{discrete difference}\{(\text{sex} = 1) - (\text{sex} = 0)\} = \beta_1 + \beta_5 \times \text{bp}$$

We obtain the same formula because our model is linear regression. Outside of linear regression, and outside of linear response functions generally, the discrete difference is not equal to the derivative. The discrete difference is not equal to the derivative for logistic regression, probit, etc. The discrete difference calculation is generally viewed as better for factor variables than the derivative calculation because the discrete difference is what would actually be observed.

If you want the derivative calculation for your factor variables, specify the `continuous` option on the `margins` command.

### Example 16: Average marginal effect (partial effects)

Concerning the title of this example, the way we use the term marginal effect, the effects of factor variables are calculated using discrete first-differences. If you wanted the continuous calculation, you would specify `margins`' `continuous` option in what follows.



```
. use http://www.stata-press.com/data/r11/margex, clear
(Artificial data for margins)
. logistic outcome treatment##group age c.age#c.age treatment#c.age
(output omitted)
. margins, dydx(treatment)
Average marginal effects Number of obs = 3000
Model VCE : OIM
Expression : Pr(outcome), predict()
dy/dx w.r.t. : 1.treatment
```

|             | Delta-method |           | z    | P> z  | [95% Conf. Interval] |          |
|-------------|--------------|-----------|------|-------|----------------------|----------|
|             | dy/dx        | Std. Err. |      |       |                      |          |
| 1.treatment | .0385625     | .0162848  | 2.37 | 0.018 | .0066449             | .0704801 |

Note: dy/dx for factor levels is the discrete change from the base level.

The average marginal effect of treatment on the probability of a positive outcome is 0.039.

### Example 17: Average marginal effect of all covariates

We will continue with the model

```
. logistic outcome treatment##group age c.age#c.age treatment#c.age
```

if we wanted the average marginal effects for all covariates, we would type `margins, dydx(*)` or `margins, dydx(_all)`; they mean the same thing. This is probably the most common way `margins, dydx()` is used.

```
. margins, dydx(*)
Average marginal effects Number of obs = 3000
Model VCE : OIM
Expression : Pr(outcome), predict()
dy/dx w.r.t. : 1.treatment 2.group 3.group age
```

|             | Delta-method |           | z     | P> z  | [95% Conf. Interval] |           |
|-------------|--------------|-----------|-------|-------|----------------------|-----------|
|             | dy/dx        | Std. Err. |       |       |                      |           |
| 1.treatment | .0385625     | .0162848  | 2.37  | 0.018 | .0066449             | .0704801  |
| group       |              |           |       |       |                      |           |
| 2           | -.0776906    | .0181584  | -4.28 | 0.000 | -.1132805            | -.0421007 |
| 3           | -.1505652    | .0400882  | -3.76 | 0.000 | -.2291366            | -.0719937 |
| age         | .0095868     | .0007796  | 12.30 | 0.000 | .0080589             | .0111148  |

Note: dy/dx for factor levels is the discrete change from the base level.

### Example 18: Evaluating marginal effects over the response surface

Continuing with the model

```
. logistic outcome treatment##group age c.age#c.age treatment#c.age
```

What follows maps out the entire response surface of our fitted model. We report the marginal effect of treatment evaluated at  $\text{age} = 20, 30, \dots, 60$ , by each level of group.

```
. margins group, dydx(treatment) at(age=(20(10)60))
Conditional marginal effects Number of obs = 3000
Model VCE : OIM
Expression : Pr(outcome), predict()
dy/dx w.r.t. : 1.treatment

1._at : age = 20
2._at : age = 30
3._at : age = 40
4._at : age = 50
5._at : age = 60
```

|             | Delta-method |           | z     | P> z  | [95% Conf. Interval] |          |
|-------------|--------------|-----------|-------|-------|----------------------|----------|
|             | dy/dx        | Std. Err. |       |       |                      |          |
| 1.treatment |              |           |       |       |                      |          |
| _at#group   |              |           |       |       |                      |          |
| 1 1         | -.0208409    | .0152863  | -1.36 | 0.173 | -.0508014            | .0091196 |
| 1 2         | .009324      | .0059896  | 1.56  | 0.120 | -.0024155            | .0210635 |
| 1 3         | .0006558     | .0048682  | 0.13  | 0.893 | -.0088856            | .0101972 |
| 2 1         | -.0436964    | .0279271  | -1.56 | 0.118 | -.0984325            | .0110397 |
| 2 2         | .0382959     | .0120405  | 3.18  | 0.001 | .014697              | .0618949 |
| 2 3         | .0064564     | .0166581  | 0.39  | 0.698 | -.0261929            | .0391057 |
| 3 1         | -.055676     | .0363191  | -1.53 | 0.125 | -.1268601            | .015508  |
| 3 2         | .1152235     | .0209858  | 5.49  | 0.000 | .074092              | .156355  |
| 3 3         | .0284808     | .0471293  | 0.60  | 0.546 | -.0638908            | .1208524 |
| 4 1         | -.027101     | .0395501  | -0.69 | 0.493 | -.1046177            | .0504158 |
| 4 2         | .2447682     | .0362623  | 6.75  | 0.000 | .1736954             | .315841  |
| 4 3         | .0824401     | .1025028  | 0.80  | 0.421 | -.1184616            | .2833418 |
| 5 1         | .0292732     | .0587751  | 0.50  | 0.618 | -.0859239            | .1444703 |
| 5 2         | .3757777     | .0578106  | 6.50  | 0.000 | .2624709             | .4890844 |
| 5 3         | .1688268     | .1642191  | 1.03  | 0.304 | -.1530368            | .4906904 |

Note: dy/dx for factor levels is the discrete change from the base level.

### Obtaining margins with survey data and representative samples

The standard errors and confidence intervals produced by `margins` are based by default on the delta method applied to the VCE of the current estimates. Delta-method standard errors treat the covariates at which the response is evaluated as given or fixed. Such standard errors are appropriate if you specify `at()` to fix the covariates, and they are appropriate when you are making inferences about groups exactly like your sample whether you specify `at()` or not.

On the other hand, if you have a representative sample of the population or if you have complex survey data and if you want to make inferences about the underlying population, you need to account for the variation in the covariates that would arise in repeated sampling. You do that using `vce(unconditional)`, which invokes a different standard-error calculation based on [Korn and Graubard \(1999\)](#). Syntactically, there are three cases. They all involve specifying the `vce(unconditional)` option on the `margins` command:

1. You have a representative random sample, and you have not `svyset` your data.
- When you fit the model, you need to specify the `vce(robust)` or `vce(cluster clustvar)` option. When you issue the `margins` command, you need to specify the `vce(unconditional)` option.

2. *You have a weighted sample, and you have not svyset your data.*

You need to specify `[pw=weight]` when you fit the model and, of course, specify the `vce(unconditional)` option on the `margins` command. You do not need to specify the weights on the `margins` command because `margins` will obtain them from the estimation results.

3. *You have svyset your data, whether it be a simple random sample or something more complex including weights, strata, sampling units, or poststratification.*

You need to use the `svy` prefix when you fit the model. You need to specify `vce(unconditional)` when you issue the `margins` command. You do not need to respecify the weights.

Even though the data are `svyset`, and even though the estimation was `svy` estimation, `margins` does not default to `vce(unconditional)`. It does not default to `vce(unconditional)` because there are valid reasons to want the data-specific, `vce(delta)` standard-error estimates. Whether you specify `vce(unconditional)` or not, `margins` uses the weights, so you do not need to respecify them even if you are using `vce(unconditional)`.

`vce(unconditional)` is allowed only after estimation with `vce(robust)`, `vce(cluster ...)`, or the `svy` prefix. If the VCE of the current estimates was specified as clustered, so will be the VCE estimates of `margins`. If the estimates were from a survey estimation, the survey settings in the dataset will be used by `margins`.

When you use `vce(unconditional)`, never specify `if exp` or `in range` on the `margins` command; instead, specify the `subpop(if exp)` option. You do that for the usual reasons; see [\[SVY\] subpopulation estimation](#). If you specify `over(varlist)` to examine subgroups, the subgroups will automatically be treated as subpopulations.

## Example 19: Inferences for populations, margins of response

In [example 6](#), we fit the model

```
. logistic outcome i.sex i.group sex#group age
```

and we obtained margins by sex and margins by group,

```
. margins sex group
```

If our data were randomly drawn from the population of interest and we wanted to account for this, we would have typed

```
. logistic outcome i.sex i.group sex#group age, vce(robust)
. margins sex group, vce(unconditional)
```

We do that below:

(Continued on next page)

```
. logistic outcome i.sex i.group sex#group age, vce(robust)
(output omitted)
. margins sex group, vce(unconditional)
Predictive margins Number of obs = 3000
Expression : Pr(outcome), predict()
```

|       | Unconditional |           | z     | P> z  | [95% Conf. Interval] |          |
|-------|---------------|-----------|-------|-------|----------------------|----------|
|       | Margin        | Std. Err. |       |       |                      |          |
| sex   |               |           |       |       |                      |          |
| 0     | .1600644      | .0131685  | 12.16 | 0.000 | .1342546             | .1858743 |
| 1     | .1966902      | .0104563  | 18.81 | 0.000 | .1761963             | .2171841 |
| group |               |           |       |       |                      |          |
| 1     | .2251302      | .0127069  | 17.72 | 0.000 | .200225              | .2500354 |
| 2     | .150603       | .0118399  | 12.72 | 0.000 | .1273972             | .1738088 |
| 3     | .0736157      | .0343188  | 2.15  | 0.032 | .0063522             | .1408793 |

The estimated margins are the same as they were in [example 6](#), but the standard errors and confidence intervals differ, although not by much. Given that we have 3,000 observations in our randomly drawn sample, we should expect this.

Example 20: Inferences for populations, marginal effects

In [example 17](#), we fit a logistic model and then obtained the average marginal effects for all covariates by typing

```
. logistic outcome treatment##group age c.age#c.age treatment#c.age
. margins, dydx(*)
```

To repeat that and also obtain standard errors for our population, we would type

```
. logistic outcome treatment##group age c.age#c.age treatment#c.age,
> vce(robust)
. margins, dydx(*) vce(unconditional)
```

The results are

```
. logistic outcome treatment##group age c.age#c.age treatment#c.age, vce(robust)
(output omitted)
. margins, dydx(*) vce(unconditional)
Average marginal effects Number of obs = 3000
Expression : Pr(outcome), predict()
dy/dx w.r.t. : 1.treatment 2.group 3.group age
```

|             | Unconditional |           | z     | P> z  | [95% Conf. Interval] |           |
|-------------|---------------|-----------|-------|-------|----------------------|-----------|
|             | dy/dx         | Std. Err. |       |       |                      |           |
| 1.treatment | .0385625      | .0163872  | 2.35  | 0.019 | .0064442             | .0706808  |
| group       |               |           |       |       |                      |           |
| 2           | -.0776906     | .0179573  | -4.33 | 0.000 | -.1128863            | -.0424949 |
| 3           | -.1505652     | .0411842  | -3.66 | 0.000 | -.2312848            | -.0698456 |
| age         | .0095868      | .0007814  | 12.27 | 0.000 | .0080553             | .0111183  |

Note: dy/dx for factor levels is the discrete change from the base level.

## Example 21: Inferences for populations with svyset data

See [example 3](#) in [\[SVY\] svy postestimation](#).

## Standardizing margins

A standardized margin is the margin calculated on data different from the data used to fit the model. Typically, the word *standardized* is reserved for situations in which the alternate population is a reference population, which may be real or artificial, and which is treated as fixed.

Say that you work for a hospital and have fit a model of mortality on the demographic characteristics of the hospital's patients. At this stage, were you to type

```
. margins
```

you would obtain the mortality rate for your hospital. You have another dataset, `hstandard.dta`, that contains demographic characteristics of patients across all hospitals along with the population of each hospital recorded in the `pop` variable. You could obtain the expected mortality rate at your hospital if your patients matched the characteristics of the standard population by typing

```
. use http://www.stata-press.com/data/r11/hstandard, clear
. margins [fw=pop], noesample
```

You specified `noesample` because the margin is being calculated on data other than the data used to estimate the model. You specified `[fw=pop]` because the reference dataset you are using included population counts, as many reference datasets do.

## Obtaining margins as though the data were balanced

Here we discuss what are commonly called estimated marginal means or least-squares means. These are margins assuming that all levels of factor variables are equally likely or, equivalently, that the design is balanced. The seminal reference on these margins is [Searle, Speed, and Milliken \(1980\)](#).

In designed experiments, observations are often allocated in a balanced way so that the variances can be easily compared and decomposed. At the Acme Portable Widget Company, they are experimenting with a new machine. The machine has three temperature settings and two pressure settings; a combination of settings will be optimal on any particular day, determined by the weather. At start-up, one runs a quick test and chooses the optimal setting for the day. Across different days, each setting will be used about equally, says the manufacturer.

In experiments with the machine, 10 widgets were collected for stress testing at each of the settings over a six-week period. We wish to know the average stress-test value that can be expected from these machines over a long period.

## Balancing using asbalanced

The data were intended to be balanced, but unfortunately, the stress test sometimes destroys samples before the stress can be measured. Thus even though the experiment was designed to be balanced, the data are not balanced. You specify the `asbalanced` option to estimate the margins as if the data were balanced. We will type

```
. use http://www.stata-press.com/data/r11/acmemanuf, clear
. regress y pressure##temp
. margins, asbalanced
```

So that you can compare the `asbalanced` results with the observed results, we will also include margins without the `asbalanced` option in what follows:

```
. use http://www.stata-press.com/data/r11/acmemanuf, clear
. regress y pressure##temp
 (output omitted)
. margins
Predictive margins Number of obs = 49
Model VCE : OLS
Expression : Linear prediction, predict()
```

|       | Delta-method |           | z     | P> z  | [95% Conf. Interval] |          |
|-------|--------------|-----------|-------|-------|----------------------|----------|
|       | Margin       | Std. Err. |       |       |                      |          |
| _cons | 109.9214     | 1.422629  | 77.27 | 0.000 | 107.1331             | 112.7097 |

```
. margins, asbalanced
Adjusted predictions Number of obs = 49
Model VCE : OLS
Expression : Linear prediction, predict()
at : pressure (asbalanced)
 : temp (asbalanced)
```

|       | Delta-method |           | z     | P> z  | [95% Conf. Interval] |         |
|-------|--------------|-----------|-------|-------|----------------------|---------|
|       | Margin       | Std. Err. |       |       |                      |         |
| _cons | 115.3758     | 1.530199  | 75.40 | 0.000 | 112.3767             | 118.375 |

❑ Technical note

Concerning how `asbalanced` calculations are performed, if a factor variable has  $l$  levels, then each level’s coefficient contributes to the response weighted by  $1/l$ . If two factors,  $a$  and  $b$ , interact, then each coefficient associated with their interaction is weighted by  $1/(l_a \times l_b)$ .

If a balanced factor interacts with a continuous variable, then each coefficient in the interaction is applied to the value of the continuous variable, and the results are weighted equally. So, if the factor being interacted has  $l_a$  levels, the effect of each coefficient on the value of the continuous covariate is weighted by  $1/l_a$ .



Balancing by standardization

To better understand the balanced results, we can perform the balancing ourselves by using the standardizing method shown in [Standardizing margins](#). To do that, we will input a balanced dataset and then type `margins, noesample`.

```

. use http://www.stata-press.com/data/r11/acmemanuf, clear
. regress y pressure##temp
 (output omitted)
. drop _all
. input pressure temp
 pressure temp
1. 1 1
2. 1 2
3. 1 3
4. 2 1
5. 2 2
6. 2 3
7. end
. margins, noesample
Predictive margins Number of obs = 6
Model VCE : OLS
Expression : Linear prediction, predict()

```

|       | Delta-method |           |       |       |                      |         |
|-------|--------------|-----------|-------|-------|----------------------|---------|
|       | Margin       | Std. Err. | z     | P> z  | [95% Conf. Interval] |         |
| _cons | 115.3758     | 1.530199  | 75.40 | 0.000 | 112.3767             | 118.375 |

We obtain the same results as previously.

## Balancing nonlinear responses

If our testing had produced a binary outcome, say, acceptable/unacceptable, rather than a continuous variable, we would type

```

. use http://www.stata-press.com/data/r11/acmemanuf, clear
. logistic acceptable pressure##temp
. margins, asbalanced

```

The result of doing that would be 0.680. If we omitted the `asbalanced` option, the result would have been 0.667. The two results are so similar because `acmemanuf.dta` is nearly balanced.

Even though the `asbalanced` option can be used on both linear and nonlinear responses, such as probabilities, there is an issue of which you should be aware. The most widely used formulas for balancing responses apply the balancing to the linear prediction, average that as if it were balanced, and then apply the nonlinear transform. That is the calculation that produced 0.680.

An alternative would be to apply the standardization method. That amounts to making the linear predictions observation by observation, applying the nonlinear transform to each, and then averaging the nonlinear result as if it were balanced. You could do that by typing

```

. use http://www.stata-press.com/data/r11/acmemanuf, clear
. logistic acceptable pressure##temp
. clear
. input pressure temp
 (see above for entered data)
. margins, noesample

```

The result from the standardization procedure would be 0.672. These two ways of averaging nonlinear responses are discussed in detail in [Lane and Nelder \(1982\)](#) within the context of general linear models.

Concerning the method used by the `asbalanced` option, if your data start balanced and you have a nonlinear response, you will get different results with and without the `asbalanced` option!

## Treating a subset of covariates as balanced

So far, we have treated all the covariates as if they were balanced. `margins` will allow you to treat a subset of the covariates as balanced, too. For instance, you might be performing an experiment in which you are randomly allocating patients to a treatment arm and so want to balance on arm, but you do not want to balance the other characteristics because you want mean effects for the experiment's population.

In this example, we will imagine that the outcome of the experiment is continuous. We type

```
. use http://www.stata-press.com/data/r11/margex, clear
. regress y arm##sex sex##agegroup
. margins, at((asbalanced) arm)
```

If we wanted results balanced on `agegroup` as well, we could type

```
. margins, at((asbalanced) arm agegroup)
```

If we wanted results balanced on all three covariates, we could type

```
. margins, at((asbalanced) arm agegroup sex)
```

or we could type

```
. margins, at((asbalanced) _factor)
```

or we could type

```
. margins, asbalanced
```

## Using `fvset` design

As a convenience feature, equivalent to

```
. regress y arm##sex sex##agegroup
. margins, at((asbalanced) arm sex)
```

is

```
. fvset design asbalanced arm sex
. regress y arm##sex sex##agegroup
. margins
```

The advantage of the latter is that you have to set the variables as balanced only once. This is useful when balancing is a design characteristic of certain variables and you wish to avoid accidentally treating them as unbalanced.

If you save your data after `fvsetting`, the settings will be remembered in future sessions. If you want to clear the setting(s), type

```
. fvset clear varlist
```

See [\[R\] fvset](#).



## Balancing in the presence of empty cells

The issue of empty cells is not exclusively an issue of balancing, but there are special considerations when balancing. Empty cells are discussed generally in [Estimability of margins](#).

An empty cell is an interaction of levels of two or more factor variables for which you have no data. Usually, margins involving empty cells cannot be estimated. When balancing, there is an alternate definition of the margin that allows the margin to be estimated. `margins` makes the alternate calculation when you specify the `emptycells(reweight)` option. By default, `margins` uses the `emptycells(strict)` option.

If you have empty cells in your data and you request margins involving the empty cells, those margins will be marked as not estimable even if you specify the `asbalanced` option.

```
. use http://www.stata-press.com/data/r11/estimability, clear
(margins estimability)
. regress y sex##group
(output omitted)
. margins sex, asbalanced
```

```
Adjusted predictions Number of obs = 69
Model VCE : OLS
Expression : Linear prediction, predict()
at : sex (asbalanced)
 group (asbalanced)
```

|     | Delta-method    |           | z     | P> z  | [95% Conf. Interval] |          |
|-----|-----------------|-----------|-------|-------|----------------------|----------|
|     | Margin          | Std. Err. |       |       |                      |          |
| sex |                 |           |       |       |                      |          |
| 0   | 21.91389        | 1.119295  | 19.58 | 0.000 | 19.72011             | 24.10767 |
| 1   | (not estimable) |           |       |       |                      |          |

This example is discussed in [Estimability of margins](#), although without the `asbalanced` option. What is said there is equally relevant to the `asbalanced` case. For reasons explained there, the margin for `sex = 1` cannot be estimated.

The margin for `sex = 1` can be estimated in the `asbalanced` case if you are willing to make an assumption. Remember that `margins` makes the balanced calculation by summing the responses associated with the levels and then dividing by the number of levels. If you specify `emptycells(reweight)`, `margins` sums what is available and divides by the number available. Thus you are assuming that, whatever the responses in the empty cells, those responses are such that they would not change the overall mean of what is observed.

The results of specifying `emptycells(reweight)` are

```
. margins sex, asbalanced emptycells(reweight)
Adjusted predictions Number of obs = 69
Model VCE : OLS
Expression : Linear prediction, predict()
at : sex (asbalanced)
 group (asbalanced)
```

|     | Delta-method |           | z     | P> z  | [95% Conf. Interval] |          |
|-----|--------------|-----------|-------|-------|----------------------|----------|
|     | Margin       | Std. Err. |       |       |                      |          |
| sex |              |           |       |       |                      |          |
| 0   | 21.91389     | 1.119295  | 19.58 | 0.000 | 19.72011             | 24.10767 |
| 1   | 24.85185     | 1.232304  | 20.17 | 0.000 | 22.43658             | 27.26712 |

## Obtaining margins with nested designs

### Introduction

Factors whose meaning depends on other factors are called nested factors, and the factors on which their meaning depends are called the nesting factors. For instance, assume that we have a sample of patients and each patient is assigned to one doctor. Then patient is nested within doctor. Let the identifiers of the first 5 observations of our data be

| doctor | patient | name  |
|--------|---------|-------|
| 1      | 1       | Fred  |
| 1      | 2       | Mary  |
| 1      | 3       | Bob   |
| 2      | 1       | Karen |
| 2      | 2       | Hank  |

The first patient on one doctor's list has nothing whatsoever to do with the first patient on another doctor's list. The meaning of `patient = 1` is defined only when the value of `doctor` is supplied.

Nested factors enter into models as interactions of nesting and nested; the nested factor does not appear by itself. We might estimate a model such as

```
. regress y ... i.doctor doctor#patient ...
```

You do not include `i.patient` because the coding for patient has no meaning except within doctor. Patient 1 is Fred for doctor 1 and Karen for doctor 2, etc.

`margins` provides an option to help account for the structure of nested models. The `within(varlist)` option specifies that `margins` estimate and report a set of margins for the value combinations of `varlist`. We might type

```
. margins, within(doctor)
```

Margin calculations are performed first for `doctor = 1`, then for `doctor = 2`, and so on.

Sometimes you need to specify `within()`, and other times you do not. Let's consider the particular model

```
. regress y i.doctor doctor#patient i.sex sex#doctor#patient
```

The guidelines are the following:

1. You may compute overall margins by typing `margins`.
2. You may compute overall margins within levels of a nesting factor by typing `margins, within(doctor)`.
3. You may compute margins of a nested factor within levels of its nesting factor by typing `margins patient, within(doctor)`.
4. You may compute margins of factors in your model, as long as the factor does not nest other factors and is not nested within other factors, by typing `margins sex`.
5. You may not compute margins of a nesting factor, such as `margins doctor`, because they are not estimable.

For examples using `within()`, see [\[R\] anova](#).

## Margins with nested designs as though the data were balanced

To obtain margins with nested designs as though the data were balanced, the guidelines are the same as above except that 1) you add the `asbalanced` option and 2) whenever you do not specify `within()`, you specify `emptycells(reweight)`. The updated guidelines are

1. You may compute overall margins by typing `margins, asbalanced emptycells(reweight)`.
2. You may compute overall margins within levels of a nesting factor by typing `margins, asbalanced within(doctor)`.
3. You may compute margins of a nested factor within levels of its nesting factor by typing `margins patient, asbalanced within(doctor)`.
4. You may compute margins of factors in your model, as long as the factor does not nest other factors and is not nested within other factors, by typing `margins sex, asbalanced emptycells(reweight)`.
5. You may not compute margins of a nesting factor, such as `margins doctor`, because they are not estimable.

Just as explained in [Using `fvset design`](#), rather than specifying the `asbalanced` option, you may set the balancing characteristic on the factor variables once and for all by using the command `fvset design asbalanced varlist`.

### □ Technical note

Specifying either `emptycells(reweight)` or `within(varlist)` causes `margins` to rebalance over all empty cells in your model. If you have interactions in your model that are not involved in the nesting, `margins` will lose its ability to detect estimability.

□

### □ Technical note

Careful readers will note that the description of `within(varlist)` matches closely the description of `over(varlist)`. The concept of nesting is similar to the concept of subpopulations. `within()` differs from `over()` in that it gracefully handles the missing cells when margins are computed as balanced.

□

## Coding of nested designs

In the [Introduction](#) to this section, we showed a coding of the nested variable `patient`, where the coding started over with each `doctor`:

| doctor | patient | name  |
|--------|---------|-------|
| 1      | 1       | Fred  |
| 1      | 2       | Mary  |
| 1      | 3       | Bob   |
| 2      | 1       | Karen |
| 2      | 2       | Hank  |

That coding style is not required. The data could just as well have been coded

| doctor | patient | name  |
|--------|---------|-------|
| 1      | 1       | Fred  |
| 1      | 2       | Mary  |
| 1      | 3       | Bob   |
| 2      | 4       | Karen |
| 2      | 5       | Hank  |

or even

| doctor | patient | name  |
|--------|---------|-------|
| 1      | 1037239 | Fred  |
| 1      | 2223942 | Mary  |
| 1      | 0611393 | Bob   |
| 2      | 4433329 | Karen |
| 2      | 6110271 | Hank  |

Actually, either of the above two alternatives are better than the first one because `margins` will be better able to give you feedback about estimability should you make a mistake following the guidelines. On the other hand, both of these two alternatives require more memory at the estimation step. If you run short of memory, you will need to recode your patient ID to the first coding style, which you could do by typing

```
. sort doctor patient
. by doctor: gen newpatient = _n
```

Alternatively, you can set `emptycells drop` and continue to use your patient ID variable just as it is coded. If you do this, we recommend that you remember to type `set emptycells keep` when you are finished; `margins` is better able to determine estimability that way. If you regularly work with large nested models, you can set `emptycells keep`, permanently so that the setting persists across sessions. See [\[R\] set emptycells](#).

## Special topics

### Requirements for model specification

The results that `margins` reports are based on the most recently fit model or, in Stata jargon, the most recently issued estimation command. Here we discuss 1) mechanical requirements for how you specify that estimation command, 2) work-arounds to use when those restrictions prove impossible, and 3) requirements for `margins`' `predict(pred_opt)` option to work.

Concerning 1, when you specify the estimation command, covariates that are logically factor variables must be Stata factor variables, and that includes indicator variables, binary variables, and dummies. It will not do to type

```
. regress y ... female ...
```

even if `female` is a 0/1 variable. You must type

```
. regress y ... i.female ...
```

If you violate this rule, you will not get incorrect results, but you will discover that you will be unable to obtain margins on `female`:

```
. margins female
factor female not found in e(b)
r(111);
```

It is also important that if the same continuous variable appears in your model more than once, differently transformed, those transforms be performed via Stata's factor-variable notation. It will not do to type

```
. generate age2 = age^2
. regress y ... age age2 ...
```

You must type

```
. regress y ... age c.age#c.age ...
```

You must do that because `margins` needs to know everywhere that variable appears in the model if it is to be able to set covariates to fixed values.

Concerning 2, sometimes the transformations you desire may not be achievable using the factor-variable notation; in those situations, there is a work-around. Let's assume you wish to estimate

```
. generate age1_5 = age^1.5
. regress y ... age age1_5 ...
```

There is no factor-variable notation for including `age` and `age1.5` in a model, so obviously you are going to obtain the estimates by typing just what we have shown. In what follows, it would be okay if there are interactions of `age` and `age1_5` with other variables specified by the factor-variable notation, so the model could just as well be

```
. regress y ... age age1_5 sex#c.age sex#c.age1_5 ...
```

Let's assume you have fit one of these two models. On any subsequent `margins` command where you leave `age` free to vary, there will be no issue. You can type

```
. margins female
```

and results will be correct. Issues arise when you attempt to fix `age` at predetermined values. The following would produce incorrect results:

```
. margins female, at(age=20)
```

The results would be incorrect because they leave `age1_5` free to vary, and, logically, fixing `age` implies that `age1_5` should also be fixed. Because we were unable to state the relationship between `age` and `age1_5` using the factor-variable notation, `margins` does not know to fix `age1_5` at `201.5` when it fixes `age` at 20. To get the correct results, you must fix the value of `age1_5` yourself:

```
. margins female, at(age=20 age1_5=89.442719)
```

That command produces correct results. In the command, 89.442719 is `201.5`.

In summary, when there is a functional relationship between covariates of your model and that functional relationship is not communicated to `margins` via the factor-variable notation, then it becomes your responsibility to ensure that all variables that are functionally related are set to the appropriate fixed values when any one of them is set to a fixed value.

Concerning 3, we wish to amend our claim that you can calculate margins for anything that `predict` will produce. We need to add a qualifier. Let us show you an example where the statement is not true. After `regress`, `predict` will predict something it calls  $\text{pr}(a,b)$ , which is the probability  $a \leq y \leq b$ . Yet if we attempted to use `pr()` with `margins` after estimation by `regress`, we would obtain

```
. margins sex, predict(pr(10,20))
prediction is a function of possibly stochastic quantities other than e(b)
r(498);
```

What we should have stated was that you can calculate margins for anything that `predict` will produce for which all the estimated quantities used in its calculation appear in `e(V)`, the estimated VCE. `pr()` is a function of  $\beta$ , the estimated coefficients, and of  $s^2$ , the estimated variance of the residual. `regress` does not post the variance of the residual variance (sic) in `e(V)`, or even estimate it, and therefore, `predict(pr(10,20))` cannot be specified with `margins` after estimation by `regress`.

It is unlikely that you will ever encounter these kinds of problems because there are so few predictions where the components are not posted to `e(V)`. If you do encounter the problem, the solution may be to specify `nose` to suppress the standard-error calculation. If the problem is not with computing the margin, but with computing its standard error, `margins` will report the result:

```
. margins sex, predict(pr(10,20)) nose
(output appears with SEs, tests, and CIs left blank)
```

❑ Technical note

Programmers: If you run into this after running an estimation command that you have written, be aware that as of Stata 11, you are supposed to set in `e(marginsok)` the list of options allowed with `predict` that are okay to use with `margins`. When that list is not set, `margins` looks for violations of its assumptions and, if it finds any, refuses to proceed.



Estimability of margins

Sometimes `margins` will report that a margin cannot be estimated:

```
. use http://www.stata-press.com/data/r11/estimability, clear
(margins estimability)

. regress y sex##group
(output omitted)

. margins sex

Predictive margins Number of obs = 69
Model VCE : OLS
Expression : Linear prediction, predict()
```

|     | Delta-method    |           | z     | P> z  | [95% Conf. Interval] |          |
|-----|-----------------|-----------|-------|-------|----------------------|----------|
|     | Margin          | Std. Err. |       |       |                      |          |
| sex |                 |           |       |       |                      |          |
| 0   | 21              | .8500245  | 24.71 | 0.000 | 19.33398             | 22.66602 |
| 1   | (not estimable) |           |       |       |                      |          |

In the above output, the margin for `sex = 0` is estimated, but the margin for `sex = 1` is not estimable. This occurs because of empty cells. An empty cell is an interaction of levels of two or more factor variables for which you have no data. In the example, the lack of estimability arises because we have two empty cells:

```
. table sex group
```

| sex | group |   |    |   |   |
|-----|-------|---|----|---|---|
|     | 1     | 2 | 3  | 4 | 5 |
| 0   | 2     | 9 | 27 | 8 | 2 |
| 1   | 9     | 9 | 3  |   |   |

To calculate the marginal mean response for `sex = 1`, we have no responses to average over for `group = 4` and `group = 5`. We obviously could calculate that mean for the observations that really are `sex = 1`, but remember, the marginal calculation for `sex = 1` treats everyone as if female, and we will thus have 8 and 2 observations for which we have no basis for estimating the response.

There is no solution for this problem unless you are willing to treat the data as if it were balanced and adjust your definition of a margin; see [Balancing in the presence of empty cells](#).

## Manipulability of tests

Manipulability is a problem that arises with some tests, and in particular, arises with Wald tests. Tests of margins are based on Wald tests, hence our interest. This is a generic issue and not specific to the `margins` command.

Let's understand the problem. Consider performing a test of whether some statistic  $\phi$  is 0. Whatever the outcome of that test, it would be desirable if the outcome were the same were we to test whether the  $\sqrt{\phi}$  were 0, or whether  $\phi^2$  were 0, or whether any other monotonic transform of  $\phi$  were 0 (for  $\phi^2$ , we were considering only the positive half of the number line). If a test does not have that property, it is manipulable.

Wald tests are manipulable, and that means the tests produced by `margins` are manipulable. You can see this for yourself by typing

```
. use http://www.stata-press.com/data/r11/margex, clear
. replace y = y - 65
. regress y sex##group
. margins
. margins, expression(predict(xb)^2)
```

We would prefer if the test against zero produced by `margins` was equal to the test produced by `margins, expression(predict(xb)^2)`. But alas, they produce different results. The first produces  $z = 12.93$ , and the second produces  $z = 12.57$ .

The difference is not much in our example, but behind the scenes, we worked to make it small. We subtracted 65 from `y` so that the experiment would be for a case where it might be reasonable that you would be testing against 0. One does not typically test whether the mean income in the United States is zero or whether the mean blood pressure of live patients is zero. Had we left `y` as it was originally, we would have obtained  $z = 377$  and  $z = 128$ . We did not want to show that comparison to you first because the mean of `y` is so far from 0 that you probably would never be testing it. The corresponding difference in  $\phi$  is tiny.

Regardless of the example, it is important that you base your tests in the metric where the likelihood surface is most quadratic. For further discussion on manipulability, see [Manipulability in \[R\] predictnl](#).

This manipulability is not limited to Wald tests after estimation; you can also see the manipulability of results produced by linear regression just by applying nonlinear transforms to a covariate ([Phillips and Park 1988](#); [Gould 1996](#)).

## Using margins after the estimates use command

Assume you fit and used `estimates save` (see [\[R\] estimates save](#)) to save the estimation results:

```
. regress y sex##group age c.age*c.age if site==1
. ...
. estimates save mymodel
(file mymodel.ster saved)
```

Later, perhaps in a different Stata session, you reload the estimation results by typing

```
. estimates use mymodel
```

You plan to use `margins` with the reloaded results. You must remember that `margins` bases its results not only on the current estimation results but also on the current data in memory. Before you can use `margins`, you must reload the dataset on which you fit the model or, if you wish to produce standardized margins, some other dataset.

```
. use mydata, clear
(data for fitting models)
```

If the dataset you loaded contained the data for standardization, you can stop reading; you know that to produce standardized margins, you need to specify the `noesample` option.

We reloaded the original data and want to produce margins for the estimation sample. In addition to the data, `margins` requires that `e(sample)` be set, as `margins` will remind us:

```
. margins sex
e(sample) does not identify the estimation sample
r(322);
```

The best solution is to use `estimates esample` to rebuild `e(sample)`:

```
. estimates esample: y sex group age if site==1
```

If we knew we had no missing values in `y` and the covariates, we could type

```
. estimates esample: if site==1
```

Either way, `margins` would now work:

```
. margins sex
(usual output appears)
```

There is an alternative. We do not recommend it, but we admit that we have used it. Rather than rebuilding `e(sample)`, you can use `margins`' `noesample` option to tell `margins` to skip using `e(sample)`. You could then specify the appropriate `if` statement (if necessary) to identify the estimation sample:

```
. estimates use mymodel
. use mydata, clear
(data for fitting models)
. margins sex if !missing(y, sex, group age) & site==1, noesample
(usual output appears)
```



In the above, we are not really running on a sample different from the estimation sample; we are merely using `noesample` to fool `margins`, and then we are specifying on the `margins` command the conditions equivalent to re-create `e(sample)`.

If we wish to obtain `vce(unconditional)` results, however, `noesample` will be insufficient. We must also specify the `force` option,

```
. margins sex if !missing(y, sex, group age) & site==1,
> vce(unconditional) noesample force
(usual output appears)
```

Regardless of the approach you choose—resetting `e(sample)` or specifying `noesample` and possibly `force`—make sure you are right. In the `vce(delta)` case, you want to be right to ensure that you obtain the results you want. In the `vce(unconditional)` case, you need to be right because otherwise results will be statistically invalid.

## Syntax of `at()`

In `at(atspec)`, *atspec* may contain one or more of the following specifications:

*varlist*

(*stat*) *varlist*

*varname* = #

*varname* = (*numlist*)

where

1. *varnames* must be covariates in the previously fit model (estimation command).
2. Variable names (whether in *varname* or *varlist*) may be continuous variables, factor variables, or specific level variables, such as `age`, `group`, or `3.group`.
3. *varlist* may also be one of three standard lists:
  - a. `_all` (all covariates),
  - b. `_factor` (all factor-variable covariates), or
  - c. `_continuous` (all continuous covariates).
4. Specifications are processed from left to right with latter specifications overriding previous ones.
5. *stat* can be any of the following:

(Continued on next page)

| <i>stat</i>        | description                                | variables<br>allowed |
|--------------------|--------------------------------------------|----------------------|
| <u>as</u> observed | at observed values in the sample (default) | all                  |
| mean               | means (default for <i>varlist</i> )        | all                  |
| median             | medians                                    | continuous           |
| p1                 | 1st percentile                             | continuous           |
| p2                 | 2nd percentile                             | continuous           |
| ...                | 3rd–49th percentiles                       | continuous           |
| p50                | 50th percentile (same as <i>median</i> )   | continuous           |
| ...                | 51st–97th percentiles                      | continuous           |
| p98                | 98th percentile                            | continuous           |
| p99                | 99th percentile                            | continuous           |
| min                | minimums                                   | continuous           |
| max                | maximums                                   | continuous           |
| zero               | fixed at zero                              | continuous           |
| base               | base level                                 | factors              |
| <u>as</u> balanced | all levels equally probable and sum to 1   | factors              |

Any *stat* except *zero*, *base*, and *asbalanced* may be prefixed with an *o* to get the overall statistic—the sample over all *over()* groups. For example, *omean*, *omedian*, and *op25*. Overall statistics differ from their correspondingly named statistics only when the *over()* or *within()* option is specified. When no *stat* is specified, *mean* is assumed.

Estimation commands that may be used with margins

- margins* may be used after most estimation commands.
- margins* cannot be used after estimation commands that do not produce full variance matrices, such as *exlogistic* and *expoisson* (see [\[R\] exlogistic](#) and [\[R\] expoisson](#)).
- margins* is all about covariates and cannot be used after estimation commands that do not post the covariates, which eliminates *gmm* (see [\[R\] gmm](#)).
- margins* cannot be used after estimation commands that have an odd data organization, and that excludes *asclogit*, *asmprobit*, *asroprobit*, and *nlogit* (see [\[R\] asclogit](#), [\[R\] asmprobit](#), [\[R\] asroprobit](#), and [\[R\] nlogit](#)).

Glossary

- adjusted mean.** A *margin* when the response is the linear predictor from linear regression, ANOVA, etc. For some authors, adjusting also implies adjusting for unbalanced data. See [Obtaining margins of responses](#) and see [Obtaining margins as though the data were balanced](#).
- average marginal effect.** See [marginal effect and average marginal effect](#).
- average partial effect.** See [partial effect and average partial effect](#).
- conditional margin.** A *margin* when the response is evaluated at fixed values of all the covariates. If any covariates are left to vary, the margin is called a predictive margin.

**effect.** The effect of  $x$  is the derivative of the *response* with respect to covariate  $x$ , or it is the difference in responses caused by a discrete change in  $x$ . Also see [marginal effect](#).

The effect of  $x$  measures the change in the response for a change in  $x$ . Derivatives or differences might be reported as elasticities. If  $x$  is continuous, the effect is measured continuously. If  $x$  is a factor, the effect is measured with respect to each level of the factor and may be calculated as a discrete difference or as a continuous change, as measured by the derivative. `margins` calculates the discrete difference by default and calculates the derivative if the `continuous` option is specified.

**elasticity and semielasticity.** The elasticity of  $y$  with respect to  $x$  is  $d(\ln y)/d(\ln x) = (x/y) \times (dy/dx)$ , which is approximately equal to the proportional change in  $y$  for a proportional change in  $x$ .

The semielasticity of  $y$  with respect to  $x$  is either 1)  $dy/d(\ln x) = x \times (dy/dx)$  or 2)  $d(\ln y)/dx = (1/y) \times (dy/dx)$ , which is approximately 1) the change in  $y$  for a proportional change in  $x$  or 2) the proportional change in  $y$  for a change in  $x$ .

**empty cell.** An interaction of levels of two or more factor variables for which you have no data. For instance, you have sex interacted with group in your model, and in your data there are no females in group 1. Empty cells affect which margins can be estimated; see [Estimability of margins](#).

**estimability.** Estimability concerns whether a margin can be uniquely estimated (identified); see [Estimability of margins](#).

**estimated marginal mean.** This is one of the few terms that has the same meaning across authors. An estimated marginal mean is a margin assuming the levels of each factor covariate are equally likely (balanced), including interaction terms. This is obtained using `margins'` `asbalanced` option. In addition, there is an alternate definition of estimated marginal mean in which margins involving empty cells are redefined so that they become estimable. This is invoked by `margins'` `emptycells(reweight)` option. See [Balancing in the presence of empty cells](#).

**least-squares mean.** Synonym for *estimated marginal mean*.

**margin.** A statistic calculated from predictions or other statistics of a previously fit model at fixed values of some covariates and averaging or otherwise integrating over the remaining covariates. The prediction or other statistic on which the margin is based is called the response.

If all the covariates are fixed, then the margin is called a conditional margin. If any covariates are left to vary, the margin is called a predictive margin.

In this documentation, we divide margins on the basis of whether the statistic is a response or a derivative of a response; see [Obtaining margins of responses](#) and [Obtaining margins of derivatives of responses](#).

**marginal effect and average marginal effect.** The marginal effect of  $x$  is the *margin* of the *effect* of  $x$ . The term is popular with social scientists, and because of that, you might think the word marginal in marginal effect means derivative because of terms like marginal cost and marginal revenue. Marginal used in that way, however, refers to the derivative of revenue and the derivative of cost; it refers to the numerator, whereas marginal effect refers to the denominator. Moreover, *effect* is already a derivative or difference.

Some researchers interpret marginal in marginal effect to mean instantaneous, and thus a marginal effect is the instantaneous derivative rather than the discrete first-difference, corresponding to `margins'` `continuous` option. Researchers who use marginal in this way refer to the discrete difference calculation of an effect as a partial effect.

Other researchers define marginal effect to be the margin when all covariates are held fixed and the average marginal effect when some covariates are not fixed.

**out-of-sample prediction.** Predictions made in one dataset using the results from a model fit on another. Sample here refers to the sample on which the model was fit, and out-of-sample refers to the dataset on which the predictions are made.

**partial effect** and **average partial effect.** Some authors restrict the term *marginal effect* to mean derivatives and use the term partial effect to denote discrete differences; see [marginal effect and average marginal effect](#).

**population marginal mean.** The theoretical (true) value that is estimated by *estimated marginal mean*. We avoid this term because it can be confused with the concept of a population in survey statistics, with which the population marginal mean has no connection.

**posting results, posting margins.** A Stata concept having to do with saving the results from the `margins` command in `e()` so that those results can be used as if they were estimation results, thus allowing the subsequent use of postestimation commands, such as `test`, `testnl`, `lincom`, and `nlcom` (see [\[R\] test](#), [\[R\] testnl](#), [\[R\] lincom](#), and [\[R\] nlcom](#)). This is achieved by specifying margins' `post` option. See [Example 10: Testing margins](#).

**predictive margin.** A *margin* in which all the covariates are not fixed. When all covariates are fixed, it is called a *conditional margin*.

**recycled prediction.** A synonym for *predictive margin*.

**response.** A prediction or other statistic derived from combining the parameter estimates of a fitted model with data or specified values on covariates. Derivatives of responses are themselves responses. Responses are what we take *margins* of.

**standardized margin.** The margin calculated on data different from the data used to fit the model. The term standardized is usually reserved for situations in which the alternate population is a reference population, which may be a real or artificial, and which is treated as fixed.

**subpopulation.** A subset of your sample that represents a subset of the population, such as the males in a sample of people. In survey contexts when it is desired to account for sampling of the covariates, standard errors for marginal statistics and effects need to account for both the population and the subpopulation. This is accomplished by specifying the `vce(unconditional)` option and one of the `subpop()` or `over()` options. In fact, the above is allowed even when your data are not `svyset` because `vce(unconditional)` implies that the sample represents a population.

## Saved results

`margins` saves the following in `r()`:

### Scalars

|                              |                                               |
|------------------------------|-----------------------------------------------|
| <code>r(N)</code>            | number of observations                        |
| <code>r(N_sub)</code>        | subpopulation observations                    |
| <code>r(N_clust)</code>      | number of clusters                            |
| <code>r(N_psu)</code>        | number of samples PSUs, survey data only      |
| <code>r(N_strata)</code>     | number of strata, survey data only            |
| <code>r(df_r)</code>         | variance degrees of freedom, survey data only |
| <code>r(N_poststrata)</code> | number of post strata, survey data only       |
| <code>r(k_margins)</code>    | number of terms in <i>marginlist</i>          |
| <code>r(k_by)</code>         | number of subpopulations                      |

### Macros

|                               |                                                                                                              |
|-------------------------------|--------------------------------------------------------------------------------------------------------------|
| <code>r(title)</code>         | title in estimation output                                                                                   |
| <code>r(subpop)</code>        | <i>subspec</i> from <code>subpop()</code>                                                                    |
| <code>r(model_vce)</code>     | <i>vcetype</i> from estimation command                                                                       |
| <code>r(model_vcetype)</code> | Std. Err. title from estimation command                                                                      |
| <code>r(vce)</code>           | <i>vcetype</i> specified in <code>vce()</code>                                                               |
| <code>r(vcetype)</code>       | title used to label Std. Err.                                                                                |
| <code>r(clustvar)</code>      | name of cluster variable                                                                                     |
| <code>r(margins)</code>       | <i>marginlist</i>                                                                                            |
| <code>r(predict_label)</code> | label from <code>predict()</code>                                                                            |
| <code>r(expression)</code>    | response expression                                                                                          |
| <code>r(xvars)</code>         | <i>varlist</i> from <code>dydx()</code> , <code>dyex()</code> , <code>eydx()</code> , or <code>eyex()</code> |
| <code>r(derivatives)</code>   | “, “dy/dx”, “dy/ex”, “ey/dx”, “ey/ex”                                                                        |
| <code>r(over)</code>          | <i>varlist</i> from <code>over()</code>                                                                      |
| <code>r(within)</code>        | <i>varlist</i> from <code>within()</code>                                                                    |
| <code>r(by)</code>            | union of <code>r(over)</code> and <code>r(within)</code> lists                                               |
| <code>r(by#)</code>           | interaction notation identifying the #th subpopulation                                                       |
| <code>r(atstats#)</code>      | the #th <code>at()</code> specification                                                                      |
| <code>r(emptycells)</code>    | <i>empspec</i> from <code>emptycells()</code>                                                                |

### Matrices

|                           |                                                     |
|---------------------------|-----------------------------------------------------|
| <code>r(b)</code>         | estimates                                           |
| <code>r(V)</code>         | variance–covariance matrix of the estimates         |
| <code>r(Jacobian)</code>  | Jacobian matrix                                     |
| <code>r(_N)</code>        | sample size corresponding to each margin estimate   |
| <code>r(at)</code>        | matrix of values from the <code>at()</code> options |
| <code>r(chainrule)</code> | chainrule information from the fitted model         |

`margins` with the `post` option also saves the following in `e()`:

#### Scalars

|                              |                                               |
|------------------------------|-----------------------------------------------|
| <code>e(N)</code>            | number of observations                        |
| <code>e(N_sub)</code>        | subpopulation observations                    |
| <code>e(N_clust)</code>      | number of clusters                            |
| <code>e(N_psu)</code>        | number of samples PSUs, survey data only      |
| <code>e(N_strata)</code>     | number of strata, survey data only            |
| <code>e(df_r)</code>         | variance degrees of freedom, survey data only |
| <code>e(N_poststrata)</code> | number of post strata, survey data only       |
| <code>e(k_margins)</code>    | number of terms in <i>marginlist</i>          |
| <code>e(k_by)</code>         | number of subpopulations                      |

#### Macros

|                               |                                                                                                              |
|-------------------------------|--------------------------------------------------------------------------------------------------------------|
| <code>e(cmd)</code>           | <code>margins</code>                                                                                         |
| <code>e(title)</code>         | title in estimation output                                                                                   |
| <code>e(subpop)</code>        | <i>subspec</i> from <code>subpop()</code>                                                                    |
| <code>e(model_vce)</code>     | <i>vcetype</i> from estimation command                                                                       |
| <code>e(model_vcetype)</code> | Std. Err. title from estimation command                                                                      |
| <code>e(vce)</code>           | <i>vcetype</i> specified in <code>vce()</code>                                                               |
| <code>e(vcetype)</code>       | title used to label Std. Err.                                                                                |
| <code>e(clustvar)</code>      | name of cluster variable                                                                                     |
| <code>e(margins)</code>       | <i>marginlist</i>                                                                                            |
| <code>e(predict_label)</code> | label from <code>predict()</code>                                                                            |
| <code>e(expression)</code>    | prediction expression                                                                                        |
| <code>e(xvars)</code>         | <i>varlist</i> from <code>dydx()</code> , <code>dyex()</code> , <code>eydx()</code> , or <code>eyex()</code> |
| <code>e(derivatives)</code>   | “, “dy/dx”, “dy/ex”, “ey/dx”, “ey/ex”                                                                        |
| <code>e(over)</code>          | <i>varlist</i> from <code>over()</code>                                                                      |
| <code>e(within)</code>        | <i>varlist</i> from <code>within()</code>                                                                    |
| <code>e(by)</code>            | union of <code>r(over)</code> and <code>r(within)</code> lists                                               |
| <code>e(by#)</code>           | interaction notation identifying the #th subpopulation                                                       |
| <code>e(atstats#)</code>      | the #th <code>at()</code> specification                                                                      |
| <code>e(emptycells)</code>    | <i>empspec</i> from <code>emptycells()</code>                                                                |
| <code>e(predict)</code>       | <code>_no_predict</code>                                                                                     |

#### Matrices

|                           |                                                     |
|---------------------------|-----------------------------------------------------|
| <code>e(b)</code>         | estimates                                           |
| <code>e(V)</code>         | variance–covariance matrix of the estimates         |
| <code>e(Jacobian)</code>  | Jacobian matrix                                     |
| <code>e(_N)</code>        | sample size corresponding to each margin estimate   |
| <code>e(at)</code>        | matrix of values from the <code>at()</code> options |
| <code>e(chainrule)</code> | chainrule information from the fitted model         |

## Methods and formulas

`margins` is implemented as an ado-file.

Margins are statistics calculated from predictions of a previously fit model at fixed values of some covariates and averaging or otherwise integrating over the remaining covariates. There are many names for the different statistics that `margins` can compute: estimates marginal means (see [Searle, Speed, and Milliken \[1980\]](#)), predictive margins (see [Graubard and Korn \[2004\]](#)), marginal effects (see [Greene \[2008\]](#)), and average marginal/partial effects (see ([Wooldridge 2002](#)) and [Bartus \[2005\]](#)).

Methods and formulas are presented under the following headings:

- Notation
- Marginal effects
- Fixing covariates and balancing factors
- Estimable functions
- Standard errors conditional on the covariates
- Unconditional standard errors

Notation

Let  $\theta$  be the vector of parameters in the current model fit, let  $\mathbf{z}$  be a vector of covariate values, and let  $f(\mathbf{z}, \theta)$  be a scalar-valued function returning the value of the predictions of interest. The following table illustrates the parameters and default prediction for several of Stata’s estimation commands.

| command  | $\theta$                           | $\mathbf{z}$                 | $f(\mathbf{z}, \theta)$                                  |
|----------|------------------------------------|------------------------------|----------------------------------------------------------|
| regress  | $\beta$                            | $\mathbf{x}$                 | $\mathbf{x}\beta$                                        |
| cloglog  | $\beta$                            | $\mathbf{x}$                 | $1 - e^{-e^{x\beta}}$                                    |
| logit    | $\beta$                            | $\mathbf{x}$                 | $1/(1 + e^{-x\beta})$                                    |
| poisson  | $\beta$                            | $\mathbf{x}$                 | $e^{x\beta}$                                             |
| probit   | $\beta$                            | $\mathbf{x}$                 | $\Phi(\mathbf{x}\beta)$                                  |
| biprobit | $\beta_1, \beta_2, \rho$           | $\mathbf{x}_1, \mathbf{x}_2$ | $\Phi_2(\mathbf{x}_1\beta_1, \mathbf{x}_2\beta_2, \rho)$ |
| mlogit   | $\beta_1, \beta_2, \dots, \beta_k$ | $\mathbf{x}$                 | $e^{-x\beta_1}/(\sum_i e^{-x\beta_i})$                   |
| nbgreg   | $\beta, \ln\alpha$                 | $\mathbf{x}$                 | $e^{x\beta}$                                             |

$\Phi()$  and  $\Phi_2()$  are cumulative distribution functions:  $\Phi()$  for the standard normal distribution and  $\Phi_2()$  for the standard bivariate normal distribution.

`margins` computes estimates of

$$p(\theta) = \frac{1}{M_{S_p}} \sum_{j=1}^M \delta_j(S_p) f(\mathbf{z}_j, \theta)$$

where  $\delta_j(S_p)$  identifies elements within the subpopulation  $S_p$  (for the prediction of interest),

$$\delta_j(S_p) = \begin{cases} 1, & j \in S_p \\ 0, & j \notin S_p \end{cases}$$

$M_{S_p}$  is the subpopulation size,

$$M_{S_p} = \sum_{j=1}^M \delta_j(S_p)$$

and  $M$  is the population size.

Let  $\hat{\theta}$  be the vector of parameter estimates. Then `margins` estimates  $p(\theta)$  via

$$\hat{p} = \frac{1}{w} \sum_{j=1}^N \delta_j(S_p) w_j f(\mathbf{z}_j, \hat{\theta})$$

where

$$w_{\cdot} = \sum_{j=1}^N \delta_j(S_p) w_j$$

$\delta_j(S_p)$  indicates whether observation  $j$  is in subpopulation  $S_p$ ,  $w_j$  is the weight for the  $j$ th observation, and  $N$  is the sample size.

## Marginal effects

`margins` also computes marginal/partial effects. For the marginal effect of continuous covariate  $x$ , `margins` computes

$$\hat{p} = \frac{1}{w_{\cdot}} \sum_{j=1}^N \delta_j(S_p) w_j h(\mathbf{z}_j, \hat{\boldsymbol{\theta}})$$

where

$$h(\mathbf{z}, \boldsymbol{\theta}) = \frac{\partial f(\mathbf{z}, \boldsymbol{\theta})}{\partial x}$$

The marginal effect for level  $k$  of factor variable  $A$  is the simple contrast (a.k.a. difference) comparing its margin with the margin at the base level.

$$h(\mathbf{z}, \boldsymbol{\theta}) = f(\mathbf{z}, \boldsymbol{\theta} | A = k) - f(\mathbf{z}, \boldsymbol{\theta} | A = \text{base})$$

## Fixing covariates and balancing factors

`margins` controls the values in each  $\mathbf{z}$  vector through the *marginlist*, the `at()` option, the `atmeans` option, and the `asbalanced` and `emptycells()` options. Suppose  $\mathbf{z}$  is composed of the elements from the equation specification

$$\mathbf{A} \# \mathbf{B} \times$$

where  $\mathbf{A}$  is a factor variable with  $a$  levels,  $\mathbf{B}$  is a factor variable with  $b$  levels, and  $\mathbf{x}$  is a continuous covariate. To simplify the notation for this discussion, assume the levels of  $\mathbf{A}$  and  $\mathbf{B}$  start with 1 and are contiguous. Then

$$\mathbf{z} = (A_1, \dots, A_a, B_1, \dots, B_b, A_1 B_1, A_1 B_2, \dots, A_a B_b, x, 1)$$

where  $A_i$ ,  $B_j$ , and  $A_i B_j$  represent the indicator values for the factor variables  $\mathbf{A}$  and  $\mathbf{B}$  and the interaction  $\mathbf{A} \# \mathbf{B}$ .

When factor  $\mathbf{A}$  is in the *marginlist*, `margins` replaces  $\mathbf{A}$  with  $i$  and then computes the mean of the subsequent prediction, for  $i = 1, \dots, a$ . When the interaction term  $\mathbf{A} \# \mathbf{B}$  is in the *marginlist*, `margins` replaces  $\mathbf{A}$  with  $i$  and  $\mathbf{B}$  with  $j$ , and then computes the mean of the subsequent prediction, for all combinations of  $i = 1, \dots, a$  and  $j = 1, \dots, b$ .



The `at()` option sets model covariates to fixed values. For example, `at(x=15)` causes `margins` to temporarily set  $x$  to 15 for each observation in the dataset before computing any predictions. Similarly, `at((median) x)` causes `margins` to temporarily set  $x$  to the median of  $x$  using the current dataset.

When factor variable  $A$  is specified as `asbalanced`, `margins` sets each  $A_i$  to  $1/a$ . Thus each  $\mathbf{z}$  vector will look like

$$\mathbf{z} = (1/a, \dots, 1/a, B_1, \dots, B_b, B_1/a, B_2/a, \dots, B_b/a, x, 1)$$

If  $B$  is also specified as `asbalanced`, then each  $B_j$  is set to  $1/b$ , and each  $\mathbf{z}$  vector will look like

$$\mathbf{z} = (1/a, \dots, 1/a, 1/b, \dots, 1/b, 1/ab, 1/ab, \dots, 1/ab, x, 1)$$

If `emptycells(reweight)` is also specified, then `margins` uses a different balancing weight for each element of  $\mathbf{z}$ , depending on how many empty cells the element is associated with. Let  $\delta_{ij}$  indicate that the  $ij$ th cell of  $A\#B$  was observed in the estimation sample.

$$\delta_{ij} = \begin{cases} 0, & A = i \text{ and } B = j \text{ was an empty cell} \\ 1, & \text{otherwise} \end{cases}$$

For the grand margin, the affected elements of  $\mathbf{z}$  and their corresponding balancing weights are

$$A_i = \frac{\sum_j \delta_{ij}}{\sum_k \sum_j \delta_{kj}}$$

$$B_j = \frac{\sum_i \delta_{ij}}{\sum_i \sum_k \delta_{ik}}$$

$$A_i B_j = \frac{\delta_{ij}}{\sum_k \sum_l \delta_{kl}}$$

For the  $j$ th margin of  $B$ , the affected elements of  $\mathbf{z}$  and their corresponding balancing weights are

$$A_i = \frac{\delta_{ij}}{\sum_k \delta_{kj}}$$

$$B_l = \begin{cases} 1, & \text{if } l = j \text{ and not all } \delta_{ij} \text{ are zero} \\ 0, & \text{otherwise} \end{cases}$$

$$A_i B_l = \frac{\delta_{il}}{\sum_k \delta_{kl}} B_l$$

## Estimable functions

The fundamental idea behind estimable functions is clearly defined in the statistical literature for linear models; see [Searle \(1971\)](#). Assume that we are working with the following linear model:

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{e}$$

where  $\mathbf{y}$  is an  $N \times 1$  vector of responses,  $\mathbf{X}$  is an  $N \times p$  matrix of covariate values,  $\mathbf{b}$  is a  $p \times 1$  vector of coefficients, and  $\mathbf{e}$  is a vector of random errors. Assuming a constant variance for the random errors, the normal equations for the least-squares estimator,  $\hat{\mathbf{b}}$ , are

$$\mathbf{X}'\mathbf{X}\hat{\mathbf{b}} = \mathbf{X}'\mathbf{y}$$

When  $\mathbf{X}$  is not of full column rank, we will need a generalized inverse (g-inverse) of  $\mathbf{X}'\mathbf{X}$  to solve for  $\hat{\mathbf{b}}$ . Let  $\mathbf{G}$  be a g-inverse of  $\mathbf{X}'\mathbf{X}$ .

[Searle \(1971\)](#) defines a linear function of the parameters as *estimable* if it is identically equal to some linear function of the expected values of the  $\mathbf{y}$  vector. Let  $\mathbf{H} = \mathbf{G}\mathbf{X}'\mathbf{X}$ . Then this definition simplifies to the following rule:

$$\mathbf{z}\mathbf{b} \text{ is estimable if } \mathbf{z} = \mathbf{z}\mathbf{H}$$

`margins` generalizes this to nonlinear functions by assuming the prediction function  $f(\mathbf{z}, \boldsymbol{\theta})$  is a function of one or more of the linear predictions from the equations in the model that  $\boldsymbol{\theta}$  represents.

$$f(\mathbf{z}, \boldsymbol{\theta}) = h(\mathbf{z}_1\boldsymbol{\beta}_1, \mathbf{z}_2\boldsymbol{\beta}_2, \dots, \mathbf{z}_k\boldsymbol{\beta}_k)$$

$\mathbf{z}_i\boldsymbol{\beta}_i$  is considered estimable if  $\mathbf{z}_i = \mathbf{z}_i\mathbf{H}_i$ , where  $\mathbf{H}_i = \mathbf{G}_i\mathbf{X}_i'\mathbf{X}_i$ ,  $\mathbf{G}_i$  is a g-inverse for  $\mathbf{X}_i'\mathbf{X}_i$ , and  $\mathbf{X}_i$  is the matrix of covariates from the  $i$ th equation of the fitted model. `margins` considers  $p(\boldsymbol{\theta})$  to be estimable if every  $\mathbf{z}_i\boldsymbol{\beta}_i$  is estimable.

## Standard errors conditional on the covariates

By default, `margins` uses the delta method to estimate the variance of  $\hat{p}$ .

$$\widehat{\text{Var}}(\hat{p} | \mathbf{z}) = \mathbf{v}'\mathbf{V}\mathbf{v}$$

where  $\mathbf{V}$  is a variance estimate for  $\hat{\boldsymbol{\theta}}$  and

$$\mathbf{v} = \left. \frac{\partial \hat{p}}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}}$$

This variance estimate is conditional on the  $\mathbf{z}$  vectors used to compute the marginalized predictions.

## Unconditional standard errors

`margins` with the `vce(unconditional)` option uses linearization to estimate the unconditional variance of  $\widehat{\theta}$ . Linearization uses the variance estimator for the total of a score variable for  $\widehat{p}$  as an approximate estimator for  $\text{Var}(\widehat{p})$ ; see [SVY] [variance estimation](#). `margins` requires that the model was fit using some form of linearized variance estimator and that `predict`, `scores` computes the appropriate score values for the linearized variance estimator.

The score for  $\widehat{p}$  from the  $j$ th observation is given by

$$s_j = \frac{\partial \widehat{p}}{\partial w_j} = -\frac{\delta_j(S_p)}{w} \widehat{p} + \frac{\delta_j(S_p)}{w} f(\mathbf{z}_j, \widehat{\theta}) + \frac{1}{w} \sum_{i=1}^N \delta_i(S_p) w_i \frac{\partial f(\mathbf{z}_i, \widehat{\theta})}{\partial w_j}$$

The remaining partial derivative can be decomposed using the chain rule.

$$\frac{\partial f(\mathbf{z}_i, \widehat{\theta})}{\partial w_j} = \left( \frac{\partial f(\mathbf{z}_i, \theta)}{\partial \theta} \bigg|_{\theta=\widehat{\theta}} \right) \left( \frac{\partial \widehat{\theta}}{\partial w_j} \right)'$$

This is the inner product of two vectors, the second of which is not a function of the  $i$  index. Thus the score is

$$s_j = -\frac{\delta_j(S_p)}{w} \widehat{p} + \frac{\delta_j(S_p)}{w} f(\mathbf{z}_j, \widehat{\theta}) + \left( \frac{\partial \widehat{p}}{\partial \theta} \bigg|_{\theta=\widehat{\theta}} \right) \left( \frac{\partial \widehat{\theta}}{\partial w_j} \right)'$$

If  $\widehat{\theta}$  was derived from a system of equations (such as in linear regression or maximum likelihood estimation), then  $\widehat{\theta}$  is the solution to

$$\mathbf{G}(\theta) = \sum_{j=1}^N \delta_j(S_m) w_j \mathbf{g}(\theta, \mathbf{y}_j, \mathbf{x}_j) = \mathbf{0}$$

where  $S_m$  identifies the subpopulation used to fit the model,  $\mathbf{g}()$  is the model's gradient function, and  $\mathbf{y}_j$  and  $\mathbf{x}_j$  are the values of the dependent and independent variables for the  $j$ th observation. We can use linearization to derive a first-order approximation for  $\partial \widehat{\theta} / \partial w_j$ .

$$\mathbf{G}(\widehat{\theta}) \approx \mathbf{G}(\theta_0) + \frac{\partial \mathbf{G}(\theta)}{\partial \theta} \bigg|_{\theta=\theta_0} (\widehat{\theta} - \theta_0)$$

Let  $\mathbf{H}$  be the Hessian matrix

$$\mathbf{H} = \frac{\partial \mathbf{G}(\theta)}{\partial \theta} \bigg|_{\theta=\theta_0}$$

Then

$$\widehat{\theta} \approx \theta_0 + (-\mathbf{H})^{-1} \mathbf{G}(\theta_0)$$

and

$$\frac{\partial \hat{\boldsymbol{\theta}}}{\partial w_j} \approx (-\mathbf{H})^{-1} \frac{\partial \mathbf{G}(\boldsymbol{\theta})}{\partial w_j} \bigg|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} = (-\mathbf{H})^{-1} \delta_j(S_m) \mathbf{g}(\hat{\boldsymbol{\theta}}, \mathbf{y}_j, \mathbf{x}_j)$$

The computed value of the score for  $\hat{p}$  for the  $j$ th observation is

$$s_j = \mathbf{v}' \mathbf{u}_j$$

where

$$\mathbf{v} = \begin{bmatrix} -\frac{\hat{p}}{w} \\ \frac{1}{w} \\ \frac{\partial \hat{p}}{\partial \boldsymbol{\theta}} (-\mathbf{H})^{-1} \end{bmatrix}$$

and

$$\mathbf{u}_j = \begin{bmatrix} \delta_j(S_p) \\ \delta_j(S_p) f(\mathbf{z}_j, \hat{\boldsymbol{\theta}}) \\ \delta_j(S_m) \mathbf{g}(\hat{\boldsymbol{\theta}}, \mathbf{y}_j, \mathbf{x}_j) \end{bmatrix}$$

Thus the variance estimate for  $\hat{p}$  is

$$\widehat{\text{Var}}(\hat{p}) = \mathbf{v}' \widehat{\text{Var}}(\hat{\mathbf{U}}) \mathbf{v}$$

where

$$\hat{\mathbf{U}} = \sum_{j=1}^N w_j \mathbf{u}_j$$

`margins` uses the model-based variance estimates for  $(-\mathbf{H})^{-1}$  and the scores from `predict` for  $\mathbf{g}(\hat{\boldsymbol{\theta}}, \mathbf{y}_j, \mathbf{x}_j)$ .

## References

- Bartus, T. 2005. Estimation of marginal effects using `margeff`. *Stata Journal* 5: 309–329.
- Chang, I. M., R. Gelman, and M. Pagano. 1982. Corrected group prognostic curves and summary statistics. *Journal of Chronic Diseases* 35: 669–674.
- Gould, W. W. 1996. `crc43: Wald test of nonlinear hypotheses after model estimation`. *Stata Technical Bulletin* 29: 2–4. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 15–18. College Station, TX: Stata Press.
- Graubard, B. I., and E. L. Korn. 2004. Predictive margins with survey data. *Biometrics* 55: 652–659.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- Korn, E. L., and B. I. Graubard. 1999. *Analysis of Health Surveys*. New York: Wiley.
- Lane, P. W., and J. A. Nelder. 1982. Analysis of covariance and standardization as instances of prediction. *Biometrics* 38: 613–621.

- Phillips, P. C. B., and J. Y. Park. 1988. On the formulation of Wald tests of nonlinear restrictions. *Econometrica* 56: 1065–1083.
- Searle, S. R. 1971. *Linear Models*. New York: Wiley.
- . 1997. *Linear Models for Unbalanced Data*. New York: Wiley.
- Searle, S. R., F. M. Speed, and G. A. Milliken. 1980. Population marginal means in the linear model: An alternative to least squares means. *American Statistician* 34: 216–221.
- Wooldridge, J. M. 2002. *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA: MIT Press.

## Also see

- [R] [margins postestimation](#) — Postestimation tools for margins
- [R] [nlcom](#) — Nonlinear combinations of estimators
- [R] [predict](#) — Obtain predictions, residuals, etc., after estimation
- [R] [predictnl](#) — Obtain nonlinear predictions, standard errors, etc., after estimation

Description

The following postestimation commands are available after `margins`, `post`:

| command                | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>     | estimation sample summary; <code>estat summarize</code> only                                        |
| <code>estimates</code> | cataloging estimation results                                                                       |
| <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

Remarks

Continuing with the example from [Example 8: Margins of interactions](#) in [R] `margins`, we use the dataset and reestimate the logistic model of outcome:

```
. use http://www.stata-press.com/data/r11/margex
(Artificial data for margins)
. logistic outcome sex##group age
(output omitted)
```

We then estimate the margins for males and females and post the margins as estimation results with a full VCE.

```
. margins sex, post
Predictive margins Number of obs = 3000
Model VCE : OIM
Expression : Pr(outcome), predict()
```

|     | Delta-method |           | z     | P> z  | [95% Conf. Interval] |          |
|-----|--------------|-----------|-------|-------|----------------------|----------|
|     | Margin       | Std. Err. |       |       |                      |          |
| sex |              |           |       |       |                      |          |
| 0   | .1600644     | .0125653  | 12.74 | 0.000 | .1354368             | .184692  |
| 1   | .1966902     | .0100043  | 19.66 | 0.000 | .1770821             | .2162983 |

We can now use `nlcom` (see [R] `nlcom`) to estimate a risk ratio of females to males using the average probabilities for females and males posted by `margins`:

```
. nlcom (risk_ratio: _b[1.sex] / _b[0.sex])
 risk_ratio: _b[1.sex] / _b[0.sex]
```

|            | Coef.    | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|------------|----------|-----------|-------|-------|----------------------|----------|
| risk_ratio | 1.228819 | .1149538  | 10.69 | 0.000 | 1.003514             | 1.454124 |

We could similarly estimate the average risk difference between females and males:

```
. nlcom (risk_diff: _b[1.sex] - _b[0.sex])
 risk_diff: _b[1.sex] - _b[0.sex]
```

|           | Coef.    | Std. Err. | z    | P> z  | [95% Conf. Interval] |         |
|-----------|----------|-----------|------|-------|----------------------|---------|
| risk_diff | .0366258 | .0160632  | 2.28 | 0.023 | .0051425             | .068109 |

# Also see

[\[R\] margins](#) — Marginal means, predictive margins, and marginal effects

# Title

**matsize** — Set the maximum number of variables in a model

## Syntax

```
set matsize # [, permanently]
```

where  $10 \leq \# \leq 11000$  for Stata/MP and Stata/SE and where  $10 \leq \# \leq 800$  for Stata/IC.

## Description

`set matsize` sets the maximum number of variables that can be included in any of Stata's estimation commands.

For Stata/MP and Stata/SE, the default value is 400, but it may be changed upward or downward. The upper limit is 11,000.

For Stata/IC, the initial value is 400, but it may be changed upward or downward. The upper limit is 800.

This command may not be used with Small Stata; `matsize` is permanently frozen at 100.

Changing `matsize` has no effect on Mata.

## Option

`permanently` specifies that, in addition to making the change right now, the `matsize` setting be remembered and become the default setting when you invoke Stata.

## Remarks

`set matsize` controls the internal size of matrices that Stata uses. The default of 400 for Stata/IC, for instance, means that linear regression models are limited to 198 independent variables—198 because the constant uses one position and the dependent variable another, making a total of 200.

You may change `matsize` with data in memory, but increasing `matsize` increases the amount of memory consumed by Stata, increasing the probability of page faults and thus of making Stata run more slowly.

### ► Example 1

We wish to fit a model of `y` on the variables `x1` through `x400`. Without thinking, we type

```
. regress y x1-x400
matsize too small
 You have attempted to create a matrix with more than 400 rows or columns
 or to fit a model with more than 400 variables plus ancillary parameters.
 You need to increase matsize by using the set matsize command; see help
 matsize.
r(908);
```



We realize that we need to increase `matsize`, so we type

```
. set matsize 450
. regress y x1-x400
(output omitted)
```

◀

Programmers should note that the current setting of `matsize` is stored as the c-class value `c(matsize)`; see [P] [creturn](#).

## Also see

- [R] [query](#) — Display system parameters
- [D] [memory](#) — Memory size considerations
- [U] [6 Setting the size of memory](#)

## Syntax

Maximum likelihood optimization

```
mle_cmd ... [, options]
```

Set default maximum iterations

```
set maxiter # [, permanently]
```

| options                                    | description                                                                          |
|--------------------------------------------|--------------------------------------------------------------------------------------|
| <u>d</u> ifficult                          | use a different stepping algorithm in nonconcave regions                             |
| <u>t</u> echnique( <i>algorithm_spec</i> ) | maximization technique                                                               |
| <u>i</u> terate(#)                         | perform maximum of # iterations; default is <code>iterate(16000)</code>              |
| [ <u>n</u> o] <u>l</u> og                  | display an iteration log of the log likelihood; typically, the default               |
| <u>t</u> race                              | display current parameter vector in iteration log                                    |
| <u>g</u> radient                           | display current gradient vector in iteration log                                     |
| <u>s</u> howstep                           | report steps within an iteration in iteration log                                    |
| <u>h</u> essian                            | display current negative Hessian matrix in iteration log                             |
| <u>s</u> howtolerance                      | report the calculated result that is compared to the effective convergence criterion |
| <u>t</u> olerance(#)                       | tolerance for the coefficient vector; see <a href="#">Options</a> for the defaults   |
| <u>l</u> tolerance(#)                      | tolerance for the log likelihood; see <a href="#">Options</a> for the defaults       |
| <u>n</u> rtolerance(#)                     | tolerance for the scaled gradient; see <a href="#">Options</a> for the defaults      |
| <u>n</u> onrtolerance                      | ignore the <code>nrtolerance()</code> option                                         |
| <u>f</u> rom( <i>init_specs</i> )          | initial values for the coefficients                                                  |

where *algorithm\_spec* is

```
algorithm [# [algorithm [#]] ...]
```

*algorithm* is { `nr` | `bhhh` | `dfp` | `bfgs` }

and *init\_specs* is one of

```
matname [, skip copy]
{ [eqname:] name = # | /eqname = # } [...]
[# ...], copy
```

## Description

All Stata commands maximize likelihood functions using `moptimize()` and `optimize()`; see [Methods and formulas](#) below. Commands use the Newton–Raphson method with step halving and special fixups when they encounter nonconcave regions of the likelihood. For details, see [M-5] `moptimize()` and [M-5] `optimize()`. For more information about programming maximum likelihood estimators in ado-files and Mata, see [R] `ml` and the third edition of *Maximum Likelihood Estimation with Stata* (Gould, Pitblado, and Sribney 2006).

`set maxiter` specifies the default maximum number of iterations for estimation commands that iterate. The initial value is 16000, and # can be 0 to 16000. To change the maximum number of iterations performed by a particular estimation command, you need not reset `maxiter`; you can specify the `iterate(#)` option. When `iterate(#)` is not specified, the `maxiter` value is used.

## Maximization options

`difficult` specifies that the likelihood function is likely to be difficult to maximize because of nonconcave regions. When the message “not concave” appears repeatedly, `ml`’s standard stepping algorithm may not be working well. `difficult` specifies that a different stepping algorithm be used in nonconcave regions. There is no guarantee that `difficult` will work better than the default; sometimes it is better and sometimes it is worse. You should use the `difficult` option only when the default stepper declares convergence and the last iteration is “not concave” or when the default stepper is repeatedly issuing “not concave” messages and producing only tiny improvements in the log likelihood.

`technique(algorithm_spec)` specifies how the likelihood function is to be maximized. The following algorithms are allowed. For details, see Gould, Pitblado, and Sribney (2006).

`technique(nr)` specifies Stata’s modified Newton–Raphson (NR) algorithm.

`technique(bhhh)` specifies the Berndt–Hall–Hall–Hausman (BHHH) algorithm.

`technique(dfpr)` specifies the Davidon–Fletcher–Powell (DFP) algorithm.

`technique(bfgs)` specifies the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm.

The default is `technique(nr)`.

You can switch between algorithms by specifying more than one in the `technique()` option.

By default, an algorithm is used for five iterations before switching to the next algorithm. To specify a different number of iterations, include the number after the `technique` in the option. For example, specifying `technique(bhhh 10 nr 1000)` requests that `ml` perform 10 iterations with the BHHH algorithm followed by 1000 iterations with the NR algorithm, and then switch back to BHHH for 10 iterations, and so on. The process continues until convergence or until the maximum number of iterations is reached.

`iterate(#)` specifies the maximum number of iterations. When the number of iterations equals `iterate()`, the optimizer stops and presents the current results. If convergence is declared before this threshold is reached, it will stop when convergence is declared. Specifying `iterate(0)` is useful for viewing results evaluated at the initial value of the coefficient vector. Specifying `iterate(0)` and `from()` together allows you to view results evaluated at a specified coefficient vector; however, not all commands allow the `from()` option. The default value of `iterate(#)` for both estimators programmed internally and estimators programmed with `ml` is the current value of `set maxiter`, which is `iterate(16000)` by default.

`log` and `nolog` specify whether an iteration log showing the progress of the log likelihood is to be displayed. For most commands, the log is displayed by default, and `nolog` suppresses it. For a

few commands (such as the `svy` maximum likelihood estimators), you must specify `log` to see the log.

`trace` adds to the iteration log a display of the current parameter vector.

`gradient` adds to the iteration log a display of the current gradient vector.

`showstep` adds to the iteration log a report on the steps within an iteration. This option was added so that developers at StataCorp could view the stepping when they were improving the `ml` optimizer code. At this point, it mainly provides entertainment.

`hessian` adds to the iteration log a display of the current negative Hessian matrix.

`showtolerance` adds to the iteration log the calculated value that is compared with the effective convergence criterion at the end of each iteration. Until convergence is achieved, the smallest calculated value is reported.

`shownrtolerance` is a synonym of `showtolerance`.

---

Below we describe the three convergence tolerances. Convergence is declared when the `nrtolerance()` criterion is met and either the `tolerance()` or the `ltolerance()` criterion is also met.

`tolerance(#)` specifies the tolerance for the coefficient vector. When the relative change in the coefficient vector from one iteration to the next is less than or equal to `tolerance()`, the `tolerance()` convergence criterion is satisfied.

`tolerance(1e-4)` is the default for estimators programmed with `ml`.

`tolerance(1e-6)` is the default.

`ltolerance(#)` specifies the tolerance for the log likelihood. When the relative change in the log likelihood from one iteration to the next is less than or equal to `ltolerance()`, the `ltolerance()` convergence is satisfied.

`ltolerance(0)` is the default for estimators programmed with `ml`.

`ltolerance(1e-7)` is the default.

`nrtolerance(#)` specifies the tolerance for the scaled gradient. Convergence is declared when  $\mathbf{gH}^{-1}\mathbf{g}' < \text{nrtolerance}()$ . The default is `nrtolerance(1e-5)`.

`nonrtolerance` specifies that the default `nrtolerance()` criterion be turned off.

---

`from()` specifies initial values for the coefficients. Not all estimators in Stata support this option. You can specify the initial values in one of three ways: by specifying the name of a vector containing the initial values (e.g., `from(b0)`, where `b0` is a properly labeled vector); by specifying coefficient names with the values (e.g., `from(age=2.1 /sigma=7.4)`); or by specifying a list of values (e.g., `from(2.1 7.4, copy)`). `from()` is intended for use when doing bootstraps (see [\[R\] bootstrap](#)) and in other special situations (e.g., with `iterate(0)`). Even when the values specified in `from()` are close to the values that maximize the likelihood, only a few iterations may be saved. Poor values in `from()` may lead to convergence problems.

`skip` specifies that any parameters found in the specified initialization vector that are not also found in the model be ignored. The default action is to issue an error message.

`copy` specifies that the list of values or the initialization vector be copied into the initial-value vector by position rather than by name.

## Option for set maxiter

permanently specifies that, in addition to making the change right now, the `maxiter` setting be remembered and become the default setting when you invoke Stata.

## Remarks

Only in rare circumstances would you ever need to specify any of these options, except `nolog`. The `nolog` option is useful for reducing the amount of output appearing in log files.

The following is an example of an iteration log:

```
Iteration 0: log likelihood = -3791.0251
Iteration 1: log likelihood = -3761.738
Iteration 2: log likelihood = -3758.0632 (not concave)
Iteration 3: log likelihood = -3758.0447
Iteration 4: log likelihood = -3757.5861
Iteration 5: log likelihood = -3757.474
Iteration 6: log likelihood = -3757.4613
Iteration 7: log likelihood = -3757.4606
Iteration 8: log likelihood = -3757.4606


```

At iteration 8, the model converged. The message “not concave” at the second iteration is notable. This example was produced using the `heckman` command; its likelihood is not globally concave, so it is not surprising that this message sometimes appears. The other message that is occasionally seen is “backed up”. Neither of these messages should be of any concern unless they appear at the final iteration.

If a “not concave” message appears at the last step, there are two possibilities. One is that the result is valid, but there is collinearity in the model that the command did not otherwise catch. Stata checks for obvious collinearity among the independent variables before performing the maximization, but strange collinearities or near collinearities can sometimes arise between coefficients and ancillary parameters. The second, more likely cause for a “not concave” message at the final step is that the optimizer entered a flat region of the likelihood and prematurely declared convergence.

If a “backed up” message appears at the last step, there are also two possibilities. One is that Stata found a perfect maximum and could not step to a better point; if this is the case, all is fine, but this is a highly unlikely occurrence. The second is that the optimizer worked itself into a bad concave spot where the computed gradient and Hessian gave a bad direction for stepping.

If either of these messages appears at the last step, perform the maximization again with the `gradient` option. If the gradient goes to zero, the optimizer has found a maximum that may not be unique but is a maximum. From the standpoint of maximum likelihood estimation, this is a valid result. If the gradient is not zero, it is not a valid result, and you should try tightening up the convergence criterion, or try `ltol(0) toln(1e-7)` to see if the optimizer can work its way out of the bad region.

If you get repeated “not concave” steps with little progress being made at each step, try specifying the `difficult` option. Sometimes `difficult` works wonderfully, reducing the number of iterations and producing convergence at a good (i.e., concave) point. Other times, `difficult` works poorly, taking much longer to converge than the default stepper.

## Saved results

Maximum likelihood estimators save the following in `e()`:

### Scalars

|                            |                                                     |                                                                                                                                      |
|----------------------------|-----------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <code>e(N)</code>          | number of observations                              | always saved                                                                                                                         |
| <code>e(k)</code>          | number of parameters                                | always saved                                                                                                                         |
| <code>e(k_eq)</code>       | number of equations                                 | usually saved                                                                                                                        |
| <code>e(k_eq_model)</code> | number of equations to include in a model Wald test | usually saved                                                                                                                        |
| <code>e(k_dv)</code>       | number of dependent variables                       | usually saved                                                                                                                        |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints      | saved if command support constraints                                                                                                 |
| <code>e(df_m)</code>       | model degrees of freedom                            | always saved                                                                                                                         |
| <code>e(r2_p)</code>       | pseudo- <i>R</i> -squared                           | sometimes saved                                                                                                                      |
| <code>e(l1)</code>         | log likelihood                                      | always saved                                                                                                                         |
| <code>e(l1_0)</code>       | log likelihood, constant-only model                 | saved when constant-only model is fit                                                                                                |
| <code>e(N_clust)</code>    | number of clusters                                  | saved when <code>vce(cluster <i>clustvar</i>)</code> is specified; see <a href="#">[U] 20.16 Obtaining robust variance estimates</a> |
| <code>e(chi2)</code>       | $\chi^2$                                            | usually saved                                                                                                                        |
| <code>e(p)</code>          | significance of model of test                       | usually saved                                                                                                                        |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                           | always saved                                                                                                                         |
| <code>e(rank0)</code>      | rank of <code>e(V)</code> for constant-only model   | saved when constant-only model is fit                                                                                                |
| <code>e(ic)</code>         | number of iterations                                | usually saved                                                                                                                        |
| <code>e(rc)</code>         | return code                                         | usually saved                                                                                                                        |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                         | usually saved                                                                                                                        |

### Macros

|                                 |                                                                          |                                                                                                                                      |
|---------------------------------|--------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <code>e(cmd)</code>             | name of command                                                          | always saved                                                                                                                         |
| <code>e(cmdline)</code>         | command as typed                                                         | always saved                                                                                                                         |
| <code>e(depvar)</code>          | names of dependent variables                                             | always saved                                                                                                                         |
| <code>e(wtype)</code>           | weight type                                                              | saved when weights are specified or implied                                                                                          |
| <code>e(wexp)</code>            | weight expression                                                        | saved when weights are specified or implied                                                                                          |
| <code>e(title)</code>           | title in estimation output                                               | usually saved by commands using <code>ml</code>                                                                                      |
| <code>e(clustvar)</code>        | name of cluster variable                                                 | saved when <code>vce(cluster <i>clustvar</i>)</code> is specified; see <a href="#">[U] 20.16 Obtaining robust variance estimates</a> |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                  | usually saved                                                                                                                        |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                           | saved when command allows ( <code>vce()</code> )                                                                                     |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                            | sometimes saved                                                                                                                      |
| <code>e(opt)</code>             | type of optimization                                                     | always saved                                                                                                                         |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization | always saved                                                                                                                         |
| <code>e(ml_method)</code>       | type of <code>ml</code> method                                           | always saved by commands using <code>ml</code>                                                                                       |
| <code>e(user)</code>            | name of likelihood-evaluator program                                     | always saved                                                                                                                         |
| <code>e(technique)</code>       | from <code>technique()</code> option                                     | sometimes saved                                                                                                                      |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular              | sometimes saved                                                                                                                      |
| <code>e(crittype)</code>        | optimization criterion                                                   | always saved                                                                                                                         |
| <code>e(properties)</code>      | estimator properties                                                     | always saved                                                                                                                         |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                           | usually saved                                                                                                                        |

## Matrices

|                              |                                              |                                                                       |
|------------------------------|----------------------------------------------|-----------------------------------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           | always saved                                                          |
| <code>e(Cns)</code>          | constraints matrix                           | sometimes saved                                                       |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          | usually saved                                                         |
| <code>e(gradient)</code>     | gradient vector                              | usually saved                                                         |
| <code>e(V)</code>            | variance–covariance matrix of the estimators | always saved                                                          |
| <code>e(V_modelbased)</code> | model-based variance                         | only saved when <code>e(V)</code> is neither the OIM nor OPG variance |

## Functions

|                        |                         |              |
|------------------------|-------------------------|--------------|
| <code>e(sample)</code> | marks estimation sample | always saved |
|------------------------|-------------------------|--------------|

See *Saved results* in the manual entry for any maximum likelihood estimator for a list of returned results.

## Methods and formulas

Optimization is currently performed by `moptimize()` and `optimize()`, with the former implemented in terms of the latter; see [M-5] `moptimize()` and [M-5] `optimize()`. Some estimators use `moptimize()` and `optimize()` directly, and others use the `ml` ado-file interface to `moptimize()`.

Prior to Stata 11, Stata had three separate optimization engines: an internal one used by estimation commands implemented in C code; `ml` implemented in ado-code separately from `moptimize()` and used by most estimators; and `moptimize()` and `optimize()` used by a few recently written estimators. These days, the internal optimizer and the old version of `ml` are used only under version control. In addition, `arch` and `arma` (see [TS] `arch` and [TS] `arma`) are currently implemented using the old `ml`.

Let  $L_1$  be the log likelihood of the full model (i.e., the log-likelihood value shown on the output), and let  $L_0$  be the log likelihood of the “constant-only” model. The likelihood-ratio  $\chi^2$  model test is defined as  $2(L_1 - L_0)$ . The pseudo- $R^2$  (McFadden 1974) is defined as  $1 - L_1/L_0$ . This is simply the log likelihood on a scale where 0 corresponds to the “constant-only” model and 1 corresponds to perfect prediction for a discrete model (in which case the overall log likelihood is 0).

Some maximum likelihood routines can report coefficients in an exponentiated form, e.g., odds ratios in `logistic`. Let  $b$  be the unexponentiated coefficient,  $s$  its standard error, and  $b_0$  and  $b_1$  the reported confidence interval for  $b$ . In exponentiated form, the point estimate is  $e^b$ , the standard error  $e^b s$ , and the confidence interval  $e^{b_0}$  and  $e^{b_1}$ . The displayed  $Z$  (or  $t$ ) statistics and  $p$ -values are the same as those for the unexponentiated results. This is justified because  $e^b = 1$  and  $b = 0$  are equivalent hypotheses, and normality is more likely to hold in the  $b$  metric.

## References

- Gould, W. W., J. S. Pitblado, and W. M. Sribney. 2006. *Maximum Likelihood Estimation with Stata*. 3rd ed. College Station, TX: Stata Press.
- McFadden, D. L. 1974. Conditional logit analysis of qualitative choice behavior. In *Frontiers in Econometrics*, ed. P. Zarembka, 105–142. New York: Academic Press.

## Also see

[R] **ml** — Maximum likelihood estimation

[SVY] **ml for svy** — Maximum pseudolikelihood estimation for survey data

[M-5] **moptimize()** — Model optimization

[M-5] **optimize()** — Function optimization



Syntax

```
mean varlist [if] [in] [weight] [, options]
```

| options                                  | description                                                                                                                      |
|------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| Model                                    |                                                                                                                                  |
| <code>stdize(varname)</code>             | variable identifying strata for standardization                                                                                  |
| <code>stdweight(varname)</code>          | weight variable for standardization                                                                                              |
| <code>nostdrescale</code>                | do not rescale the standard weight variable                                                                                      |
| if/in/over                               |                                                                                                                                  |
| <code>over(varlist [ , nolabel ])</code> | group over subpopulations defined by <i>varlist</i> ; optionally, suppress group labels                                          |
| SE/Cluster                               |                                                                                                                                  |
| <code>vce(vcetype)</code>                | <i>vcetype</i> may be <code>analytic</code> , <code>cluster clustvar</code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                                |                                                                                                                                  |
| <code>level(#)</code>                    | set confidence level; default is <code>level(95)</code>                                                                          |
| <code>noheader</code>                    | suppress table header                                                                                                            |
| <code>nolegend</code>                    | suppress table legend                                                                                                            |
| <sup>†</sup> <code>coeflegend</code>     | display coefficients' legend instead of coefficient table                                                                        |

<sup>†</sup>`coeflegend` does not appear in the dialog box.

`bootstrap`, `jackknife`, `mi estimate`, `rolling`, `statsby`, and `svy` are allowed; see [U] 11.1.10 Prefix commands.

`vce(bootstrap)` and `vce(jackknife)` are not allowed with the `mi estimate` prefix.

Weights are not allowed with the `bootstrap` prefix.

`aweight`s are not allowed with the `jackknife` prefix.

`vce()` and weights are not allowed with the `svy` prefix.

`fweights`, `aweight`s, `iweight`s, and `pweight`s are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Means

Description

`mean` produces estimates of means, along with standard errors.

## Options

### Model

`stdize(varname)` specifies that the point estimates be adjusted by direct standardization across the strata identified by *varname*. This option requires the `stdweight()` option.

`stdweight(varname)` specifies the weight variable associated with the standard strata identified in the `stdize()` option. The standardization weights must be constant within the standard strata.

`nostdrescale` prevents the standardization weights from being rescaled within the `over()` groups. This option requires `stdize()` but is ignored if the `over()` option is not specified.

### if/in/over

`over(varlist [ , no label ])` specifies that estimates be computed for multiple subpopulations, which are identified by the different values of the variables in *varlist*.

When this option is supplied with one variable name, such as `over(varname)`, the value labels of *varname* are used to identify the subpopulations. If *varname* does not have labeled values (or there are unlabeled values), the values themselves are used, provided that they are nonnegative integers. Noninteger values, negative values, and labels that are not valid Stata names are substituted with a default identifier.

When `over()` is supplied with multiple variable names, each subpopulation is assigned a unique default identifier.

`no label` requests that value labels attached to the variables identifying the subpopulations be ignored.

### SE/Cluster

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

`vce(analytic)`, the default, uses the analytically derived variance estimator associated with the sample mean.

### Reporting

`level(#)`; see [R] [estimation options](#).

`noheader` prevents the table header from being displayed. This option implies `nolegend`.

`nolegend` prevents the table legend identifying the subpopulations from being displayed.

The following option is available with `mean` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

### ► Example 1

Using the fuel data from [example 2](#) of [R] [ttest](#), we estimate the average mileage of the cars without the fuel treatment (`mpg1`) and those with the fuel treatment (`mpg2`).

```
. use http://www.stata-press.com/data/r11/fuel
. mean mpg1 mpg2
```

|                 |       |               |                      |          |
|-----------------|-------|---------------|----------------------|----------|
| Mean estimation |       | Number of obs | =                    | 12       |
|                 | Mean  | Std. Err.     | [95% Conf. Interval] |          |
| mpg1            | 21    | .7881701      | 19.26525             | 22.73475 |
| mpg2            | 22.75 | .9384465      | 20.68449             | 24.81551 |

Using these results, we can test the equality of the mileage between the two groups of cars.

```
. test mpg1 = mpg2
(1) mpg1 - mpg2 = 0
 F(1, 11) = 5.04
 Prob > F = 0.0463
```



➤ Example 2

In example 1, the joint observations of mpg1 and mpg2 were used to estimate a covariance between their means.

```
. matrix list e(V)
symmetric e(V)[2,2]
 mpg1 mpg2
mpg1 .62121212
mpg2 .4469697 .88068182
```

If the data were organized this way out of convenience but the two variables represent independent samples of cars (coincidentally of the same sample size), we should reshape the data and use the `over()` option to ensure that the covariance between the means is zero.

```
. use http://www.stata-press.com/data/r11/fuel
. stack mpg1 mpg2, into(mpg) clear
. mean mpg, over(_stack)
```

|                 |               |               |   |    |
|-----------------|---------------|---------------|---|----|
| Mean estimation |               | Number of obs | = | 24 |
|                 | 1: _stack = 1 |               |   |    |
|                 | 2: _stack = 2 |               |   |    |

| Over | Mean | Std. Err. | [95% Conf. Interval] |                   |
|------|------|-----------|----------------------|-------------------|
| mpg  | 1    | 21        | .7881701             | 19.36955 22.63045 |
|      | 2    | 22.75     | .9384465             | 20.80868 24.69132 |

```
. matrix list e(V)
symmetric e(V)[2,2]
 mpg: mpg:
 1 2
mpg:1 .62121212
mpg:2 0 .88068182
```

Now we can test the equality of the mileage between the two independent groups of cars.

```
. test [mpg]1 = [mpg]2
(1) [mpg]1 - [mpg]2 = 0
 F(1, 23) = 2.04
 Prob > F = 0.1667
```



➤ Example 3: standardized means

Suppose that we collected the blood pressure data from [example 2](#) of [\[R\] dstdize](#), and we wish to obtain standardized high blood pressure rates for each city in 1990 and 1992, using, as the standard, the age, sex, and race distribution of the four cities and two years combined. Our rate is really the mean of a variable that indicates whether a sampled individual has high blood pressure. First, we generate the strata and weight variables from our standard distribution, and then use `mean` to compute the rates.

```
. use http://www.stata-press.com/data/r11/hbp, clear
. egen strata = group(age race sex) if inlist(year, 1990, 1992)
(675 missing values generated)
. by strata, sort: gen stdw = _N
. mean hbp, over(city year) stdize(strata) stdweight(stdw)

Mean estimation
N. of std strata = 24 Number of obs = 455

 Over: city year
 _subpop_1: 1 1990
 _subpop_2: 1 1992
 _subpop_3: 2 1990
 _subpop_4: 2 1992
 _subpop_5: 3 1990
 _subpop_6: 3 1992
 _subpop_7: 5 1990
 _subpop_8: 5 1992
```

| Over      | Mean     | Std. Err. | [95% Conf. Interval] |          |
|-----------|----------|-----------|----------------------|----------|
| hbp       |          |           |                      |          |
| _subpop_1 | .058642  | .0296273  | .0004182             | .1168657 |
| _subpop_2 | .0117647 | .0113187  | -.0104789            | .0340083 |
| _subpop_3 | .0488722 | .0238958  | .0019121             | .0958322 |
| _subpop_4 | .014574  | .007342   | .0001455             | .0290025 |
| _subpop_5 | .1011211 | .0268566  | .0483425             | .1538998 |
| _subpop_6 | .0810577 | .0227021  | .0364435             | .1256719 |
| _subpop_7 | .0277778 | .0155121  | -.0027066            | .0582622 |
| _subpop_8 | .0548926 | .         | .                    | .        |

The standard error of the high blood pressure rate estimate is missing for city 5 in 1992 because there was only one individual with high blood pressure; that individual was the only person observed in the stratum of white males 30–35 years old.

By default, `mean` rescales the standard weights within the `over()` groups. In the following, we use the `nostdrescale` option to prevent this, thus reproducing the results in [\[R\] dstdize](#).

```
. mean hbp, over(city year) nolegend stdize(strata) stdweight(stdw)
> nostdrescale

Mean estimation
N. of std strata = 24 Number of obs = 455
```

| Over      | Mean     | Std. Err. | [95% Conf. Interval] |          |
|-----------|----------|-----------|----------------------|----------|
| hbp       |          |           |                      |          |
| _subpop_1 | .0073302 | .0037034  | .0000523             | .0146082 |
| _subpop_2 | .0015432 | .0014847  | -.0013745            | .004461  |
| _subpop_3 | .0078814 | .0038536  | .0003084             | .0154544 |
| _subpop_4 | .0025077 | .0012633  | .0000025             | .0049904 |
| _subpop_5 | .0155271 | .0041238  | .007423              | .0236312 |
| _subpop_6 | .0081308 | .0022772  | .0036556             | .012606  |
| _subpop_7 | .0039223 | .0021904  | -.0003822            | .0082268 |
| _subpop_8 | .0088735 | 0         | .                    | .        |



## Saved results

mean saves the following in `e()`:

### Scalars

|                          |                           |                      |                                          |
|--------------------------|---------------------------|----------------------|------------------------------------------|
| <code>e(N)</code>        | number of observations    | <code>e(k_eq)</code> | number of equations in <code>e(b)</code> |
| <code>e(N_over)</code>   | number of subpopulations  | <code>e(df_r)</code> | sample degrees of freedom                |
| <code>e(N_stdize)</code> | number of standard strata | <code>e(rank)</code> | rank of <code>e(V)</code>                |
| <code>e(N_clust)</code>  | number of clusters        |                      |                                          |

### Macros

|                           |                                              |                               |                                                |
|---------------------------|----------------------------------------------|-------------------------------|------------------------------------------------|
| <code>e(cmd)</code>       | mean                                         | <code>e(over)</code>          | <i>varlist</i> from <code>over()</code>        |
| <code>e(cmdline)</code>   | command as typed                             | <code>e(over_labels)</code>   | labels from <code>over()</code> variables      |
| <code>e(varlist)</code>   | <i>varlist</i>                               | <code>e(over_namelist)</code> | names from <code>e(over_labels)</code>         |
| <code>e(stdize)</code>    | <i>varname</i> from <code>stdize()</code>    | <code>e(vce)</code>           | <i>vcetype</i> specified in <code>vce()</code> |
| <code>e(stdweight)</code> | <i>varname</i> from <code>stdweight()</code> | <code>e(vctype)</code>        | title used to label Std. Err.                  |
| <code>e(wtype)</code>     | weight type                                  | <code>e(properties)</code>    | b V                                            |
| <code>e(wexp)</code>      | weight expression                            | <code>e(estat_cmd)</code>     | program used to implement <code>estat</code>   |
| <code>e(title)</code>     | title in estimation output                   | <code>e(marginsnotok)</code>  | predictions disallowed by <code>margins</code> |
| <code>e(cluster)</code>   | name of cluster variable                     |                               |                                                |

### Matrices

|                           |                                                              |
|---------------------------|--------------------------------------------------------------|
| <code>e(b)</code>         | vector of mean estimates                                     |
| <code>e(V)</code>         | (co)variance estimates                                       |
| <code>e(_N)</code>        | vector of numbers of nonmissing observations                 |
| <code>e(_N_stdsum)</code> | number of nonmissing observations within the standard strata |
| <code>e(_p_stdize)</code> | standardizing proportions                                    |
| <code>e(error)</code>     | error code corresponding to <code>e(b)</code>                |

### Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

mean is implemented as an ado-file.

Methods and formulas are presented under the following headings:

- [The mean estimator](#)
- [Survey data](#)
- [The survey mean estimator](#)
- [The standardized mean estimator](#)
- [The poststratified mean estimator](#)
- [The standardized poststratified mean estimator](#)
- [Subpopulation estimation](#)

## The mean estimator

Let  $y$  be the variable on which we want to calculate the mean and  $y_j$  an individual observation on  $y$ , where  $j = 1, \dots, n$  and  $n$  is the sample size. Let  $w_j$  be the weight, and if no weight is specified, define  $w_j = 1$  for all  $j$ . For `aweight`s, the  $w_j$  are normalized to sum to  $n$ . See the next section for `pweight`d data.

Let  $W$  be the sum of the weights

$$W = \sum_{j=1}^n w_j$$

The mean is defined as

$$\bar{y} = \frac{1}{W} \sum_{j=1}^n w_j y_j$$

The default variance estimator for the mean is

$$\widehat{V}(\bar{y}) = \frac{1}{W(W-1)} \sum_{j=1}^n w_j (y_j - \bar{y})^2$$

The standard error of the mean is the square root of the variance.

If  $x$ ,  $x_j$ , and  $\bar{x}$  are similarly defined for another variable (observed jointly with  $y$ ), the covariance estimator between  $\bar{x}$  and  $\bar{y}$  is

$$\widehat{\text{Cov}}(\bar{x}, \bar{y}) = \frac{1}{W(W-1)} \sum_{j=1}^n w_j (x_j - \bar{x})(y_j - \bar{y})$$

## Survey data

See [\[SVY\] variance estimation](#), [\[SVY\] direct standardization](#), and [\[SVY\] poststratification](#) for discussions that provide background information for the following formulas. The following formulas are derived from the fact that the mean is a special case of the ratio estimator where the denominator variable is one,  $x_j = 1$ ; see [\[R\] ratio](#).

## The survey mean estimator

Let  $Y_j$  be a survey item for the  $j$ th individual in the population, where  $j = 1, \dots, M$  and  $M$  is the size of the population. The associated population mean for the item of interest is  $\bar{Y} = Y/M$  where

$$Y = \sum_{j=1}^M Y_j$$

Let  $y_j$  be the survey item for the  $j$ th sampled individual from the population, where  $j = 1, \dots, m$  and  $m$  is the number of observations in the sample.

The estimator for the mean is  $\bar{y} = \hat{Y}/\hat{M}$ , where

$$\hat{Y} = \sum_{j=1}^m w_j y_j \quad \text{and} \quad \hat{M} = \sum_{j=1}^m w_j$$

and  $w_j$  is a sampling weight. The score variable for the mean estimator is

$$z_j(\bar{y}) = \frac{y_j - \bar{y}}{\hat{M}} = \frac{\hat{M}y_j - \hat{Y}}{\hat{M}^2}$$

## The standardized mean estimator

Let  $D_g$  denote the set of sampled observations that belong to the  $g$ th standard stratum and define  $I_{D_g}(j)$  to indicate if the  $j$ th observation is a member of the  $g$ th standard stratum; where  $g = 1, \dots, L_D$  and  $L_D$  is the number of standard strata. Also, let  $\pi_g$  denote the fraction of the population that belongs to the  $g$ th standard stratum, thus  $\pi_1 + \dots + \pi_{L_D} = 1$ .  $\pi_g$  is derived from the `stdweight()` option.

The estimator for the standardized mean is

$$\bar{y}^D = \sum_{g=1}^{L_D} \pi_g \frac{\hat{Y}_g}{\hat{M}_g}$$

where

$$\hat{Y}_g = \sum_{j=1}^m I_{D_g}(j) w_j y_j \quad \text{and} \quad \hat{M}_g = \sum_{j=1}^m I_{D_g}(j) w_j$$

The score variable for the standardized mean is

$$z_j(\bar{y}^D) = \sum_{g=1}^{L_D} \pi_g I_{D_g}(j) \frac{\hat{M}_g y_j - \hat{Y}_g}{\hat{M}_g^2}$$

## The poststratified mean estimator

Let  $P_k$  denote the set of sampled observations that belong to poststratum  $k$  and define  $I_{P_k}(j)$  to indicate if the  $j$ th observation is a member of poststratum  $k$ ; where  $k = 1, \dots, L_P$  and  $L_P$  is the number of poststrata. Also let  $M_k$  denote the population size for poststratum  $k$ .  $P_k$  and  $M_k$  are identified by specifying the `poststrata()` and `postweight()` options on `svyset`.

The estimator for the poststratified mean is

$$\bar{y}^P = \frac{\hat{Y}^P}{\widehat{M}^P} = \frac{\hat{Y}^P}{M}$$

where

$$\hat{Y}^P = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \hat{Y}_k = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \sum_{j=1}^m I_{P_k}(j) w_j y_j$$

and

$$\widehat{M}^P = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \widehat{M}_k = \sum_{k=1}^{L_P} M_k = M$$

The score variable for the poststratified mean is

$$z_j(\bar{y}^P) = \frac{z_j(\hat{Y}^P)}{M} = \frac{1}{M} \sum_{k=1}^{L_P} I_{P_k}(j) \frac{M_k}{\widehat{M}_k} \left( y_j - \frac{\hat{Y}_k}{\widehat{M}_k} \right)$$

## The standardized poststratified mean estimator

The estimator for the standardized poststratified mean is

$$\bar{y}^{DP} = \sum_{g=1}^{L_D} \pi_g \frac{\hat{Y}_g^P}{\widehat{M}_g^P}$$

where

$$\hat{Y}_g^P = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \hat{Y}_{g,k} = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \sum_{j=1}^m I_{D_g}(j) I_{P_k}(j) w_j y_j$$

and

$$\widehat{M}_g^P = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \widehat{M}_{g,k} = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \sum_{j=1}^m I_{D_g}(j) I_{P_k}(j) w_j$$

The score variable for the standardized poststratified mean is

$$z_j(\bar{y}^{DP}) = \sum_{g=1}^{L_D} \pi_g \frac{\widehat{M}_g^P z_j(\hat{Y}_g^P) - \hat{Y}_g^P z_j(\widehat{M}_g^P)}{(\widehat{M}_g^P)^2}$$

where

$$z_j(\hat{Y}_g^P) = \sum_{k=1}^{L_P} I_{P_k}(j) \frac{M_k}{\widehat{M}_k} \left\{ I_{D_g}(j) y_j - \frac{\hat{Y}_{g,k}}{\widehat{M}_k} \right\}$$

and

$$z_j(\widehat{M}_g^P) = \sum_{k=1}^{L_P} I_{P_k}(j) \frac{M_k}{\widehat{M}_k} \left\{ I_{D_g}(j) - \frac{\widehat{M}_{g,k}}{\widehat{M}_k} \right\}$$



## Subpopulation estimation

Let  $S$  denote the set of sampled observations that belong to the subpopulation of interest, and define  $I_S(j)$  to indicate if the  $j$ th observation falls within the subpopulation.

The estimator for the subpopulation mean is  $\bar{y}^S = \hat{Y}^S / \widehat{M}^S$ , where

$$\hat{Y}^S = \sum_{j=1}^m I_S(j) w_j y_j \quad \text{and} \quad \widehat{M}^S = \sum_{j=1}^m I_S(j) w_j$$

Its score variable is

$$z_j(\bar{y}^S) = I_S(j) \frac{y_j - \bar{y}^S}{\widehat{M}^S} = I_S(j) \frac{\widehat{M}^S y_j - \hat{Y}^S}{(\widehat{M}^S)^2}$$

The estimator for the standardized subpopulation mean is

$$\bar{y}^{DS} = \sum_{g=1}^{L_D} \pi_g \frac{\hat{Y}_g^S}{\widehat{M}_g^S}$$

where

$$\hat{Y}_g^S = \sum_{j=1}^m I_{D_g}(j) I_S(j) w_j y_j \quad \text{and} \quad \widehat{M}_g^S = \sum_{j=1}^m I_{D_g}(j) I_S(j) w_j$$

Its score variable is

$$z_j(\bar{y}^{DS}) = \sum_{g=1}^{L_D} \pi_g I_{D_g}(j) I_S(j) \frac{\widehat{M}_g^S y_j - \hat{Y}_g^S}{(\widehat{M}_g^S)^2}$$

The estimator for the poststratified subpopulation mean is

$$\bar{y}^{PS} = \frac{\hat{Y}^{PS}}{\widehat{M}^{PS}}$$

where

$$\hat{Y}^{PS} = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \hat{Y}_k^S = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \sum_{j=1}^m I_{P_k}(j) I_S(j) w_j y_j$$

and

$$\widehat{M}^{PS} = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \widehat{M}_k^S = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \sum_{j=1}^m I_{P_k}(j) I_S(j) w_j$$

Its score variable is

$$z_j(\bar{y}^{PS}) = \frac{\widehat{M}^{PS} z_j(\hat{Y}^{PS}) - \hat{Y}^{PS} z_j(\widehat{M}^{PS})}{(\widehat{M}^{PS})^2}$$

where

$$z_j(\hat{Y}^{PS}) = \sum_{k=1}^{L_P} I_{P_k}(j) \frac{M_k}{\widehat{M}_k} \left\{ I_S(j) y_j - \frac{\hat{Y}_k^S}{\widehat{M}_k} \right\}$$

and

$$z_j(\widehat{M}^{PS}) = \sum_{k=1}^{L_P} I_{P_k}(j) \frac{M_k}{\widehat{M}_k} \left\{ I_S(j) - \frac{\widehat{M}_k^S}{\widehat{M}_k} \right\}$$

The estimator for the standardized poststratified subpopulation mean is

$$\bar{y}^{DPS} = \sum_{g=1}^{L_D} \pi_g \frac{\widehat{Y}_g^{PS}}{\widehat{M}_g^{PS}}$$

where

$$\widehat{Y}_g^{PS} = \sum_{k=1}^{L_p} \frac{M_k}{\widehat{M}_k} \widehat{Y}_{g,k}^S = \sum_{k=1}^{L_p} \frac{M_k}{\widehat{M}_k} \sum_{j=1}^m I_{D_g}(j) I_{P_k}(j) I_S(j) w_j y_j$$

and

$$\widehat{M}_g^{PS} = \sum_{k=1}^{L_p} \frac{M_k}{\widehat{M}_k} \widehat{M}_{g,k}^S = \sum_{k=1}^{L_p} \frac{M_k}{\widehat{M}_k} \sum_{j=1}^m I_{D_g}(j) I_{P_k}(j) I_S(j) w_j$$

Its score variable is

$$z_j(\bar{y}^{DPS}) = \sum_{g=1}^{L_D} \pi_g \frac{\widehat{M}_g^{PS} z_j(\widehat{Y}_g^{PS}) - \widehat{Y}_g^{PS} z_j(\widehat{M}_g^{PS})}{(\widehat{M}_g^{PS})^2}$$

where

$$z_j(\widehat{Y}_g^{PS}) = \sum_{k=1}^{L_P} I_{P_k}(j) \frac{M_k}{\widehat{M}_k} \left\{ I_{D_g}(j) I_S(j) y_j - \frac{\widehat{Y}_{g,k}^S}{\widehat{M}_k} \right\}$$

and

$$z_j(\widehat{M}_g^{PS}) = \sum_{k=1}^{L_P} I_{P_k}(j) \frac{M_k}{\widehat{M}_k} \left\{ I_{D_g}(j) I_S(j) - \frac{\widehat{M}_{g,k}^S}{\widehat{M}_k} \right\}$$

## References

- Bakker, A. 2003. The early history of average values and implications for education. *Journal of Statistics Education* 11(1).
- Cochran, W. G. 1977. *Sampling Techniques*. 3rd ed. New York: Wiley.
- Stuart, A., and J. K. Ord. 1994. *Kendall's Advanced Theory of Statistics: Distribution Theory, Vol I*. 6th ed. London: Arnold.

## Also see

- [R] **mean postestimation** — Postestimation tools for mean
- [R] **ameans** — Arithmetic, geometric, and harmonic means
- [R] **proportion** — Estimate proportions
- [R] **ratio** — Estimate ratios
- [R] **summarize** — Summary statistics
- [R] **total** — Estimate totals
- [SVY] **direct standardization** — Direct standardization of means, proportions, and ratios
- [SVY] **poststratification** — Poststratification for survey data
- [SVY] **subpopulation estimation** — Subpopulation estimation for survey data
- [SVY] **svy estimation** — Estimation commands for survey data
- [SVY] **variance estimation** — Variance estimation for survey data
- [U] **20 Estimation and postestimation commands**

Description

The following postestimation commands are available for `mean`:

| command                  | description                                                                                         |
|--------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>       | VCE                                                                                                 |
| <code>estat (svy)</code> | postestimation statistics for survey data                                                           |
| <code>estimates</code>   | cataloging estimation results                                                                       |
| <code>lincom</code>      | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>nlcom</code>       | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>test</code>        | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>      | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Remarks

➤ Example 1

We have a dataset with monthly rates of returns on the Dow and NASDAQ stock indices. We can use `mean` to compute the average quarterly rates of return for the two indices separately;

```
. use http://www.stata-press.com/data/r11/rates
. mean dow nasdaq
Mean estimation Number of obs = 357
```

|        | Mean     | Std. Err. | [95% Conf. Interval] |          |
|--------|----------|-----------|----------------------|----------|
| dow    | .2489137 | 6.524386  | -12.58227            | 13.0801  |
| nasdaq | 10.78477 | 4.160821  | 2.601887             | 18.96765 |

If you chose just one of the indices for your portfolio, you either did rather well or rather poorly, depending on which one you picked. However, as we now show with the postestimation command `lincom`, if you diversified your portfolio, you would have earned a respectable 5.5% rate of return without having to guess which index would be the better performer.

```
. lincom .5*dow + .5*nasdaq
(1) .5 dow + .5 nasdaq = 0
```

|     | Coef.   | Std. Err. | t    | P> t  | [95% Conf. Interval] |          |
|-----|---------|-----------|------|-------|----------------------|----------|
| (1) | 5.51684 | 4.262673  | 1.29 | 0.196 | -2.866347            | 13.90003 |



## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [mean](#) — Estimate means

[SVY] [svy postestimation](#) — Postestimation tools for svy

# Title

meta — Meta-analysis

## Remarks

Stata does not have a meta-analysis command. Stata users, however, have developed an excellent suite of commands for performing meta-analysis, including commands for performing standard and cumulative meta-analysis, commands for producing forest plots and contour-enhanced funnel plots, and commands for nonparametric analysis of publication bias.

Many articles describing these commands have been published in the *Stata Technical Bulletin* and the *Stata Journal*. Recently, these articles were updated and published in a cohesive collection: *Meta-Analysis in Stata: An Updated Collection from the Stata Journal*.

In this collection, editor Jonathan Sterne discusses how these articles relate to each other and how they fit in the overall literature of meta-analysis. Sterne has organized the collection into four areas: classic meta-analysis; meta-regression; graphical and analytic tools for detecting bias; and recent advances such as meta-analysis for dose–response curves, diagnostic accuracy, multivariate analysis, and studies containing missing values.

All meta-analysis commands discussed in this collection may be downloaded by visiting <http://www.stata-press.com/books/mais.html>.

We highly recommend that Stata users interested in meta-analysis read this book.

Please also see the following FAQ on the Stata web site:

What meta-analysis features are available in Stata?  
<http://www.stata.com/support/faqs/stat/meta.html>

## References

- Borenstein, M., L. V. Hedges, J. P. T. Higgins, and H. R. Rothstein. 2009. *Introduction to Meta-Analysis*. Chichester, UK: Wiley.
- Egger, M., G. Davey Smith, and D. G. Altman, ed. 2001. *Systematic Reviews in Health Care: Meta-analysis in Context*. 2nd ed. London: BMJ Books.
- Sterne, J. A. C., ed. 2009. *Meta-Analysis in Stata: An Updated Collection from the Stata Journal*. College Station, TX: Stata Press.
- Sutton, A. J., K. R. Abrams, D. R. Jones, T. A. Sheldon, and F. Song. 2000. *Methods for Meta-Analysis in Medical Research*. New York: Wiley.

Syntax

```
mfp [, options] : regression_cmd [yvar1 [yvar2]] xvarlist [if] [in] [weight]
[, regression_cmd_options]
```

| options                              | description                                                                              |
|--------------------------------------|------------------------------------------------------------------------------------------|
| Model 2                              |                                                                                          |
| <u>s</u> equential                   | use the Royston and Altman model-selection algorithm; default uses closed-test procedure |
| <u>c</u> ycles(#)                    | maximum number of iteration cycles; default is cycles(5)                                 |
| <u>d</u> fdefault(#)                 | default maximum degrees of freedom; default is dfdefault(4)                              |
| <u>c</u> enter( <i>cent_list</i> )   | specification of centering for the independent variables                                 |
| <u>a</u> lpha( <i>alpha_list</i> )   | <i>p</i> -values for testing between FP models; default is alpha(0.05)                   |
| <u>d</u> f( <i>df_list</i> )         | degrees of freedom for each predictor                                                    |
| <u>p</u> owers( <i>numlist</i> )     | list of FP powers to use; default is powers(-2 -1(.5)1 2 3)                              |
| Adv. model                           |                                                                                          |
| <u>x</u> order(+ - n)                | order of entry into model-selection algorithm; default is xorder(+)                      |
| <u>s</u> elect( <i>select_list</i> ) | nominal <i>p</i> -values for selection on each predictor                                 |
| <u>x</u> powers( <i>xp_list</i> )    | FP powers for each predictor                                                             |
| <u>z</u> ero( <i>varlist</i> )       | treat nonpositive values of specified predictors as zero when FP transformed             |
| <u>c</u> atzero( <i>varlist</i> )    | add indicator variable for specified predictors                                          |
| Reporting                            |                                                                                          |
| <u>l</u> evel(#)                     | set confidence level; default is level(95)                                               |
| <u>a</u> ll                          | include out-of-sample observations in generated variables                                |

| regression_cmd_options | description                                          |
|------------------------|------------------------------------------------------|
| Adv. model             |                                                      |
| regression_cmd_options | options appropriate to the regression command in use |

All weight types supported by `regression_cmd` are allowed; see [U] 11.1.6 **weight**.  
See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.  
`fracgen` may be used to create new variables containing fractional polynomial powers. See [R] **fracpoly**.

where

*regression\_cmd* may be `clogit`, `glm`, `intreg`, `logistic`, `logit`, `mlogit`, `nbreg`, `ologit`, `oprobit`, `poisson`, `probit`, `qreg`, `regress`, `rreg`, `stcox`, `streg`, or `xtgee`.

*yvar1* is not allowed for `streg` and `stcox`. For these commands, you must first `stset` your data.

*yvar1* and *yvar2* must both be specified when *regression\_cmd* is `intreg`.

*xvarlist* has elements of type *varlist* and/or (*varlist*), e.g., `x1 x2 (x3 x4 x5)`

Elements enclosed in parentheses are tested jointly for inclusion in the model and are not eligible for fractional polynomial transformation.

## Menu

Statistics > Linear models and related > Fractional polynomials > Multivariable fractional polynomial models

## Description

**mfp** selects the multivariable fractional polynomial (MFP) model that best predicts the outcome variable from the right-hand-side variables in *xvarlist*.

## Options

### Model 2

**sequential** chooses the sequential fractional polynomial (FP) selection algorithm (see [Methods of FP model selection](#)).

**cycles(#)** sets the maximum number of iteration cycles permitted. `cycles(5)` is the default.

**dfdefault(#)** determines the default maximum degrees of freedom (df) for a predictor. The default is `dfdefault(4)` (second-degree FP).

**center(*cent\_list*)** defines the centering of the covariates *xvar1*, *xvar2*, ... of *xvarlist*. The default is `center(mean)`, except for binary covariates, where it is `center(#)`, with # being the lower of the two distinct values of the covariate. A typical item in *cent\_list* is *varlist*:{*mean*|*#*|*no*}. Items are separated by commas. The first item is special in that *varlist* is optional, and if it is omitted, the default is reset to the specified value (*mean*, *#*, or *no*). For example, `center(no, age:mean)` sets the default to *no* (i.e., no centering) and the centering of *age* to *mean*.

**alpha(*alpha\_list*)** sets the significance levels for testing between FP models of different degrees. The rules for *alpha\_list* are the same as those for *df\_list* in the `df()` option (see below). The default nominal *p*-value (significance level, selection level) is 0.05 for all variables.

Example: `alpha(0.01)` specifies that all variables have an FP selection level of 1%.

Example: `alpha(0.05, weight:0.1)` specifies that all variables except *weight* have an FP selection level of 5%; *weight* has a level of 10%.

**df(*df\_list*)** sets the df for each predictor. The df (not counting the regression constant, `_cons`) is twice the degree of the FP, so, for example, an *xvar* fit as a second-degree FP (FP2) has 4 df. The first item in *df\_list* may be either # or *varlist*:#. Subsequent items must be *varlist*:#. Items are separated by commas, and *varlist* is specified in the usual way for variables. With the first type of item, the df for all predictors is taken to be #. With the second type of item, all members of *varlist* (which must be a subset of *xvarlist*) have # df.



The default number of degrees of freedom for a predictor of type *varlist* specified in *xvarlist* but not in *df\_list* is assigned according to the number of distinct (unique) values of the predictor, as follows:

| # of distinct values | default df                       |
|----------------------|----------------------------------|
| 1                    | (invalid predictor)              |
| 2–3                  | 1                                |
| 4–5                  | <code>min(2, dfdefault())</code> |
| $\geq 6$             | <code>dfdefault()</code>         |

Example: `df(4)`

All variables have 4 df.

Example: `df(2, weight displ:4)`

`weight` and `displ` have 4 df; all other variables have 2 df.

Example: `df(weight displ:4, mpg:2)`

`weight` and `displ` have 4 df, `mpg` has 2 df; all other variables have default df.

`powers(numlist)` is the set of FP powers to be used. The default set is  $-2, -1, -0.5, 0, 0.5, 1, 2, 3$  (0 means log).

#### Adv. model

`xorder(+|-|n)` determines the order of entry of the covariates into the model-selection algorithm.

The default is `xorder(+)`, which enters them in decreasing order of significance in a multiple linear regression (most significant first). `xorder(-)` places them in reverse significance order, whereas `xorder(n)` respects the original order in *xvarlist*.

`select(select_list)` sets the nominal *p*-values (significance levels) for variable selection by backward elimination. A variable is dropped if its removal causes a nonsignificant increase in deviance. The rules for *select\_list* are the same as those for *df\_list* in the `df()` option (see above). Using the default selection level of 1 for all variables forces them all into the model. Setting the nominal *p*-value to be 1 for a given variable forces it into the model, leaving others to be selected or not. The nominal *p*-value for elements of *xvarlist* bound by parentheses is specified by including (*varlist*) in *select\_list*.

Example: `select(0.05)`

All variables have a nominal *p*-value of 5%.

Example: `select(0.05, weight:1)`

All variables except `weight` have a nominal *p*-value of 5%; `weight` is forced into the model.

Example: `select(a (b c):0.05)`

All variables except `a`, `b`, and `c` are forced into the model. `b` and `c` are tested jointly with 2 df at the 5% level, and `a` is tested singly at the 5% level.

`xpowers(xp_list)` sets the permitted FP powers for covariates individually. The rules for *xp\_list* are the same as for *df\_list* in the `df()` option. The default selection is the same as that for the `powers()` option.

Example: `xpowers(-1 0 1)`

All variables have powers  $-1, 0, 1$ .

Example: `xpowers(x5:-1 0 1)`

All variables except `x5` have default powers; `x5` has powers  $-1, 0, 1$ .

`zero(varlist)` treats negative and zero values of members of *varlist* as zero when FP transformations are applied. By default, such variables are subjected to a preliminary linear transformation to avoid negative and zero values (see [R] [fracpoly](#)). *varlist* must be part of *xvarlist*.

`catzero(varlist)` is a variation on `zero()`; see *Zeros and zero categories* below. *varlist* must be part of *xvarlist*.

*regression\_cmd\_options* may be any of the options appropriate to *regression\_cmd*.

#### Reporting

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see [U] [20.7 Specifying the width of confidence intervals](#).

`all` includes out-of-sample observations when generating the FP variables. By default, the generated FP variables contain missing values outside the estimation sample.

## Remarks

Remarks are presented under the following headings:

[Iteration report](#)

[Estimation algorithm](#)

[Methods of FP model selection](#)

[Zeros and zero categories](#)

For elements in *xvarlist* not enclosed in parentheses, `mfp` leaves variables in the data named `lxvar__1`, `lxvar__2`, ..., where *xvar* represents the first four letters of the name of *xvar1*, and so on, for *xvar2*, *xvar3*, etc. The new variables contain the best-fitting FP powers of *xvar1*, *xvar2*, ...

## Iteration report

By default, for each continuous predictor, *x*, `mfp` compares null, linear, and FP1 models for *x* with an FP2 model. The deviance for each of these nested submodels is given in the column labeled “Deviance”. The line labeled “Final” gives the deviance for the selected model and its powers. All the other predictors currently selected are included, with their transformations (if any). For models specified as having 1 df, the only choice is whether the variable enters the model.

## Estimation algorithm

The estimation algorithm in `mfp` processes the *xvars* in turn. Initially, `mfp` silently arranges *xvarlist* in order of increasing *p*-value (i.e., of decreasing statistical significance) for omitting each predictor from the model comprising *xvarlist*, with each term linear. The aim is to model relatively important variables before unimportant ones. This approach may help to reduce potential model-fitting difficulties caused by collinearity or, more generally, “concurvity” among the predictors. See the `xorder()` option above for details on how to change the ordering.

At the initial cycle, the best-fitting FP function for *xvar1* (the first of *xvarlist*) is determined, with all the other variables assumed to be linear. Either the default or the alternative procedure is used (see [Methods of FP model selection](#) below). The functional form (but not the estimated regression coefficients) for *xvar1* is kept, and the process is repeated for *xvar2*, *xvar3*, etc. The first iteration concludes when all the variables have been processed in this way. The next cycle is similar, except that the functional forms from the initial cycle are retained for all variables except the one currently being processed.

A variable whose functional form is prespecified to be linear (i.e., to have 1 df) is tested for exclusion within the above procedure when its nominal  $p$ -value (selection level) according to `select()` is less than 1; otherwise, it is included.

Updating of FP functions and candidate variables continues until the functions and variables included in the overall model do not change (convergence). Convergence is usually achieved within 1–4 cycles.

## Methods of FP model selection

`mfp` includes two algorithms for FP model selection, both of which combine backward elimination with the selection of an FP function. For each continuous variable in turn, they start from a most-complex permitted FP model and attempt to simplify the model by reducing the degree. The default algorithm resembles a closed-test procedure, a sequence of tests maintaining the overall type I error rate at a prespecified nominal level, such as 5%. All significance tests are approximate; therefore, the algorithm is not precisely a closed-test procedure (Royston and Sauerbrei 2008, chap. 6).

The closed-test algorithm for choosing an FP model with maximum permitted degree  $m = 2$  (i.e., an FP2 model with 4 df) for one continuous predictor,  $x$ , is as follows:

1. Inclusion: Test FP2 against the null model for  $x$  on 4 df at the significance level determined by `select()`. If  $x$  is significant, continue; otherwise, drop  $x$  from the model.
2. Nonlinearity: Test FP2 against a straight line in  $x$  on 3 df at the significance level determined by `alpha()`. If significant, continue; otherwise, stop, with the chosen model for  $x$  being a straight line.
3. Simplification: Test FP2 against FP1 on 2 df at the significance level determined by `alpha()`. If significant, the final model is FP2; otherwise, it is FP1.

The first step is omitted if  $x$  is to be retained in the model, that is, if its nominal  $p$ -value, according to the `select()` option, is 1.

An alternative algorithm is available with the `sequential` option, as originally suggested by Royston and Altman (1994):

1. Test FP2 against FP1 on 2 df at the `alpha()` significance level. If significant, the final model is FP2; otherwise, continue.
2. Test FP1 against a straight line on 1 df at the `alpha()` level. If significant, the final model is FP1; otherwise, continue.
3. Test a straight line against omitting  $x$  on 1 df at the `select()` level. If significant, the final model is a straight line; otherwise, drop  $x$ .

The final step is omitted if  $x$  is to be retained in the model, that is, if its nominal  $p$ -value, according to the `select()` option, is 1.

If  $x$  is uninfluential, the overall type I error rate of this procedure is about double that of the closed-test procedure, for which the rate is close to the nominal value. This inflated type I error rate confers increased apparent power to detect nonlinear relationships.

## Zeros and zero categories

The `zero()` option permits fitting an FP model to the positive values of a covariate, taking nonpositive values as zero. An application is the assessment of the effect of cigarette smoking as a risk factor in an epidemiological study. Nonsmokers may be qualitatively different from smokers, so the effect of smoking (regarded as a continuous variable) may not be continuous between one and

zero cigarettes. To allow for this, the risk may be modeled as constant for the nonsmokers and as an FP function of the number of cigarettes for the smokers:

```
. generate byte nonsmokr = cond(n_cigs==0, 1, 0) if n_cigs != .
. mfp, zero(n_cigs) df(4, nonsmokr:1): logit case n_cigs nonsmokr age
```

Omission of `zero(n_cigs)` would cause `n_cigs` to be transformed before analysis by the addition of a suitable constant, probably 1.

A closely related approach involves the `catzero()` option. The command

```
. mfp, catzero(n_cigs): logit case n_cigs age
```

would achieve a similar result to the previous command but with important differences. First, `mfp` would create the equivalent of the binary variable `nonsmokr` automatically and include it in the model. Second, the two smoking variables would be treated as one predictor in the model. With the `select()` option active, the two variables would be tested jointly for inclusion in the model. A modified version is described in [Royston and Sauerbrei \(2008, sec. 4.15\)](#).

► Example 1

We illustrate two of the analyses performed by [Sauerbrei and Royston \(1999\)](#). We use `brcancer.dta`, which contains prognostic factors data from the German Breast Cancer Study Group of patients with node-positive breast cancer. The response variable is recurrence-free survival time (`rectime`), and the censoring variable is `censrec`. There are 686 patients with 299 events. We use Cox regression to predict the log hazard of recurrence from prognostic factors, of which five are continuous (`x1`, `x3`, `x5`, `x6`, `x7`) and three are binary (`x2`, `x4a`, `x4b`). Hormonal therapy (`hormon`) is known to reduce recurrence rates and is forced into the model. We use `mfp` to build a model from the initial set of eight predictors by using the backfitting model-selection algorithm. We set the nominal  $p$ -value for variable and FP selection to 0.05 for all variables except `hormon`, for which it is set to 1:

```
. use http://www.stata-press.com/data/r11/brcancer
(German breast cancer data)
. stset rectime, fail(censrec)
(output omitted)
. mfp, alpha(.05) select(.05, hormon:1): stcox x1 x2 x3 x4a x4b x5 x6 x7 hormon,
> nohr
```

Deviance for model with all terms untransformed = 3471.637, 686 observations

| Variable                             | Model | (vs.) | Deviance | Dev diff. | P      | Powers | (vs.) |
|--------------------------------------|-------|-------|----------|-----------|--------|--------|-------|
| x5                                   | null  | FP2   | 3503.610 | 61.366    | 0.000* | .      | .5 3  |
|                                      | lin.  |       | 3471.637 | 29.393    | 0.000+ | 1      |       |
|                                      | FP1   |       | 3449.203 | 6.959     | 0.031+ | 0      |       |
|                                      | Final |       | 3442.244 |           |        | .5 3   |       |
| x6                                   | null  | FP2   | 3464.113 | 29.917    | 0.000* | .      | -2 .5 |
|                                      | lin.  |       | 3442.244 | 8.048     | 0.045+ | 1      |       |
|                                      | FP1   |       | 3435.550 | 1.354     | 0.508  | .5     |       |
|                                      | Final |       | 3435.550 |           |        | .5     |       |
| [hormon included with 1 df in model] |       |       |          |           |        |        |       |
| x4a                                  | null  | lin.  | 3440.749 | 5.199     | 0.023* | .      | 1     |
|                                      | Final |       | 3435.550 |           |        | 1      |       |
| x3                                   | null  | FP2   | 3436.832 | 3.560     | 0.469  | .      | -2 3  |
|                                      | Final |       | 3436.832 |           |        | .      |       |
| x2                                   | null  | lin.  | 3437.589 | 0.756     | 0.384  | .      | 1     |
|                                      | Final |       | 3437.589 |           |        | .      |       |
| x4b                                  | null  | lin.  | 3437.848 | 0.259     | 0.611  | .      | 1     |

|    |       |     |          |        |        |        |        |
|----|-------|-----|----------|--------|--------|--------|--------|
|    | Final |     | 3437.848 |        |        | .      |        |
| x1 | null  | FP2 | 3437.893 | 18.085 | 0.001* | .      | -2 -.5 |
|    | lin.  |     | 3437.848 | 18.040 | 0.000+ | 1      |        |
|    | FP1   |     | 3433.628 | 13.820 | 0.001+ | -2     |        |
|    | Final |     | 3419.808 |        |        | -2 -.5 |        |
| x7 | null  | FP2 | 3420.805 | 3.715  | 0.446  | .      | -.5 3  |
|    | Final |     | 3420.805 |        |        | .      |        |

End of Cycle 1: deviance = 3420.805

|    |       |     |          |        |        |       |       |
|----|-------|-----|----------|--------|--------|-------|-------|
| x5 | null  | FP2 | 3494.867 | 74.143 | 0.000* | .     | -2 -1 |
|    | lin.  |     | 3451.795 | 31.071 | 0.000+ | 1     |       |
|    | FP1   |     | 3428.023 | 7.299  | 0.026+ | 0     |       |
|    | Final |     | 3420.724 |        |        | -2 -1 |       |
| x6 | null  | FP2 | 3452.093 | 32.704 | 0.000* | .     | 0 0   |
|    | lin.  |     | 3427.703 | 8.313  | 0.040+ | 1     |       |
|    | FP1   |     | 3420.724 | 1.334  | 0.513  | .5    |       |
|    | Final |     | 3420.724 |        |        | .5    |       |

[hormon included with 1 df in model]

|     |       |      |          |        |        |        |        |
|-----|-------|------|----------|--------|--------|--------|--------|
| x4a | null  | lin. | 3425.310 | 4.586  | 0.032* | .      | 1      |
|     | Final |      | 3420.724 |        |        | 1      |        |
| x3  | null  | FP2  | 3420.724 | 5.305  | 0.257  | .      | -.5 0  |
|     | Final |      | 3420.724 |        |        | .      |        |
| x2  | null  | lin. | 3420.724 | 0.214  | 0.644  | .      | 1      |
|     | Final |      | 3420.724 |        |        | .      |        |
| x4b | null  | lin. | 3420.724 | 0.145  | 0.703  | .      | 1      |
|     | Final |      | 3420.724 |        |        | .      |        |
| x1  | null  | FP2  | 3440.057 | 19.333 | 0.001* | .      | -2 -.5 |
|     | lin.  |      | 3440.038 | 19.314 | 0.000+ | 1      |        |
|     | FP1   |      | 3436.949 | 16.225 | 0.000+ | -2     |        |
|     | Final |      | 3420.724 |        |        | -2 -.5 |        |
| x7  | null  | FP2  | 3420.724 | 2.152  | 0.708  | .      | -1 3   |
|     | Final |      | 3420.724 |        |        | .      |        |

Fractional polynomial fitting algorithm converged after 2 cycles.

Transformations of covariates:

```

-> gen double lx1__1 = X^-2-.0355294635 if e(sample)
-> gen double lx1__2 = X^-.5-.4341573547 if e(sample)
 (where: X = x1/10)
-> gen double lx5__1 = X^-2-3.983723313 if e(sample)
-> gen double lx5__2 = X^-1-1.99592668 if e(sample)
 (where: X = x5/10)
-> gen double lx6__1 = X^.5-.3331600619 if e(sample)
 (where: X = (x6+1)/1000)

```

Final multivariable fractional polynomial model for \_t

| Variable | Initial |        |        | Final  |    |        |
|----------|---------|--------|--------|--------|----|--------|
|          | df      | Select | Alpha  | Status | df | Powers |
| x1       | 4       | 0.0500 | 0.0500 | in     | 4  | -2 -.5 |
| x2       | 1       | 0.0500 | 0.0500 | out    | 0  |        |
| x3       | 4       | 0.0500 | 0.0500 | out    | 0  |        |
| x4a      | 1       | 0.0500 | 0.0500 | in     | 1  | 1      |
| x4b      | 1       | 0.0500 | 0.0500 | out    | 0  |        |
| x5       | 4       | 0.0500 | 0.0500 | in     | 4  | -2 -1  |
| x6       | 4       | 0.0500 | 0.0500 | in     | 2  | .5     |
| x7       | 4       | 0.0500 | 0.0500 | out    | 0  |        |
| hormon   | 1       | 1.0000 | 0.0500 | in     | 1  | 1      |

Cox regression -- Breslow method for ties  
Entry time \_t0  
  
Log likelihood = -1710.3619

Number of obs    =        686  
LR chi2(7)       =       155.62  
Prob > chi2      =       0.0000  
Pseudo R2        =       0.0435

| _t     | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|--------|-----------|-----------|-------|-------|----------------------|-----------|
| Ix1__1 | 44.73377  | 8.256682  | 5.42  | 0.000 | 28.55097             | 60.91657  |
| Ix1__2 | -17.92302 | 3.909611  | -4.58 | 0.000 | -25.58571            | -10.26032 |
| x4a    | .5006982  | .2496324  | 2.01  | 0.045 | .0114276             | .9899687  |
| Ix5__1 | .0387904  | .0076972  | 5.04  | 0.000 | .0237041             | .0538767  |
| Ix5__2 | -.5490645 | .0864255  | -6.35 | 0.000 | -.7184554            | -.3796736 |
| Ix6__1 | -1.806966 | .3506314  | -5.15 | 0.000 | -2.494191            | -1.119741 |
| hormon | -.4024169 | .1280843  | -3.14 | 0.002 | -.6534575            | -.1513763 |

Deviance: 3420.724.

Some explanation of the output from the model-selection algorithm is desirable. Consider the first few lines of output in the iteration log:

1. Deviance for model with all terms untransformed = 3471.637, 686 observations

| Variable | Model (vs.) | Deviance | Dev diff. | P      | Powers | (vs.) |
|----------|-------------|----------|-----------|--------|--------|-------|
| 2. x5    | null    FP2 | 3503.610 | 61.366    | 0.000* | .      | .5 3  |
| 3.       | lin.        | 3471.637 | 29.393    | 0.000+ | 1      |       |
| 4.       | FP1         | 3449.203 | 6.959     | 0.031+ | 0      |       |
| 5.       | Final       | 3442.244 |           |        | .5 3   |       |

Line 1 gives the deviance ( $-2 \times \log$  partial likelihood) for the Cox model with all terms linear, the place where the algorithm starts. The model is modified variable by variable in subsequent steps. The most significant linear term turns out to be x5, which is therefore processed first. Line 2 compares the best-fitting FP2 for x5 with a model omitting x5. The FP has powers (0.5, 3), and the test for inclusion of x5 is highly significant. The reported deviance of 3,503.610 is of the null model, not for the FP2 model. The deviance for the FP2 model may be calculated by subtracting the deviance difference (Dev diff.) from the reported deviance, giving  $3,503.610 - 61.366 = 3,442.244$ . Line 3 shows that the FP2 model is also a significantly better fit than a straight line (lin.) and line 4 that FP2 is also somewhat better than FP1 ( $p = 0.031$ ). Thus at this stage in the model-selection procedure, the final model for x5 (line 5) is FP2 with powers (0.5, 3). The overall model with an FP2 for x5 and all other terms linear has a deviance of 3,442.244.

After all the variables have been processed (cycle 1) and reprocessed (cycle 2) in this way, convergence is achieved because the functional forms (FP powers and variables included) after cycle 2 are the same as they were after cycle 1. The model finally chosen is Model II as given in tables 3 and 4 of [Sauerbrei and Royston \(1999\)](#). Because of scaling of variables, the regression coefficients reported there are different, but the model and its deviance are identical. The model includes x1 with powers  $(-2, -0.5)$ , x4a, x5 with powers  $(-2, -1)$ , and x6 with power 0.5. There is strong evidence of nonlinearity for x1 and for x5, the deviance differences for comparison with a straight-line model (FP2 vs lin.) being, respectively, 19.3 and 31.1 at convergence (cycle 2). Predictors x2, x3, x4b, and x7 are dropped, as may be seen from their status out in the table Final multivariable fractional polynomial model for \_t (the assumed *devar* when using *stcox*).

All predictors except x4a and hormon, which are binary, have been centered on the mean of the original variable. For example, the mean of x1 (age) is 53.05 years. The first FP-transformed variable for x1 is  $x1^{-2}$  and is created by the expression `gen double Ix1__1 = X^-2-.0355 if e(sample)`. The value 0.0355 is obtained from  $(53.05/10)^{-2}$ . The division by 10 is applied automatically to improve the scaling of the regression coefficient for Ix1\_\_1.

According to [Sauerbrei and Royston \(1999\)](#), medical knowledge dictates that the estimated risk function for `x5` (number of positive nodes), which was based on the above FP with powers  $(-2, -1)$ , should be monotonic, but it was not. They improved Model II by estimating a preliminary exponential transformation,  $x5e = \exp(-0.12 \cdot x5)$ , for `x5` and fitting a degree 1 FP for `x5e`, thus obtaining a monotonic risk function. The value of  $-0.12$  was estimated univariately using nonlinear Cox regression with the `ado-file` `boxtid` ([Royston and Ambler 1999b, 1999d](#)). To ensure a negative exponent, [Sauerbrei and Royston \(1999\)](#) restricted the powers for `x5e` to be positive. Their Model III may be fit by using the following command:

```
. mfp, alpha(.05) select(.05, hormon:1) df(x5e:2) xpowers(x5e:0.5 1 2 3):
> stcox x1 x2 x3 x4a x4b x5e x6 x7 hormon
```

Other than the customization for `x5e`, the command is the same as it was before. The resulting model is as reported in table 4 of [Sauerbrei and Royston \(1999\)](#):

```
. use http://www.stata-press.com/data/r11/brcancer, clear
(German breast cancer data)
. stset rectime, fail(censrec)
(output omitted)
. mfp, alpha(.05) select(.05, hormon:1) df(x5e:2) xpowers(x5e:0.5 1 2 3):
> stcox x1 x2 x3 x4a x4b x5e x6 x7 hormon, nohr
(output omitted)
```

Final multivariable fractional polynomial model for `_t`

| Variable | ——Initial—— |        |        | ——Final—— |    |        |
|----------|-------------|--------|--------|-----------|----|--------|
|          | df          | Select | Alpha  | Status    | df | Powers |
| x1       | 4           | 0.0500 | 0.0500 | in        | 4  | -2 -.5 |
| x2       | 1           | 0.0500 | 0.0500 | out       | 0  |        |
| x3       | 4           | 0.0500 | 0.0500 | out       | 0  |        |
| x4a      | 1           | 0.0500 | 0.0500 | in        | 1  | 1      |
| x4b      | 1           | 0.0500 | 0.0500 | out       | 0  |        |
| x5e      | 2           | 0.0500 | 0.0500 | in        | 1  | 1      |
| x6       | 4           | 0.0500 | 0.0500 | in        | 2  | .5     |
| x7       | 4           | 0.0500 | 0.0500 | out       | 0  |        |
| hormon   | 1           | 1.0000 | 0.0500 | in        | 1  | 1      |

Cox regression -- Breslow method for ties

|                             |                 |   |        |
|-----------------------------|-----------------|---|--------|
| Entry time <code>_t0</code> | Number of obs   | = | 686    |
|                             | LR $\chi^2(6)$  | = | 153.11 |
|                             | Prob > $\chi^2$ | = | 0.0000 |
| Log likelihood = -1711.6186 | Pseudo R2       | = | 0.0428 |

| <code>_t</code> | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-----------------|-----------|-----------|-------|-------|----------------------|-----------|
| Ix1__1          | 43.55382  | 8.253433  | 5.28  | 0.000 | 27.37738             | 59.73025  |
| Ix1__2          | -17.48136 | 3.911882  | -4.47 | 0.000 | -25.14851            | -9.814212 |
| x4a             | .5174351  | .2493739  | 2.07  | 0.038 | .0286713             | 1.006199  |
| Ix5e__1         | -1.981213 | .2268903  | -8.73 | 0.000 | -2.425909            | -1.536516 |
| Ix6__1          | -1.84008  | .3508432  | -5.24 | 0.000 | -2.52772             | -1.15244  |
| hormon          | -.3944998 | .128097   | -3.08 | 0.002 | -.6455654            | -.1434342 |

Deviance: 3423.237.

◀

(Continued on next page)

## Saved results

In addition to what *regression\_cmd* saves, *mfp* saves the following in `e()`:

### Scalars

|                        |                                                                             |
|------------------------|-----------------------------------------------------------------------------|
| <code>e(fp_nx)</code>  | number of predictors in <i>xvarlist</i>                                     |
| <code>e(fp_dev)</code> | deviance of final model fit                                                 |
| <code>e(fp_id#)</code> | initial degrees of freedom for the #th element of <i>xvarlist</i>           |
| <code>e(Fp_fd#)</code> | final degrees of freedom for the #th element of <i>xvarlist</i>             |
| <code>e(Fp_al#)</code> | FP selection level for the #th element of <i>xvarlist</i>                   |
| <code>e(Fp_se#)</code> | backward elimination selection level for the #th element of <i>xvarlist</i> |

### Macros

|                         |                                                           |
|-------------------------|-----------------------------------------------------------|
| <code>e(fp_cmd)</code>  | <code>fracpoly</code>                                     |
| <code>e(fp_cmd2)</code> | <code>mfp</code>                                          |
| <code>e(cmdline)</code> | command as typed                                          |
| <code>e(fp_fv1)</code>  | variables in final model                                  |
| <code>e(fp_depv)</code> | <code>yvar1</code> ( <code>yvar2</code> )                 |
| <code>e(fp_opts)</code> | estimation command options                                |
| <code>e(fp_x1)</code>   | first variable in <i>xvarlist</i>                         |
| <code>e(fp_x2)</code>   | second variable in <i>xvarlist</i>                        |
| <code>...</code>        |                                                           |
| <code>e(fp_xN)</code>   | last variable in <i>xvarlist</i> , $N=e(fp\_nx)$          |
| <code>e(fp_k1)</code>   | power for first variable in <i>xvarlist</i> (*)           |
| <code>e(fp_k2)</code>   | power for second variable in <i>xvarlist</i> (*)          |
| <code>...</code>        |                                                           |
| <code>e(fp_kN)</code>   | power for last var. in <i>xvarlist</i> (*), $N=e(fp\_nx)$ |

Note: (\*) contains ‘.’ if the variable is not selected in the final model.

## Methods and formulas

*mfp* is implemented as an ado-file.

## Acknowledgments

*mfp* is an update of *mfracpol* by [Royston and Ambler \(1998\)](#).

## References

- Ambler, G., and P. Royston. 2001. Fractional polynomial model selection procedures: Investigation of Type I error rate. *Journal of Statistical Computation and Simulation* 69: 89–108.
- Royston, P., and D. G. Altman. 1994. Regression using fractional polynomials of continuous covariates: Parsimonious parametric modelling (with discussion). *Applied Statistics* 43: 429–467.
- Royston, P., and G. Ambler. 1998. [sg81: Multivariable fractional polynomials](#). *Stata Technical Bulletin* 43: 24–32. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 123–132. College Station, TX: Stata Press.
- . 1999a. [sg112: Nonlinear regression models involving power or exponential functions of covariates](#). *Stata Technical Bulletin* 49: 25–30. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 173–179. College Station, TX: Stata Press.
- . 1999b. [sg81.1: Multivariable fractional polynomials: Update](#). *Stata Technical Bulletin* 49: 17–23. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 161–168. College Station, TX: Stata Press.



- . 1999c. [sg112.1: Nonlinear regression models involving power or exponential functions of covariates: Update](#). *Stata Technical Bulletin* 50: 26. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, p. 180. College Station, TX: Stata Press.
- . 1999d. [sg81.2: Multivariable fractional polynomials: Update](#). *Stata Technical Bulletin* 50: 25. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, p. 168. College Station, TX: Stata Press.
- Royston, P., and W. Sauerbrei. 2007. [Multivariable modeling with cubic regression splines: A principled approach](#). *Stata Journal* 7: 45–70.
- . 2008. [Multivariable Model-building: A Pragmatic Approach to Regression Analysis Based on Fractional Polynomials for Modelling Continuous Variables](#). Chichester, UK: Wiley.
- Sauerbrei, W., and P. Royston. 1999. Building multivariable prognostic and diagnostic models: Transformation of the predictors by using fractional polynomials. *Journal of the Royal Statistical Society, Series A* 162: 71–94.
- . 2002. Corrigendum: Building multivariable prognostic and diagnostic models: Transformation of the predictors by using fractional polynomials. *Journal of the Royal Statistical Society, Series A* 165: 399–400.

## Also see

- [R] [mfp postestimation](#) — Postestimation tools for mfp
- [R] [fracpoly](#) — Fractional polynomial regression
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are of special interest after `mfp`:

| command               | description                                                                        |
|-----------------------|------------------------------------------------------------------------------------|
| <code>fracplot</code> | plot data and fit from most recently fit fractional polynomial model               |
| <code>fracpred</code> | create variable containing prediction, deviance residuals, or SES of fitted values |

For `fracplot` and `fracpred`, see [\[R\] fracpoly postestimation](#).

The following standard postestimation commands are also available if available after `regression_cmd`:

| command                | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>     | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estimates</code> | cataloging estimation results                                                                       |
| <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>  | link test for model specification                                                                   |
| <code>lrtest</code>    | likelihood-ratio test                                                                               |
| <code>margins</code>   | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details.

Methods and formulas

All postestimation commands listed above are implemented as ado-files.

Also see

- [\[R\] mfp](#) — Multivariable fractional polynomial models
- [\[R\] fracpoly postestimation](#) — Postestimation tools for fracpoly
- [\[U\] 20 Estimation and postestimation commands](#)

Syntax

*Report counts of missing values*

```
misstable summarize [varlist] [if] [in] [, summarize_options]
```

*Report pattern of missing values*

```
misstable patterns [varlist] [if] [in] [, patterns_options]
```

*Present a tree view of the pattern of missing values*

```
misstable tree [varlist] [if] [in] [, tree_options]
```

*List the nesting rules that describe the missing-value pattern*

```
misstable nested [varlist] [if] [in] [, nested_options]
```

| <i>summarize_options</i> | description         |
|--------------------------|---------------------|
| <u>all</u>               | show all variables  |
| <u>showzeros</u>         | show zeros in table |

| <i>patterns_options</i> | description                                     |
|-------------------------|-------------------------------------------------|
| <u>asis</u>             | use variables in order given                    |
| <u>frequency</u>        | report frequencies instead of percentages       |
| <u>exok</u>             | treat .a, .b, ..., .z as nonmissing             |
| <u>replace</u>          | replace data in memory with dataset of patterns |
| <u>clear</u>            | okay to replace even if original unsaved        |
| <u>bypatterns</u>       | list by patterns rather than by frequency       |

| <i>tree_options</i> | description                               |
|---------------------|-------------------------------------------|
| <u>asis</u>         | use variables in order given              |
| <u>frequency</u>    | report frequencies instead of percentages |
| <u>exok</u>         | treat .a, .b, ..., .z as nonmissing       |

| <i>nested_options</i> | description                         |
|-----------------------|-------------------------------------|
| <u>exok</u>           | treat .a, .b, ..., .z as nonmissing |

In addition, programmer’s option `nopreserve` is allowed with all syntaxes; see [\[P\] nopreserve option](#).

## Menu

Statistics > Summaries, tables, and tests > Tables > Tabulate missing values

## Description

`misstable` makes tables that help you understand the pattern of missing values in your data.

## Options for `misstable summarize`

`all` specifies that the table should include all the variables specified or all the variables in the dataset. The default is to include only numeric variables that contain missing values.

`showzeros` specifies that zeros in the table should display as 0 rather than being omitted.

## Options for `misstable patterns`

`asis`, `frequency`, and `exok` – see [Common options](#) below.

`replace` specifies that the data in memory be replaced with a dataset corresponding to the table just displayed; see [misstable patterns](#) under *Remarks* below.

`clear` is for use with `replace`; it specifies that it is okay to change the data in memory even if they have not been saved to disk.

`bypatterns` specifies the table be ordered by pattern rather than by frequency. That is, `bypatterns` specifies that patterns containing one incomplete variable be listed first, followed by those for two incomplete variables, and so on. The default is to list the most frequent pattern first, followed by the next most frequent pattern, etc.

## Options for `misstable tree`

`asis`, `frequency`, and `exok` – see [Common options](#) below.

## Option for `misstable nested`

`exok` – see [Common options](#) below.

## Common options

`asis` specifies that the order of the variables in the table be the same as the order in which they are specified on the `misstable` command. The default is to order the variables by the number of missing values, and within that, by the amount of overlap of missing values.

`frequency` specifies that the table should report frequencies instead of percentages.

`exok` specifies that the extended missing values `.a`, `.b`, ..., `.z` should be treated as if they do not designate missing. Some users use extended missing values to designate values that are missing for a known and valid reason.

`nopreserve` is a programmer's option allowed with all `misstable` commands; see [\[P\] nopreserve option](#).

## Remarks

Remarks are presented under the following headings:

*misstable summarize*  
*misstable patterns*  
*misstable tree*  
*misstable nested*  
*Execution time of misstable nested*

In what follows, we will use data from a 125-observation, fictional, student-satisfaction survey:

```
. use http://www.stata-press.com/data/r11/studentsurvey
(Student Survey)
. summarize
```

| Variable  | Obs | Mean     | Std. Dev. | Min | Max |
|-----------|-----|----------|-----------|-----|-----|
| m1        | 125 | 2.456    | .8376619  | 1   | 4   |
| m2        | 125 | 2.472    | .8089818  | 1   | 4   |
| age       | 122 | 18.97541 | .8763477  | 17  | 21  |
| female    | 122 | .5245902 | .5014543  | 0   | 1   |
| dept      | 116 | 2.491379 | 1.226488  | 1   | 4   |
| offcampus | 125 | .36      | .4819316  | 0   | 1   |
| comment   | 0   |          |           |     |     |

The m1 and m2 variables record the student's satisfaction with teaching and with academics. comment is a string variable recording any comments the student might have had.

## misstable summarize

### ► Example 1

misstable summarize reports counts of missing values:

```
. misstable summarize
```

| Variable | Obs=. | Obs>. | Obs<. | Obs<.         |     |     |
|----------|-------|-------|-------|---------------|-----|-----|
|          |       |       |       | Unique values | Min | Max |
| age      | 3     |       | 122   | 5             | 17  | 21  |
| female   | 3     |       | 122   | 2             | 0   | 1   |
| dept     | 9     |       | 116   | 4             | 1   | 4   |

Stata provides 27 different missing values, namely, ., .a, .b, ..., .z. The first of those, ., is often called system missing. The remaining missing values are called extended missings. The nonmissing and missing values are ordered *nonmissing* < . < .a < .b < ... < .z. Thus reported in the column “Obs=.” are counts of system missing values; in the column “Obs>.”, extended missing values; and in the column “Obs<.”, nonmissing values.

The rightmost portion of the table is included to remind you how the variables are encoded.

Our data contain seven variables and yet misstable reported only three of them. The omitted variables contain no missing values or are string variables. Even if we specified the varlist explicitly, those variables would not appear in the table unless we specified the all option.

misstable patterns

➤ Example 2

misstable patterns reports the pattern of missing values:

```
. misstable patterns
Missing-value patterns
```

| Percent | Pattern |   |   |
|---------|---------|---|---|
|         | 1       | 2 | 3 |
| 93%     | 1       | 1 | 1 |
| 5       | 1       | 1 | 0 |
| 2       | 0       | 0 | 0 |
| 100%    |         |   |   |

```
Variables are (1) age (2) female (3) dept
```

There are three patterns in these data: (1,1,1), (1,1,0), and (0,0,0). By default, the rows of the table are ordered by frequency. In larger tables that have more patterns, it is sometimes useful to order the rows by pattern. We could have obtained that by typing `mi misstable patterns, bypatterns`.

In a pattern, 1 indicates that all values of the variable are nonmissing and 0 indicates that all values are missing. Thus pattern (1,1,1) means no missing values, and 93% of our data have that pattern. There are two patterns in which variables are missing, (1,1,0) and (0,0,0). Pattern (1,1,0) means that `age` is nonmissing, `female` is nonmissing, and `dept` is missing. The order of the variables in the patterns appears in the key at the bottom of the table. Five percent of the observations have pattern (1,1,0). The remaining 2% have pattern (0,0,0), meaning that all three variables contain missing.

As with `misstable summarize`, only numeric variables that contain missing are listed, so had we typed `misstable patterns comments age female offcampus dept`, we still would have obtained the same table. Variables that are automatically omitted contain no missing values or are string variables.

The variables in the table are ordered from lowest to highest frequency of missing values, although you cannot see that from the information presented in the table. The variables are ordered this way even if you explicitly specify the varlist with a different ordering. Typing `misstable patterns dept female age` would produce the same table as above. Specify the `asis` option if you want the variables in the order in which you specify them.

You can obtain a dataset of the patterns by specifying the `replace` option:

```
. misstable patterns, replace
Missing-value patterns
```

| Percent | Pattern |   |   |
|---------|---------|---|---|
|         | 1       | 2 | 3 |
| 93%     | 1       | 1 | 1 |
| 5       | 1       | 1 | 0 |
| 2       | 0       | 0 | 0 |
| 100%    |         |   |   |

```
Variables are (1) age (2) female (3) dept
(summary data now in memory)
```

```
. list
```

|    | _freq | age | female | dept |
|----|-------|-----|--------|------|
| 1. | 3     | 0   | 0      | 0    |
| 2. | 6     | 1   | 1      | 0    |
| 3. | 116   | 1   | 1      | 1    |

The differences between the dataset and the printed table are that 1) the dataset always records frequency and 2) the rows are reversed.

◀

## misstable tree

### ► Example 3

`misstable tree` presents a tree view of the pattern of missing values:

```
. use http://www.stata-press.com/data/r11/studentsurvey, clear
(Student Survey)
```

```
. misstable tree, frequency
```

| Nested pattern of missing values |     |        |  |
|----------------------------------|-----|--------|--|
| dept                             | age | female |  |
| 9                                | 3   | 3      |  |
|                                  |     | 0      |  |
|                                  | 6   | 0      |  |
| 116                              | 0   | 6      |  |
|                                  |     | 0      |  |
|                                  | 116 | 0      |  |
|                                  |     | 116    |  |

(number missing listed first)

In this example, we specified the `frequency` option to see the table in frequency rather than percentage terms. In the table, each column sums to the total number of observations in the data, 125. Variables are ordered from those with the most missing values to those with the least. Start with the first column. The `dept` variable is missing in 9 observations and, farther down, the table reports that it is not missing in 116 observations.

Go back to the first row and read across, but only to the second column. The `dept` variable is missing in 9 observations. Within those 9, `age` is missing in 3 of them and is not missing in the remaining 6. Reading down the second column, within the 116 observations that `dept` is not missing, `age` is missing in 0 and not missing in 116.

Reading straight across the first row again, `dept` is missing in 9 observations, and within the 9, `age` is missing in 3, and within the 3, `female` is also missing in 3. Skipping down just a little, within the 6 observations for which `dept` is missing and `age` is not missing, `female` is not missing, too.

◀

## misstable nested

### ► Example 4

`misstable nested` lists the nesting rules that describe the missing-value pattern,

```
. misstable nested
 1. female(3) <-> age(3) -> dept(9)
```

This line says that in observations in which `female` is missing, so is `age` missing, and vice versa, and in observations in which `age` (or `female`) is missing, so is `dept`. The numbers in parentheses are counts of the missing values. The `female` variable happens to be missing in 3 observations, and the same is true for `age`; the `dept` variable is missing in 9 observations. Thus `dept` is missing in the 3 observations for which `age` and `female` are missing, and in 6 more observations, too.

In these data, it turns out that the missing-value pattern can be summarized in one statement. In a larger dataset, you might see something like this:

```
. misstable nested
 1. female(50) <-> age(50) -> dept(120)
 2. female(50) -> m1(58)
 3. offcampus(11)
```

`misstable nested` accounts for every missing value. In the above, in addition to `female <-> age -> dept`, we have that `female -> m1`, and we have `offcampus`, the last all by itself. The last line says that the 11 missing values in `offcampus` are not themselves nested in the missing value of any other variable, nor do they imply the missing values in another variable. In some datasets, all the statements will be of this last form.

In our data, however, we have one statement:

```
. misstable nested
 1. female(3) <-> age(3) -> dept(9)
```

When the missing-value pattern can be summarized in one `misstable nested` statement, the pattern of missing values in the data is said to be monotone.



## Execution time of misstable nested

The execution time of `misstable nested` is affected little by the number of observations but can grow quickly with the number of variables, depending on the fraction of missing values within variable. The execution time of the example above, which has 3 variables containing missing, is instant. In worst-case scenarios, with 500 variables, the time might be 25 seconds; with 1,000 variables, the execution time might be closer to an hour.

In situations where `misstable nested` takes a long time to complete, it will produce thousands of rules that will defy interpretation. A 523-variable dataset we have seen ran in 20 seconds and produced 8,040 rules. Although we spotted a few rules in the output that did not surprise us, such as the year of the date being missing implied that the month and the day were also missing, mostly the output was not helpful.

If you have such a dataset, we recommend you run `misstable` on groups of variables that you have reason to believe the pattern of missing values might be related.



## Saved results

`misstable summarize` saves the following values of the last variable summarized in `r()`:

### Scalars

`r(N_eq_dot)` number of observations containing .  
`r(N_gt_dot)` number of observations containing .a, .b, ..., .z  
`r(N_lt_dot)` number of observations containing nonmissing  
`r(K_uniq)` number of unique, nonmissing values  
`r(min)` variable's minimum value  
`r(max)` variable's maximum value

### Macros

`r(vartype)` numeric, string, or none

`r(K_uniq)` contains . if the number of unique, nonmissing values is greater than 500. `r(vartype)` contains none if no variables are summarized, and in that case, the value of the scalars are all set to missing (.). Programmers intending to access results after `misstable summarize` should specify the all option.

`misstable patterns` saves the following in `r()`:

### Scalars

`r(N_complete)` number of complete observations  
`r(N_incomplete)` number of incomplete observations  
`r(K)` number of patterns

### Macros

`r(vars)` variables used in order presented

`r(N_complete)` and `r(N_incomplete)` are defined with respect to the variables specified if variables were specified and otherwise, defined with respect to all the numeric variables in the dataset. `r(N_complete)` is the number of observations that contain no missing values.

`misstable tree` saves the following in `r()`:

### Macros

`r(vars)` variables used in order presented

`misstable nested` saves the following in `r()`:

### Scalars

`r(K)` number of statements

### Macros

`r(stmt1)` first statement  
`r(stmt2)` second statement  
. .  
. .  
`r(stmt'r(K)')` last statement  
`r(stmt1wc)` `r(stmt1)` with missing-value counts  
`r(vars)` variables considered

A statement is encoded “*varname*”, “*varname op varname*”, or “*varname op varname op varname*”, and so on; *op* is either “->” or “<->”.

## Methods and formulas

`misstable` is implemented as an ado-file.

## Also see

[MI] **mi misstable** — Tabulate pattern of missing values

[R] **summarize** — Summary statistics

[R] **tabulate oneway** — One-way tables of frequencies

[R] **tabulate twoway** — Two-way tables of frequencies

# Title

mkspline — Linear and restricted cubic spline construction

## Syntax

*Linear spline with knots at specified points*

```
mkspline newvar1 #1 [newvar2 #2 [...]] newvark = oldvar [if] [in] [, mmarginal
 displayknots]
```

*Linear spline with knots equally spaced or at percentiles of data*

```
mkspline stubname # = oldvar [if] [in] [weight] [, mmarginal pctile
 displayknots]
```

*Restricted cubic spline*

```
mkspline stubname = oldvar [if] [in] [weight], cubic [nknots(#) knots(numlist)
 displayknots]
```

*fweights* are allowed with the second and third syntax; see [U] 11.1.6 *weight*.

## Menu

Data > Create or change data > Other variable-creation commands > Linear and cubic spline construction

## Description

**mkspline** creates variables containing a linear spline or a restricted cubic spline of *oldvar*.

In the first syntax, **mkspline** creates *newvar*<sub>1</sub>, ..., *newvar*<sub>k</sub> containing a linear spline of *oldvar* with knots at the specified #<sub>1</sub>, ..., #<sub>k-1</sub>.

In the second syntax, **mkspline** creates # variables named *stubname*<sub>1</sub>, ..., *stubname*<sub>#</sub> containing a linear spline of *oldvar*. The knots are equally spaced over the range of *oldvar* or are placed at the percentiles of *oldvar*.

In the third syntax, **mkspline** creates variables containing a restricted cubic spline of *oldvar*. This is also known as a natural spline. The location and spacing of the knots is determined by the specification of the `nknots()` and `knots()` options.

## Options

### Options

**marginal** is allowed with the first or second syntax. It specifies that the new variables be constructed so that, when used in estimation, the coefficients represent the change in the slope from the preceding interval. The default is to construct the variables so that, when used in estimation, the coefficients measure the slopes for the interval.

`displayknots` displays the values of the knots that were used in creating the linear or restricted cubic spline.

`pctile` is allowed only with the second syntax. It specifies that the knots be placed at percentiles of the data rather than being equally spaced based on the range.

`nknots(#)` is allowed only with the third syntax. It specifies the number of knots that are to be used for a restricted cubic spline. This number must be between 3 and 7 unless the knot locations are specified using `knots()`. The default number of knots is 5.

`knots(numlist)` is allowed only with the third syntax. It specifies the exact location of the knots to be used for a restricted cubic spline. The values of these knots must be given in increasing order. When this option is omitted, the default knot values are based on Harrell's recommended percentiles with the additional restriction that the smallest knot may not be less than the fifth-smallest value of *oldvar* and the largest knot may not be greater than the fifth-largest value of *oldvar*. If both `nknots()` and `knots()` are given, they must specify the same number of knots.

## Remarks

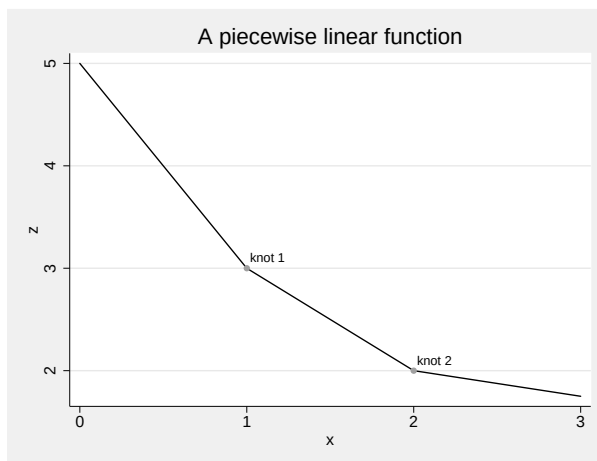
Remarks are presented under the following headings:

[Linear splines](#)

[Restricted cubic splines](#)

## Linear splines

Linear splines allow estimating the relationship between  $y$  and  $x$  as a piecewise linear function, which is a function composed of linear segments—straight lines. One linear segment represents the function for values of  $x$  below  $x_0$ , another linear segment handles values between  $x_0$  and  $x_1$ , and so on. The linear segments are arranged so that they join at  $x_0, x_1, \dots$ , which are called the knots. An example of a piecewise linear function is shown below.



## ► Example 1

We wish to fit a model of log income on education and age by using a piecewise linear function for age:

$$\text{lninc} = b_0 + b_1 \text{educ} + f(\text{age}) + u$$

The knots are to be placed at 10-year intervals: 20, 30, 40, 50, and 60.

```
. use http://www.stata-press.com/data/r11/mksp1
. mkspline age1 20 age2 30 age3 40 age4 50 age5 60 age6 = age, marginal
. regress lninc educ age1-age6
(output omitted)
```

Because we specified the `marginal` option, we could test whether the age effect is the same in the 30–40 and 40–50 intervals by asking whether the `age4` coefficient is zero. With the `marginal` option, coefficients measure the change in slope from the preceding group. Specifying `marginal` changes only the interpretation of the coefficients; the same model is fit in either case. Without the `marginal` option, the interpretation of the coefficients would have been

$$\frac{dy}{d\text{age}} = \begin{cases} a_1 & \text{if age} < 20 \\ a_2 & \text{if } 20 \leq \text{age} < 30 \\ a_3 & \text{if } 30 \leq \text{age} < 40 \\ a_4 & \text{if } 40 \leq \text{age} < 50 \\ a_5 & \text{if } 50 \leq \text{age} < 60 \\ a_6 & \text{otherwise} \end{cases}$$

With the `marginal` option, the interpretation is

$$\frac{dy}{d\text{age}} = \begin{cases} a_1 & \text{if age} < 20 \\ a_1 + a_2 & \text{if } 20 \leq \text{age} < 30 \\ a_1 + a_2 + a_3 & \text{if } 30 \leq \text{age} < 40 \\ a_1 + a_2 + a_3 + a_4 & \text{if } 40 \leq \text{age} < 50 \\ a_1 + a_2 + a_3 + a_4 + a_5 & \text{if } 50 \leq \text{age} < 60 \\ a_1 + a_2 + a_3 + a_4 + a_5 + a_6 & \text{otherwise} \end{cases}$$

◀

## ► Example 2

Say that we have a binary outcome variable called `outcome`. We are beginning an analysis and wish to parameterize the effect of dosage on outcome. We wish to divide the data into five equal-width groups of dosage for the piecewise linear function.

```
. use http://www.stata-press.com/data/r11/mksp2
. mkspline dose 5 = dosage, displayknots
```

|        | knot1 | knot2 | knot3 | knot4 |
|--------|-------|-------|-------|-------|
| dosage | 20    | 40    | 60    | 80    |

```
. logistic outcome dose1-dose5
(output omitted)
```

`mkspline dose 5 = dosage` creates five variables—`dose1`, `dose2`, . . . , `dose5`—equally spacing the knots over the range of `dosage`. Because `dosage` varied between 0 and 100, the `mkspline` command above has the same effect as typing

```
. mkspline dose1 20 dose2 40 dose3 60 dose4 80 dose5 = dosage
```

The `pctile` option sets the knots to divide the data into five equal sample-size groups rather than five equal-width ranges. Typing

```
. mkspline pctdose 5 = dosage, pctile displayknots
```

|        | knot1 | knot2 | knot3 | knot4 |
|--------|-------|-------|-------|-------|
| dosage | 16    | 36.4  | 55.6  | 82    |

places the knots at the 20th, 40th, 60th, and 80th percentiles of the data.



Restricted cubic splines

A linear spline can be used to fit many functions well. However, a restricted cubic spline may be a better choice than a linear spline when working with a very curved function. When using a restricted cubic spline, one obtains a continuous smooth function that is linear before the first knot, a piecewise cubic polynomial between adjacent knots, and linear again after the last knot.

➤ Example 3

Returning to the data from example 1, we may feel that a curved function is a better fit. First, we will use the `knots()` option to specify the five knots that we used previously.

```
. use http://www.stata-press.com/data/r11/mksp1, clear
. mkspline agesp = age, cubic knots(20 30 40 50 60)
. regress lninc educ agesp*
(output omitted)
```

Harrell Jr. (2001, 23) recommends placing knots at equally spaced percentiles of the original variable’s marginal distribution. If we do not specify the `knots()` option, variables will be created containing a restricted cubic spline with five knots determined by Harrell’s default percentiles.

```
. use http://www.stata-press.com/data/r11/mksp1, clear
. mkspline agesp = age, cubic displayknots
. regress lninc educ agesp*
(output omitted)
```



Methods and formulas

`mkspline` is implemented as an ado-file.

Methods and formulas are presented under the following headings:

- Linear splines
- Restricted cubic splines

## Linear splines

Let  $V_i$ ,  $i = 1, \dots, n$ , be the variables to be created;  $k_i$ ,  $i = 1, \dots, n - 1$ , be the corresponding knots; and  $\mathcal{V}$  be the original variable (the command is `mkspline V1 k1 V2 k2 ... Vn =  $\mathcal{V}$` ). Then

$$V_1 = \min(\mathcal{V}, k_1)$$

$$V_i = \max\left\{\min(\mathcal{V}, k_i), k_{i-1}\right\} - k_{i-1} \quad i = 2, \dots, n$$

If the `marginal` option is specified, the definitions are

$$V_1 = \mathcal{V}$$

$$V_i = \max(0, \mathcal{V} - k_{i-1}) \quad i = 2, \dots, n$$

In the second syntax, `mkspline stubname # =  $\mathcal{V}$` , so let  $m$  and  $M$  be the minimum and maximum of  $\mathcal{V}$ . Without the `ptile` option, knots are set at  $m + (M - m)(i/n)$  for  $i = 1, \dots, n - 1$ . If `ptile` is specified, knots are set at the 100( $i/n$ ) percentiles, for  $i = 1, \dots, n - 1$ . Percentiles are calculated by `centile`; see [R] [centile](#).

## Restricted cubic splines

Let  $k_i$ ,  $i = 1, \dots, n$ , be the knot values;  $V_i$ ,  $i = 1, \dots, n - 1$ , be the variables to be created; and  $\mathcal{V}$  be the original variable. Then

$$V_1 = \mathcal{V}$$

$$V_{i+1} = \frac{(\mathcal{V} - k_i)_+^3 - (k_n - k_{n-1})^{-1}\{(\mathcal{V} - k_{n-1})_+^3(k_n - k_i) - (\mathcal{V} - k_n)_+^3(k_{n-1} - k_i)\}}{(k_n - k_1)^2}$$

$$i = 1, \dots, n - 2$$

where

$$(u)_+ = \begin{cases} u, & \text{if } u > 0 \\ 0, & \text{if } u \leq 0 \end{cases}$$

Without the `knots()` option, the locations of the knots are determined by the percentiles recommended in [Harrell Jr. \(2001, 23\)](#). These percentiles are based on the chosen number of knots as follows:

| No.<br>of knots | Percentiles |       |       |      |       |       |      |
|-----------------|-------------|-------|-------|------|-------|-------|------|
| 3               | 10          | 50    | 90    |      |       |       |      |
| 4               | 5           | 35    | 65    | 95   |       |       |      |
| 5               | 5           | 27.5  | 50    | 72.5 | 95    |       |      |
| 6               | 5           | 23    | 41    | 59   | 77    | 95    |      |
| 7               | 2.5         | 18.33 | 34.17 | 50   | 65.83 | 81.67 | 97.5 |

Harrell provides default percentiles when the number of knots is between 3 and 7. When using a number of knots outside this range, the location of the knots must be specified in `knots()`.

## Acknowledgment

The restricted cubic spline portion of `mkspline` is based on the `rc_spline` command by William Dupont.

## References

- Gould, W. W. 1993. [sg19: Linear splines and piecewise linear functions](#). *Stata Technical Bulletin* 15: 13–17. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 98–104. College Station, TX: Stata Press.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- Harrell Jr., F. E. 2001. *Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis*. New York: Springer.
- Newson, R. 2000. [sg151: B-splines and splines parameterized by their values at reference points on the x-axis](#). *Stata Technical Bulletin* 57: 20–27. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 221–230. College Station, TX: Stata Press.
- Panis, C. 1994. [sg24: The piecewise linear spline transformation](#). *Stata Technical Bulletin* 18: 27–29. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 146–149. College Station, TX: Stata Press.

## Also see

[\[R\] fracpoly](#) — Fractional polynomial regression



The documentation for [R] **ml** has been updated. To see the latest PDF of [R] **ml**, click [here](#).

## Syntax

*ml model in interactive mode*

```
ml model method proname eq [eq ...] [if] [in] [weight]
 [, model_options svy diparm_options]

ml model method funcname() eq [eq ...] [if] [in] [weight]
 [, model_options svy diparm_options]
```

*ml model in noninteractive mode*

```
ml model method proname eq [eq ...] [if] [in] [weight], maximize
 [model_options svy diparm_options noninteractive_options]

ml model method funcname() eq [eq ...] [if] [in] [weight], maximize
 [model_options svy diparm_options noninteractive_options]
```

Noninteractive mode is invoked by specifying the `maximize` option. Use `maximize` when `ml` will be used as a subroutine of another ado-file or program and you want to carry forth the problem, from definition to posting of results, in one command.

```
ml clear

ml query

ml check

ml search [[/] eqname [:] #lb #ub] [...] [, search_options]

ml plot [eqname :] name [# [# [#]]] [, saving(filename [, replace])]

ml init { [eqname :] name = # [/ eqname = # } [...]
ml init # [# ...], copy
ml init matname [, copy skip]

ml report

ml trace { on | off }

ml count [clear | on | off]

ml maximize [, ml_maximize_options display_options eform_option]
```

```
ml graph [#] [, saving(filename[, replace])]

ml display [, display_options eform_option]

ml footnote

ml score newvar [if] [in] [, equation(eqname) forcescores missing]

ml score newvarlist [if] [in] [, missing]

ml score [type] stub* [if] [in] [, missing]
```

where *method* is one of

|    |         |         |         |
|----|---------|---------|---------|
| lf | d0      |         | v0      |
|    | d1      | e1      | v1      |
|    | d1debug | e1debug | v1debug |
|    | d2      | e2      | v2      |
|    | d2debug | e2debug | v2debug |

*eq* is the equation to be estimated, enclosed in parentheses, and optionally with a name to be given to the equation, preceded by a colon,

```
([eqname:] [varlisty =] [varlistx] [, eq_options])
```

or *eq* is the name of a parameter, such as sigma, with a slash in front

```
/eqname which is equivalent to (eqname:)
```

and *diparm\_options* is one or more `diparm(diparm_args)` options where *diparm\_args* is either `__sep__` or anything accepted by the `_diparm` command; see help `_diparm`.

| eq_options                                 | description                                                                   |
|--------------------------------------------|-------------------------------------------------------------------------------|
| <code>noconstant</code>                    | do not include an intercept in the equation                                   |
| <code>offset(varname<sub>o</sub>)</code>   | include <i>varname<sub>o</sub></i> in model with coefficient constrained to 1 |
| <code>exposure(varname<sub>e</sub>)</code> | include $\ln(\text{varname}_e)$ in model with coefficient constrained to 1    |

| model_options                                  | description                                                                                                                          |
|------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <code>vce(vcetype)</code>                      | <i>vcetype</i> may be <code>robust</code> , <code>cluster clustvar</code> , <code>oim</code> , or <code>opg</code>                   |
| <code>constraints(numlist)</code>              | constraints by number to be applied                                                                                                  |
| <code>constraints(matname)</code>              | matrix that contains the constraints to be applied                                                                                   |
| <code>nocnsnotes</code>                        | do not display notes when constraints are dropped                                                                                    |
| <code>title(string)</code>                     | place a title on the estimation output                                                                                               |
| <code>nopreserve</code>                        | do not preserve the estimation subsample in memory                                                                                   |
| <code>collinear</code>                         | keep collinear variables within equations                                                                                            |
| <code>missing</code>                           | keep observations containing variables with missing values                                                                           |
| <code>lfo(#<sub>k</sub> #<sub>ll</sub>)</code> | number of parameters and log-likelihood value of the constant-only model                                                             |
| <code>continue</code>                          | specifies that a model has been fit and sets the initial values <b>b</b> <sub>0</sub> for the model to be fit based on those results |

---

|                               |                                                                                                           |
|-------------------------------|-----------------------------------------------------------------------------------------------------------|
| <code>waldtest(#)</code>      | perform a Wald test; see <i>Options for use with ml model in interactive or noninteractive mode</i> below |
| <code>obs(#)</code>           | number of observations                                                                                    |
| <code>noscvvars</code>        | do not create and pass score variables to likelihood-evaluator program; seldom used                       |
| <code>crittype(string)</code> | describe the criterion optimized by ml                                                                    |
| <code>subpop(varname)</code>  | compute estimates for the single subpopulation                                                            |
| <code>nosvyadjust</code>      | carry out Wald test as $W/k \sim F(k, d)$                                                                 |
| <code>technique(nr)</code>    | Stata's modified Newton–Raphson (NR) algorithm                                                            |
| <code>technique(bhhh)</code>  | Berndt–Hall–Hall–Hausman (BHHH) algorithm                                                                 |
| <code>technique(dfpr)</code>  | Davidon–Fletcher–Powell (DFP) algorithm                                                                   |
| <code>technique(bfgs)</code>  | Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm                                                         |

---

| <i>noninteractive_options</i>         | description                                                            |
|---------------------------------------|------------------------------------------------------------------------|
| <code>init(ml_init_args)</code>       | set the initial values <b>b</b> <sub>0</sub>                           |
| <code>search(on)</code>               | equivalent to <code>ml search, repeat(0)</code> ; the default          |
| <code>search(norescale)</code>        | equivalent to <code>ml search, repeat(0) norescale</code>              |
| <code>search(quietly)</code>          | same as <code>search(on)</code> , except that output is suppressed     |
| <code>search(off)</code>              | prevents calling <code>ml search</code>                                |
| <code>repeat(#)</code>                | <code>ml search</code> 's <code>repeat()</code> option; see below      |
| <code>bounds(ml_search_bounds)</code> | specify bounds for <code>ml search</code>                              |
| <code>nowarning</code>                | suppress “convergence not achieved” message of <code>iterate(0)</code> |
| <code>novce</code>                    | substitute the zero matrix for the variance matrix                     |
| <code>negh</code>                     | indicates that the evaluator returns the negative Hessian matrix       |
| <code>score(newvars)</code>           | new variables containing the contribution to the score                 |
| <code>maximize_options</code>         | control the maximization process; seldom used                          |

---

| <i>search_options</i>         | description                                                                                                                                                             |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>repeat(#)</code>        | number of random attempts to find better initial-value vector; default is <code>repeat(10)</code> in interactive mode and <code>repeat(0)</code> in noninteractive mode |
| <code>restart</code>          | use random actions to find starting values; not recommended                                                                                                             |
| <code>norescale</code>        | do not rescale to improve parameter vector; not recommended                                                                                                             |
| <code>maximize_options</code> | control the maximization process; seldom used                                                                                                                           |

---

| <i>ml_maximize_options</i>          | description                                                            |
|-------------------------------------|------------------------------------------------------------------------|
| <code>nowarning</code>              | suppress “convergence not achieved” message of <code>iterate(0)</code> |
| <code>novce</code>                  | substitute the zero matrix for the variance matrix                     |
| <code>negh</code>                   | indicates that the evaluator returns the negative Hessian matrix       |
| <code>score(newvars   stub*)</code> | new variables containing the contribution to the score                 |
| <code>nooutput</code>               | suppress display of results                                            |
| <code>noclear</code>                | do not clear ml problem definition after model has converged           |
| <code>maximize_options</code>       | control the maximization process; seldom used                          |

---

| <i>display_options</i> | description                                                                                                     |
|------------------------|-----------------------------------------------------------------------------------------------------------------|
| <u>noheader</u>        | suppress header display above the coefficient table                                                             |
| <u>nofootnote</u>      | suppress footnote display below the coefficient table                                                           |
| <u>level(#)</u>        | set confidence level; default is <code>level(95)</code>                                                         |
| <u>first</u>           | display coefficient table reporting results for first equation only                                             |
| <u>neq(#)</u>          | display coefficient table reporting first # equations                                                           |
| <u>showeqns</u>        | display equation names in the coefficient table                                                                 |
| <u>plus</u>            | display coefficient table ending in dashes–plus–sign–dashes                                                     |
| <u>nocnsreport</u>     | suppress constraints display above the coefficient table                                                        |
| <u>noomitted</u>       | suppress display of omitted variables                                                                           |
| <u>vsquish</u>         | suppress blank space separating factor-variable terms or<br>time-series–operated variables from other variables |
| <u>noemptycells</u>    | suppress empty cells for interactions of factor variables                                                       |
| <u>baselevels</u>      | report base levels of factor variables and interactions                                                         |
| <u>allbaselevels</u>   | display all base levels of factor variables and interactions                                                    |
| <u>coeflegend</u>      | display coefficients’ legend instead of coefficient table                                                       |

| <i>eform_option</i>  | description                                                           |
|----------------------|-----------------------------------------------------------------------|
| <u>eform(string)</u> | display exponentiated coefficients; column title is “ <i>string</i> ” |
| <u>eform</u>         | display exponentiated coefficients; column title is “ <b>exp(b)</b> ” |
| <u>hr</u>            | report hazard ratios                                                  |
| <u>shr</u>           | report subhazard ratios                                               |
| <u>irr</u>           | report incidence-rate ratios                                          |
| <u>or</u>            | report odds ratios                                                    |
| <u>rrr</u>           | report relative-risk ratios                                           |

`fweights`, `awweights`, `iweights`, and `pweights` are allowed; see [U] 11.1.6 **weight**. With all but method `lf`, you must write your likelihood-evaluation program carefully if `pweights` are to be specified, and `pweights` may not be specified with method `d0`, `d1`, `d1debug`, `d2`, or `d2debug`. See Gould, Pitblado, and Sribney (2006, chap. 4) for details.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

To redisplay results, type `ml display`.

## Syntax of subroutines for use by method `d0`, `d1`, and `d2` evaluators

```

mleval newvar = vecname [, eq(#)]
mleval scalarname = vecname , scalar [eq(#)]

mlsum scalarnamelnf = exp [if] [, noweight]

mlvecsum scalarnamelnf rowvecname = exp [if] [, eq(#)]

mlmatsum scalarnamelnf matrixname = exp [if] [, eq(#[,#])]

mlmatbysum scalarnamelnf matrixname varnamea varnameb [varnamec] [if] ,
 by(varname) [eq(#[,#])]

```

## Syntax of user-written evaluator

### Summary of notation

The log-likelihood function is  $\ln L(\theta_{1j}, \theta_{2j}, \dots, \theta_{Ej})$ , where  $\theta_{ij} = \mathbf{x}_{ij} \mathbf{b}_i$ ,  $j = 1, \dots, N$  indexes observations, and  $i = 1, \dots, E$  indexes the linear equations defined by `ml model`. If the likelihood satisfies the linear-form restrictions, it can be decomposed as  $\ln L = \sum_{j=1}^N \ln \ell(\theta_{1j}, \theta_{2j}, \dots, \theta_{Ej})$ .

### Method lf evaluators:

```

program progname
 version 11
 args lnf theta1 [theta2 ...]
 /* if you need to create any intermediate results: */
 tempvar tmp1 tmp2 ...
 quietly gen double 'tmp1' = ...
 ...
 quietly replace 'lnf' = ...
end

where
'lnf' variable to be filled in with observation-by-observation values of $\ln \ell_j$
'theta1' variable containing evaluation of first equation $\theta_{1j} = \mathbf{x}_{1j} \mathbf{b}_1$
'theta2' variable containing evaluation of second equation $\theta_{2j} = \mathbf{x}_{2j} \mathbf{b}_2$

```

### Method d0 evaluators:

```

program progname
 version 11
 args todo b lnf
 tempvar theta1 theta2 ...
 mlevel 'theta1' = 'b', eq(1)
 mlevel 'theta2' = 'b', eq(2) // if there is a θ_2
 ...
 // if you need to create any intermediate results:
 tempvar tmp1 tmp2 ...
 gen double 'tmp1' = ...
 ...
 mlsum 'lnf' = ...
end

where
'todo' always contains 0 (may be ignored)
'b' full parameter row vector $\mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_E)$
'lnf' scalar to be filled in with overall $\ln L$

```

### Method d1 evaluators:

```

program progname
 version 11
 args todo b lnf g [negH g1 [g2 ...]]
 tempvar theta1 theta2 ...
 mlevel 'theta1' = 'b', eq(1)
 mlevel 'theta2' = 'b', eq(2) // if there is a θ_2
 ...
 // if you need to create any intermediate results:
 tempvar tmp1 tmp2 ...
 gen double 'tmp1' = ...
 ...

```

```

mlsum 'lnf' = ...
if ('todo'==0 | 'lnf'>=.) exit

tempname d1 d2 ...
mlvecsum 'lnf' 'd1' = formula for $\partial \ln \ell_j / \partial \theta_{1j}$, eq(1)
mlvecsum 'lnf' 'd2' = formula for $\partial \ln \ell_j / \partial \theta_{2j}$, eq(2)
...
matrix 'g' = ('d1','d2', ...)

end

where
'todo' contains 0 or 1
 0 $\Rightarrow$ 'lnf' to be filled in; 1 $\Rightarrow$ 'lnf' and 'g' to be filled in
'b' full parameter row vector $\mathbf{b}=(\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_E)$
'lnf' scalar to be filled in with overall $\ln L$
'g' row vector to be filled in with overall $\mathbf{g}=\partial \ln L / \partial \mathbf{b}$
'negH' argument to be ignored
'g1' variable optionally to be filled in with $\partial \ln \ell_j / \partial \mathbf{b}_1$
'g2' variable optionally to be filled in with $\partial \ln \ell_j / \partial \mathbf{b}_2$
...

```

### Method d2 evaluators:

```

program progname
version 11
args todo b lnf g negH [g1 [g2 ...]]

tempvar theta1 theta2 ...
mleval 'theta1' = 'b', eq(1)
mleval 'theta2' = 'b', eq(2) // if there is a θ_2
...

// if you need to create any intermediate results:
tempvar tmp1 tmp2 ...
gen double 'tmp1' = ...
...

mlsum 'lnf' = ...
if ('todo'==0 | 'lnf'>=.) exit

tempname d1 d2 ...
mlvecsum 'lnf' 'd1' = formula for $\partial \ln \ell_j / \partial \theta_{1j}$, eq(1)
mlvecsum 'lnf' 'd2' = formula for $\partial \ln \ell_j / \partial \theta_{2j}$, eq(2)
...
matrix 'g' = ('d1','d2', ...)
if ('todo'==1 | 'lnf'>=.) exit

tempname d11 d12 d22 ...
mlmatsum 'lnf' 'd11' = formula for $-\partial^2 \ln \ell_j / \partial \theta_{1j}^2$, eq(1)
mlmatsum 'lnf' 'd12' = formula for $-\partial^2 \ln \ell_j / \partial \theta_{1j} \partial \theta_{2j}$, eq(1,2)
mlmatsum 'lnf' 'd22' = formula for $-\partial^2 \ln \ell_j / \partial \theta_{2j}^2$, eq(2)
...
matrix 'negH' = ('d11','d12', ... \ 'd12','d22', ...)

end

where
'todo' contains 0, 1, or 2
 0 $\Rightarrow$ 'lnf' to be filled in; 1 $\Rightarrow$ 'lnf' and 'g' to be filled in;
 2 $\Rightarrow$ 'lnf', 'g', and 'negH' to be filled in
'b' full parameter row vector $\mathbf{b}=(\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_E)$
'lnf' scalar to be filled in with overall $\ln L$
'g' row vector to be filled in with overall $\mathbf{g}=\partial \ln L / \partial \mathbf{b}$
'negH' matrix to be filled in with overall negative Hessian $-\mathbf{H}=-\partial^2 \ln L / \partial \mathbf{b} \partial \mathbf{b}'$
'g1' variable optionally to be filled in with $\partial \ln \ell_j / \partial \mathbf{b}_1$
'g2' variable optionally to be filled in with $\partial \ln \ell_j / \partial \mathbf{b}_2$
...

```

## Global macros for use by all evaluators

|                        |                                                                                     |
|------------------------|-------------------------------------------------------------------------------------|
| <code>\$ML_y1</code>   | name of first dependent variable                                                    |
| <code>\$ML_y2</code>   | name of second dependent variable, if any                                           |
| <code>...</code>       |                                                                                     |
| <code>\$ML_samp</code> | variable containing 1 if observation to be used; 0 otherwise                        |
| <code>\$ML_w</code>    | variable containing weight associated with observation or 1 if no weights specified |

Method `lf` evaluators can ignore `$ML_samp`, but restricting calculations to the `$ML_samp==1` subsample will speed execution. Method `lf` evaluators must ignore `$ML_w`; application of weights is handled by the method itself.

Methods `d0`, `d1`, and `d2` can ignore `$ML_samp` as long as `ml model`'s `nopreserve` option is not specified. Methods `d0`, `d1`, and `d2` will run more quickly if `nopreserve` is specified. Method `d0`, `d1`, and `d2` evaluators can ignore `$ML_w` only if they use `mlsum`, `mlvecsum`, `mlmatsum`, and `mlmatbysum` to produce all final results.

## Description

`ml model` defines the current problem.

`ml clear` clears the current problem definition. This command is rarely used because, when you type `ml model`, any previous problem is automatically cleared.

`ml query` displays a description of the current problem.

`ml check` verifies that the log-likelihood evaluator you have written works. We strongly recommend using this command.

`ml search` searches for (better) initial values. We recommend using this command.

`ml plot` provides a graphical way of searching for (better) initial values.

`ml init` provides a way to specify initial values.

`ml report` reports  $\ln L$ 's values, gradient, and negative Hessian at the initial values or current parameter estimates,  $b_0$ .

`ml trace` traces the execution of the user-defined log-likelihood evaluation program.

`ml count` counts the number of times the user-defined log-likelihood evaluation program is called; this command is seldom used. `ml count clear` clears the counter. `ml count on` turns on the counter. `ml count` without arguments reports the current values of the counter. `ml count off` stops counting calls.

`ml maximize` maximizes the likelihood function and reports results. Once `ml maximize` has successfully completed, the previously mentioned `ml` commands may no longer be used unless `noclear` is specified. `ml graph` and `ml display` may be used whether or not `noclear` is specified.

`ml graph` graphs the log-likelihood values against the iteration number.

`ml display` redisplay results.

`ml footnote` displays a warning message when the model did not converge within the specified number of iterations.

`ml score` creates new variables containing the equation-level scores. The variables generated by `ml score` are equivalent to those generated by specifying the `score()` option of `ml maximize` (and `ml model ... , ... maximize`).

*progrname* is the name of a Stata program you write to evaluate the log-likelihood function.

*funcname()* is the name of a Mata function you write to evaluate the log-likelihood function.

In this documentation, *progname* and *funcname()* are referred to as the user-written evaluator, the likelihood evaluator, or sometimes simply as the evaluator. The program you write is written in the style required by the method you choose. The methods are lf, d0, d1, d2, e1, e2, v1, and v2. Thus, if you choose to use method lf, your program is called a method of lf evaluator.

Method lf evaluators are required to evaluate the observation-by-observation log likelihood  $\ln \ell_j$ ,  $j = 1, \dots, N$ . Method d0 evaluators are required to evaluate the overall log likelihood  $\ln L$ . Method d1 evaluators are required to evaluate the overall log likelihood and its gradient vector  $\mathbf{g} = \partial \ln L / \partial \mathbf{b}$ . Method d2 evaluators are required to evaluate the overall log likelihood, its gradient, and its negative Hessian matrix  $-\mathbf{H} = -\partial^2 \ln L / \partial \mathbf{b} \partial \mathbf{b}'$ .

`mleval` is a subroutine used by method d0, d1, and d2 evaluators to evaluate the coefficient vector,  $\mathbf{b}$ , that they are passed.

`mlsum` is a subroutine used by method d0, d1, and d2 evaluators to define the value,  $\ln L$ , that is to be returned.

`mlvecsum` is a subroutine used by method d1 and d2 evaluators to define the gradient vector,  $\mathbf{g}$ , that is to be returned. It is suitable for use only when the likelihood function meets the linear-form restrictions.

`mlmatsum` is a subroutine used by method d2 evaluators to define the negative Hessian matrix,  $-\mathbf{H}$ , that is to be returned. It is suitable for use only when the likelihood function meets the linear-form restrictions.

`mlmatbysum` is a subroutine used by method d2 evaluators to help define the negative Hessian matrix,  $-\mathbf{H}$ , that is to be returned. It is suitable for use when the likelihood function contains terms made up of grouped sums, such as in panel-data models. For such models, use `mlmatsum` to compute the observation-level outer products and `mlmatbysum` to compute the group-level outer products. `mlmatbysum` requires that the data be sorted by the variable identified in the `by()` option.

## Options for use with ml model in interactive or noninteractive mode

`vce(vcetype)` specifies the type of standard error reported, which includes types that are robust to some kinds of misspecification, that allow for intragroup correlation, and that are derived from asymptotic theory; see [R] [vce\\_option](#).

`vce(robust)`, `vce(cluster clustvar)`, `pweight`, and `svy` will work with a method lf, e1, e2, v0, v1, and v2 evaluators; all you need do is specify them.

These options will not work with a method d0, d1, or d2 evaluators, and specifying these options will produce an error message.

`constraints(numlist | matname)` specifies the linear constraints to be applied during estimation. `constraints(numlist)` specifies the constraints by number. Constraints are defined by using the `constraint` command; see [R] [constraint](#). `constraint(matname)` specifies a matrix that contains the constraints.

`nocnsnotes` prevents notes from being displayed when constraints are dropped. A constraint will be dropped if it is inconsistent, contradicts other constraints, or causes some other error when the constraint matrix is being built. Constraints are checked in the order in which they are specified.

`title(string)` specifies the title for the estimation output when results are complete.

`nopreserve` specifies that `ml` need not ensure that only the estimation subsample is in memory when the user-written likelihood evaluator is called. `nopreserve` is irrelevant when you use method lf.



For the other methods, if `nopreserve` is not specified, `ml` saves the data in a file (preserves the original dataset) and drops the irrelevant observations before calling the user-written evaluator. This way, even if the evaluator does not restrict its attentions to the `$ML_samp==1` subsample, results will still be correct. Later, `ml` automatically restores the original dataset.

`ml` need not go through these machinations for method `lf` because the user-written evaluator calculates observation-by-observation values, and `ml` itself sums the components.

`ml` goes through these machinations if and only if the estimation sample is a subsample of the data in memory. If the estimation sample includes every observation in memory, `ml` does not preserve the original dataset. Thus programmers must not alter the original dataset unless they `preserve` the data themselves.

We recommend that interactive users of `ml` not specify `nopreserve`; the speed gain is not worth the possibility of getting incorrect results.

We recommend that programmers specify `nopreserve`, but only after verifying that their evaluator really does restrict its attentions solely to the `$ML_samp==1` subsample.

`collinear` specifies that `ml` not remove the collinear variables within equations. There is no reason to leave collinear variables in place, but this option is of interest to programmers who, in their code, have already removed collinear variables and do not want `ml` to waste computer time checking again.

`missing` specifies that observations containing variables with missing values not be eliminated from the estimation sample. There are two reasons you might want to specify `missing`:

Programmers may wish to specify `missing` because, in other parts of their code, they have already eliminated observations with missing values and do not want `ml` to waste computer time looking again.

You may wish to specify `missing` if your model explicitly deals with missing values. Stata's `heckman` command is a good example of this. In such cases, there will be observations where missing values are allowed and other observations where they are not—where their presence should cause the observation to be eliminated. If you specify `missing`, it is your responsibility to specify an `if exp` that eliminates the irrelevant observations.

`lf0(#k #ll)` is typically used by programmers. It specifies the number of parameters and log-likelihood value of the constant-only model so that `ml` can report a likelihood-ratio test rather than a Wald test. These values may have been analytically determined, or they may have been determined by a previous fitting of the constant-only model on the estimation sample.

Also see the `continue` option directly below.

If you specify `lf0()`, it must be safe for you to specify the `missing` option, too, else how did you calculate the log likelihood for the constant-only model on the same sample? You must have identified the estimation sample, and done so correctly, so there is no reason for `ml` to waste time rechecking your results. All of which is to say, do not specify `lf0()` unless you are certain your code identifies the estimation sample correctly.

`lf0()`, even if specified, is ignored if `vce(robust)`, `vce(cluster clustvar)`, `pweight`, or `svy` is specified because, in that case, a likelihood-ratio test would be inappropriate.

`continue` is typically specified by programmers and does two things:

First, it specifies that a model has just been fit by either `ml` or some other estimation command, such as `logit`, and that the likelihood value stored in `e(ll)` and the number of parameters stored in `e(b)` as of that instant are the relevant values of the constant-only model. The current value of the log likelihood is used to present a likelihood-ratio test unless `vce(robust)`, `vce(cluster`

`clustvar`), `pweight`, `svy`, or `constraints()` is specified. A likelihood-ratio test is inappropriate when `vce(robust)`, `vce(cluster clustvar)`, `pweight`, or `svy` is specified. We suggest using `lrtest` when `constraints()` is specified.

Second, `continue` sets the initial values,  $b_0$ , for the model about to be fit according to the  $e(b)$  currently stored.

The comments made about specifying missing with `lf0()` apply equally well here.

`waldtest(#)` is typically specified by programmers. By default, `ml` presents a Wald test, but that is overridden if the `lf0()` or `continue` option is specified. A Wald test is performed if `vce(robust)`, `vce(cluster clustvar)`, or `pweight` is specified.

`waldtest(0)` prevents even the Wald test from being reported.

`waldtest(-1)` is the default. It specifies that a Wald test be performed by constraining all coefficients except the intercept to 0 in the first equation. Remaining equations are to be unconstrained. A Wald test is performed if neither `lf0()` nor `continue` was specified, and a Wald test is forced if `vce(robust)`, `vce(cluster clustvar)`, or `pweight` was specified.

`waldtest( $k$ )` for  $k \leq -1$  specifies that a Wald test be performed by constraining all coefficients except intercepts to 0 in the first  $|k|$  equations; remaining equations are to be unconstrained. A Wald test is performed if neither `lf0()` nor `continue` was specified, and a Wald test is forced if `vce(robust)`, `vce(cluster clustvar)`, or `pweight` was specified.

`waldtest( $k$ )` for  $k \geq 1$  works like the options above, except that it forces a Wald test to be reported even if the information to perform the likelihood-ratio test is available and even if none of `vce(robust)`, `vce(cluster clustvar)`, or `pweight` was specified. `waldtest( $k$ )`,  $k \geq 1$ , may not be specified with `lf0()`.

`obs(#)` is used mostly by programmers. It specifies that the number of observations reported and ultimately stored in `e(N)` be  $\#$ . Ordinarily, `ml` works that out for itself. Programmers may want to specify this option when, for the likelihood evaluator to work for  $N$  observations, they first had to modify the dataset so that it contained a different number of observations.

`noscvvars` is used mostly by programmers. It specifies that method `d0`, `d1`, or `d2` is being used but that the likelihood-evaluation program does not calculate or use arguments `'g1'`, `'g2'`, etc., which are the score vectors. Thus `ml` can save a little time by not generating and passing those arguments.

`crittype(string)` is used mostly by programmers. It allows programmers to supply a string (up to 32 characters long) that describes the criterion that is being optimized by `ml`. The default is `"log likelihood"` for nonrobust and `"log pseudolikelihood"` for robust estimation.

`svy` indicates that `ml` is to pick up the `svy` settings set by `svyset` and use the robust variance estimator. This option requires the data to be `svyset`. `svy` may not be specified with `vce()` or `weights`.

`subpop(varname)` specifies that estimates be computed for the single subpopulation defined by the observations for which `varname`  $\neq 0$ . Typically, `varname = 1` defines the subpopulation, and `varname = 0` indicates observations not belonging to the subpopulation. For observations whose subpopulation status is uncertain, `varname` should be set to missing (`'.'`). This option requires the `svy` option.

`nosvyadjust` specifies that the model Wald test be carried out as  $W/k \sim F(k, d)$ , where  $W$  is the Wald test statistic,  $k$  is the number of terms in the model excluding the constant term,  $d$  is the total number of sampled PSUs minus the total number of strata, and  $F(k, d)$  is an  $F$  distribution with  $k$  numerator degrees of freedom and  $d$  denominator degrees of freedom. By default, an adjusted Wald test is conducted:  $(d - k + 1)W/(kd) \sim F(k, d - k + 1)$ . See [Korn and Graubard \(1990\)](#) for a discussion of the Wald test and the adjustments thereof. This option requires the `svy` option.

`technique(algorithm_spec)` specifies how the likelihood function is to be maximized. The following algorithms are currently implemented in `ml`. For details, see [Gould, Pitblado, and Sribney \(2006\)](#).

`technique(nr)` specifies Stata's modified Newton–Raphson (NR) algorithm.

`technique(bhhh)` specifies the Berndt–Hall–Hall–Hausman (BHHH) algorithm.

`technique(dfpr)` specifies the Davidon–Fletcher–Powell (DFP) algorithm.

`technique(bfgs)` specifies the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm.

The default is `technique(nr)`.

You can switch between algorithms by specifying more than one in the `technique()` option. By default, `ml` will use an algorithm for five iterations before switching to the next algorithm. To specify a different number of iterations, include the number after the technique in the option. For example, `technique(bhhh 10 nr 1000)` requests that `ml` perform 10 iterations using the BHHH algorithm, followed by 1,000 iterations using the NR algorithm, and then switch back to BHHH for 10 iterations, and so on. The process continues until convergence or until reaching the maximum number of iterations.

## Options for use with `ml model` in noninteractive mode

The following extra options are for use with `ml model` in noninteractive mode. Noninteractive mode is for programmers who use `ml` as a subroutine and want to issue one command that will carry forth the estimation from start to finish.

`maximize` is required. It specifies noninteractive mode.

`init(ml_init_args)` sets the initial values,  $b_0$ . *ml\_init\_args* are whatever you would type after the `ml init` command.

`search(on | norescale | quietly | off)` specifies whether `ml search` is to be used to improve the initial values. `search(on)` is the default and is equivalent to separately running `ml search`, `repeat(0)`. `search(norescale)` is equivalent to separately running `ml search`, `repeat(0) norescale`. `search(quietly)` is equivalent to `search(on)`, except that it suppresses `ml search`'s output. `search(off)` prevents calling `ml search`.

`repeat(#)` is `ml search`'s `repeat()` option. `repeat(0)` is the default.

`bounds(ml_search_bounds)` specifies the search bounds. The `ml model` command issues `ml search ml_search_bounds`, `repeat(#)`. Specifying search bounds is optional.

`nowarning`, `novce`, `negh`, and `score()` are `ml maximize`'s equivalent options.

*maximize\_options*: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, [`no`]`log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [\[R\] maximize](#). These options are seldom used.

## Options for use when specifying equations

`noconstant` specifies that the equation not include an intercept.

`offset(varnameo)` specifies that the equation be  $\mathbf{x}\mathbf{b} + \textit{varname}_o$ —that it include *varname<sub>o</sub>* with coefficient constrained to be 1.

`exposure(varnamee)` is an alternative to `offset(varnameo)`; it specifies that the equation be  $\mathbf{x}\mathbf{b} + \ln(\textit{varname}_e)$ . The equation is to include  $\ln(\textit{varname}_e)$  with coefficient constrained to be 1.

## Options for use with ml search

`repeat(#)` specifies the number of random attempts that are to be made to find a better initial-value vector. The default is `repeat(10)`.

`repeat(0)` specifies that no random attempts be made. More precisely, `repeat(0)` specifies that no random attempts be made if the first initial-value vector is a feasible starting point. If it is not, `ml search` will make random attempts, even if you specify `repeat(0)`, because it has no alternative. The `repeat()` option refers to the number of random attempts to be made to improve the initial values. When the initial starting value vector is not feasible, `ml search` will make up to 1,000 random attempts to find starting values. It stops when it finds one set of values that works and then moves into its improve-initial-values logic.

`repeat(k)`,  $k > 0$ , specifies the number of random attempts to be made to improve the initial values.

`restart` specifies that random actions be taken to obtain starting values and that the resulting starting values not be a deterministic function of the current values. Generally, you should not specify this option because, with `restart`, `ml search` intentionally does not produce as good a set of starting values as it could. `restart` is included for use by the optimizer when it gets into serious trouble. The random actions ensure that the optimizer and `ml search`, working together, do not cause an endless loop.

`restart` implies `norescale`, which is why we recommend that you do not specify `restart`. In testing, sometimes `rescale` worked so well that, even after randomization, the rescaler would bring the starting values right back to where they had been the first time and thus defeat the intended randomization.

`norescale` specifies that `ml search` not engage in its rescaling actions to improve the parameter vector. We do not recommend specifying this option because rescaling tends to work so well.

`maximize_options`: `[no]log`, `trace`; see [\[R\] maximize](#). These options are seldom used.

## Option for use with ml plot

`saving(filename[, replace])` specifies that the graph be saved in `filename.gph`.

See [\[G\] saving-option](#).

## Options for use with ml init

`copy` specifies that the list of numbers or the initialization vector be copied into the initial-value vector by position rather than by name.

`skip` specifies that any parameters found in the specified initialization vector that are not also found in the model be ignored. The default action is to issue an error message.

## Options for use with ml maximize

`nowarning` is allowed only with `iterate(0)`. `nowarning` suppresses the “convergence not achieved” message. Programmers might specify `iterate(0) nowarning` when they have a vector `b` already containing the final estimates and want `ml` to calculate the variance matrix and postestimation results. Then specify `init(b) search(off) iterate(0) nowarning nolog`.

`novce` is allowed only with `iterate(0)`. `novce` substitutes the zero matrix for the variance matrix, which in effect posts estimation results as fixed constants.

`negh` indicates that the evaluator returns the negative Hessian matrix. By default, `ml` assumes `d2`, `e2`, and `v2` evaluators return the Hessian matrix.

`score(newvars|stub*)` creates new variables containing the contributions to the score for each equation and ancillary parameter in the model; see [U] 20.17 Obtaining scores.

If `score(newvars)` is specified, the `newvars` must contain  $k$  new variables, one for each equation in the model. If `score(stub*)` is specified, variables named `stub1`, `stub2`, ..., `stubk` are created.

The first variable contains  $\partial \ln L_j / \partial (\mathbf{x}_{1j} \mathbf{b}_1)$ , the second variable contains  $\partial \ln L_j / \partial (\mathbf{x}_{2j} \mathbf{b}_2)$ , and so on.

`nooutput` suppresses display of results. This option is different from prefixing `ml maximize` with `quietly` in that the iteration log is still displayed (assuming that `nolog` is not specified).

`noclear` specifies that the `ml` problem definition not be cleared after the model has converged. Perhaps you are having convergence problems and intend to run the model to convergence. If so, use `ml search` to see if those values can be improved, and then restart the estimation.

*maximize\_options*: difficult, iterate(#), [no]log, trace, gradient, showstep, hessian, showtolerance, tolerance(#), ltolerance(#), nrtolerance(#), nonrtolerance; see [R] [maximize](#). These options are seldom used.

*display\_options*; see *Options for use with ml display* below.

*eform\_option*; see *Options for use with ml display* below.

## Option for use with ml graph

`saving(filename[, replace])` specifies that the graph be saved in `filename.gph`.

See [G] [saving\\_option](#).

## Options for use with ml display

`noheader` suppresses the header display above the coefficient table that displays the final log-likelihood value, the number of observations, and the model significance test.

`nofootnote` suppresses the footnote display below the coefficient table, which displays a warning if the model fit did not converge within the specified number of iterations. Use `ml footnote` to display the warning if 1) you add to the coefficient table using the `plus` option or 2) you have your own footnotes and want the warning to be last.

`level(#)` is the standard confidence-level option. It specifies the confidence level, as a percentage, for confidence intervals of the coefficients. The default is `level(95)` or as set by `set level`; see [U] 20.7 Specifying the width of confidence intervals.

`first` displays a coefficient table reporting results for the first equation only, and the report makes it appear that the first equation is the only equation. This option is used by programmers who estimate ancillary parameters in the second and subsequent equations and who wish to report the values of such parameters themselves.

`neq(#)` is an alternative to `first`. `neq(#)` displays a coefficient table reporting results for the first  $\#$  equations. This option is used by programmers who estimate ancillary parameters in the  $\# + 1$  and subsequent equations and who wish to report the values of such parameters themselves.

`showeqns` is a seldom-used option that displays the equation names in the coefficient table. `ml display` uses the numbers stored in `e(k_eq)` and `e(k_aux)` to determine how to display the coefficient table. `e(k_eq)` identifies the number of equations, and `e(k_aux)` identifies how many of these are for ancillary parameters. The `first` option is implied when `showeqns` is not specified and all but the first equation are for ancillary parameters.

`plus` displays the coefficient table, but rather than ending the table in a line of dashes, ends it in dashes-plus-sign-dashes. This is so that programmers can write additional display code to add more results to the table and make it appear as if the combined result is one table. Programmers typically specify `plus` with the `first` or `neq()` options. This option implies `nofootnote`.

`nocnsreport` suppresses the display of constraints above the coefficient table. This option is ignored if constraints were not used to fit the model.

`noomitted` specifies that variables that were omitted because of collinearity not be displayed. The default is to include in the table any variables omitted because of collinearity and to label them as “(omitted)”.

`vsquish` specifies that the blank space separating factor-variable terms or time-series-operated variables from other variables in the model be suppressed.

`noemptycells` specifies that empty cells for interactions of factor variables not be displayed. The default is to include in the table interaction cells that do not occur in the estimation sample and to label them as “(empty)”.

`baselevels` and `allbaselevels` control whether the base levels of factor variables and interactions are displayed. The default is to exclude from the table all base categories.

`baselevels` specifies that base levels be reported for factor variables and for interactions whose bases cannot be inferred from their component factor variables.

`allbaselevels` specifies that all base levels of factor variables and interactions be reported.

`coeflegend` specifies that the legend of the coefficients and how to specify them in an expression be displayed rather than the coefficient table.

*eform\_option*: `eform(string)`, `eform`, `hr`, `shr`, `irr`, `or`, and `rrr` display the coefficient table in exponentiated form: for each coefficient,  $\exp(b)$  rather than  $b$  is displayed, and standard errors and confidence intervals are transformed. Display of the intercept, if any, is suppressed. *string* is the table header that will be displayed above the transformed coefficients and must be 11 characters or shorter in length—for example, `eform("Odds ratio")`. The options `eform`, `hr`, `shr`, `irr`, `or`, and `rrr` provide a default *string* equivalent to “exp(b)”, “Haz. Ratio”, “SHR”, “IRR”, “Odds Ratio”, and “RRR”, respectively. These options may not be combined.

`ml display` looks at `e(k_eform)` to determine how many equations are affected by an *eform\_option*; by default, only the first equation is affected.

## Options for use with `mlevel`

`eq(#)` specifies the equation number,  $i$ , for which  $\theta_{ij} = \mathbf{x}_{ij}\mathbf{b}_i$  is to be evaluated. `eq(1)` is assumed if `eq()` is not specified.

`scalar` asserts that the  $i$ th equation is known to evaluate to a constant, meaning that the equation was specified as `()`, `(name:)`, or `/name` on the `ml model` statement. If you specify this option, the new variable created is created as a scalar. If the  $i$ th equation does not evaluate to a scalar, an error message is issued.

## Option for use with mlsum

`noweight` specifies that weights (`$ML_w`) be ignored when summing the likelihood function.

## Option for use with mlvecsum

`eq(#)` specifies the equation for which a gradient vector  $\partial \ln L / \partial \mathbf{b}_i$  is to be constructed. The default is `eq(1)`.

## Option for use with mlmatsum

`eq([#, #])` specifies the equations for which the negative Hessian matrix is to be constructed. The default is `eq(1)`, which is the same as `eq(1,1)`, which means  $-\partial^2 \ln L / \partial \mathbf{b}_1 \partial \mathbf{b}'_1$ . Specifying `eq(i,j)` results in  $-\partial^2 \ln L / \partial \mathbf{b}_i \partial \mathbf{b}'_j$ .

## Options for use with mlmatbysum

`by(varname)` is required and specifies the group variable.

`eq([#, #])` specifies the equations for which the negative Hessian matrix is to be constructed. The default is `eq(1)`, which is the same as `eq(1,1)`, which means  $-\partial^2 \ln L / \partial \mathbf{b}_1 \partial \mathbf{b}'_1$ . Specifying `eq(i,j)` results in  $-\partial^2 \ln L / \partial \mathbf{b}_i \partial \mathbf{b}'_j$ .

## Options for use with ml score

`equation(eqname)` identifies from which equation the observation scores are to come. This option may be used only when generating one variable.

`forcescores` causes `ml score` to generate scores from results generated from methods `d1` and `d2` even though the estimation command does not have the `ml_score` property.

`missing` specifies that observations containing variables with missing values not be eliminated from the estimation sample.

(Continued on next page)

## Remarks

For a thorough discussion of `ml`, see the third edition of *Maximum Likelihood Estimation with Stata*, (Gould, Pitblado, and Sribney 2006). The book provides a tutorial introduction to `ml`, notes on advanced programming issues, and a discourse on maximum likelihood estimation from both theoretical and practical standpoints. See *Survey options and ml* at the end of *Remarks* for examples of the new `svy` options. For more information about survey estimation, see [\[SVY\] survey](#), [\[SVY\] svy estimation](#), and [\[SVY\] variance estimation](#).

`ml` requires that you write a program that evaluates the log-likelihood function and, possibly, its first and second derivatives. The style of the program you write depends upon the method you choose: methods `lf` and `d0` require that your program evaluate the log likelihood only, method `d1` requires that your program evaluate the log likelihood and gradient, and method `d2` requires that your program evaluate the log likelihood, gradient, and negative Hessian. Methods `lf` and `d0` differ from each other in that, with method `lf`, your program is required to produce observation-by-observation log-likelihood values  $\ln \ell_j$  and it is assumed that  $\ln L = \sum_j \ln \ell_j$ ; with method `d0`, your program is required to produce only the overall value  $\ln L$ .

Once you have written the program—called an evaluator—you define a model to be fit using `ml model` and obtain estimates using `ml maximize`. You might type

```
. ml model ...
. ml maximize
```

but we recommend that you type

```
. ml model ...
. ml check
. ml search
. ml maximize
```

`ml check` verifies your evaluator has no obvious errors, and `ml search` finds better initial values.

You fill in the `ml model` statement with 1) the method you are using, 2) the name of your program, and 3) the “equations”. You write your evaluator in terms of  $\theta_1, \theta_2, \dots$ , each of which has a linear equation associated with it. That linear equation might be as simple as  $\theta_i = b_0$ , it might be  $\theta_i = b_1\text{mpg} + b_2\text{weight} + b_3$ , or it might omit the intercept  $b_3$ . The equations are specified in parentheses on the `ml model` line.

Suppose that you are using method `lf` and the name of your evaluator program is `myprog`. The statement

```
. ml model lf myprog (mpg weight)
```

would specify one equation with  $\theta_i = b_1\text{mpg} + b_2\text{weight} + b_3$ . If you wanted to omit  $b_3$ , you would type

```
. ml model lf myprog (mpg weight, nocons)
```

and if all you wanted was  $\theta_i = b_0$ , you would type

```
. ml model lf myprog ()
```

With multiple equations, you list the equations one after the other; so, if you typed

```
. ml model lf myprog (mpg weight) ()
```

you would be specifying  $\theta_1 = b_1\text{mpg} + b_2\text{weight} + b_3$  and  $\theta_2 = b_4$ . You would write your likelihood in terms of  $\theta_1$  and  $\theta_2$ . If the model was linear regression,  $\theta_1$  might be the `xb` part and  $\theta_2$  the variance of the residuals.



When you specify the equations, you also specify any dependent variables. If you typed

```
. ml model lf myprog (price = mpg weight) ()
```

price would be the one and only dependent variable, and that would be passed to your program in `$ML_y1`. If your model had two dependent variables, you could type

```
. ml model lf myprog (price displ = mpg weight) ()
```

Then `$ML_y1` would be `price` and `$ML_y2` would be `displ`. You can specify however many dependent variables are necessary and specify them on any equation. It does not matter on which equation you specify them; the first one specified is placed in `$ML_y1`, the second in `$ML_y2`, and so on.

## ► Example 1: Method lf

Using method `lf`, we want to produce observation-by-observation values of the log likelihood. The probit log-likelihood function is

$$\ln \ell_j = \begin{cases} \ln \Phi(\theta_{1j}) & \text{if } y_j = 1 \\ \ln \Phi(-\theta_{1j}) & \text{if } y_j = 0 \end{cases}$$

$$\theta_{1j} = \mathbf{x}_j \mathbf{b}_1$$

The following is the method `lf` evaluator for this likelihood function:

```
program myprobit
 version 11
 args lnf theta1
 quietly replace `lnf' = ln(normal(`theta1')) if $ML_y1==1
 quietly replace `lnf' = ln(normal(-`theta1')) if $ML_y1==0
end
```

If we wanted to fit a model of `foreign` on `mpg` and `weight`, we would type

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. ml model lf myprobit (foreign = mpg weight)

. ml maximize
```

The `'foreign ='` part specifies that  $y$  is `foreign`. The `'mpg weight'` part specifies that  $\theta_{1j} = b_1 \text{mpg}_j + b_2 \text{weight}_j + b_3$ . The result of running this is

*(Continued on next page)*

```
. ml model lf myprobit (foreign = mpg weight)
. ml maximize

initial: log likelihood = -51.292891
alternative: log likelihood = -45.055272
rescale: log likelihood = -45.055272
Iteration 0: log likelihood = -45.055272
Iteration 1: log likelihood = -27.904114
Iteration 2: log likelihood = -26.858048
Iteration 3: log likelihood = -26.844198
Iteration 4: log likelihood = -26.844189
Iteration 5: log likelihood = -26.844189

 Number of obs = 74
 Wald chi2(2) = 20.75
 Prob > chi2 = 0.0000

Log likelihood = -26.844189
```

| foreign | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|---------|-----------|-----------|-------|-------|----------------------|-----------|
| mpg     | -.1039503 | .0515689  | -2.02 | 0.044 | -.2050235            | -.0028772 |
| weight  | -.0023355 | .0005661  | -4.13 | 0.000 | -.003445             | -.0012261 |
| _cons   | 8.275464  | 2.554142  | 3.24  | 0.001 | 3.269438             | 13.28149  |



➤ Example 2: Method lf for two-equation, two-dependent-variable model

A two-equation, two-dependent-variable model is a little different. Rather than receiving one  $\theta$ , our program will receive two. Rather than there being one dependent variable in `$ML_y1`, there will be dependent variables in `$ML_y1` and `$ML_y2`. For instance, the Weibull regression log-likelihood function is

$$\ln \ell_j = -(t_j e^{-\theta_{1j}})^{\exp(\theta_{2j})} + d_j \{ \theta_{2j} - \theta_{1j} + (e^{\theta_{2j}} - 1)(\ln t_j - \theta_{1j}) \}$$
$$\theta_{1j} = \mathbf{x}_j \mathbf{b}_1$$
$$\theta_{2j} = s$$

where  $t_j$  is the time of failure or censoring and  $d_j = 1$  if failure and 0 if censored. We can make the log likelihood a little easier to program by introducing some extra variables:

$$p_j = \exp(\theta_{2j})$$
$$M_j = \{t_j \exp(-\theta_{1j})\}^{p_j}$$
$$R_j = \ln t_j - \theta_{1j}$$
$$\ln \ell_j = -M_j + d_j \{ \theta_{2j} - \theta_{1j} + (p_j - 1)R_j \}$$

The method lf evaluator for this is

```

program myweib
 version 11
 args lnf theta1 theta2

 tempvar p M R
 quietly gen double 'p' = exp('theta2')
 quietly gen double 'M' = ($ML_y1*exp(-'theta1'))^'p'
 quietly gen double 'R' = ln($ML_y1)-'theta1'

 quietly replace 'lnf' = -'M' + $ML_y2*('theta2'-'theta1' + ('p'-1)*'R')
end

```

We can fit a model by typing

```

. ml model lf myweib (studytime died = i.drug age) ()
. ml maximize

```

Note that we specified `('')` for the second equation. The second equation corresponds to the Weibull shape parameter  $s$ , and the linear combination we want for  $s$  contains just an intercept. Alternatively, we could type

```

. ml model lf myweib (studytime died = i.drug age) /s

```

Typing `/s` means the same thing as typing `(s:)`, and both really mean the same thing as `( )`. The `s`, either after a slash or in parentheses before a colon, labels the equation. It makes the output look prettier, and that is all:

```

. use http://www.stata-press.com/data/r11/cancer, clear
(Patient Survival in Drug Trial)

. ml model lf myweib (studytime died = i.drug age) /s
. ml maximize

initial: log likelihood = -744
alternative: log likelihood = -356.14276
rescale: log likelihood = -200.80201
rescale eq: log likelihood = -136.69232
Iteration 0: log likelihood = -136.69232 (not concave)
Iteration 1: log likelihood = -124.11726
Iteration 2: log likelihood = -113.89828
Iteration 3: log likelihood = -110.30451
Iteration 4: log likelihood = -110.26747
Iteration 5: log likelihood = -110.26736
Iteration 6: log likelihood = -110.26736

```

```

Log likelihood = -110.26736
Number of obs = 48
Wald chi2(3) = 35.25
Prob > chi2 = 0.0000

```

|     |       | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-----|-------|-----------|-----------|-------|-------|----------------------|-----------|
| eq1 | drug  |           |           |       |       |                      |           |
|     | 2     | 1.012966  | .2903917  | 3.49  | 0.000 | .4438086             | 1.582123  |
|     | 3     | 1.45917   | .2821195  | 5.17  | 0.000 | .9062261             | 2.012114  |
|     | age   | -.0671728 | .0205688  | -3.27 | 0.001 | -.1074868            | -.0268587 |
|     | _cons | 6.060723  | 1.152845  | 5.26  | 0.000 | 3.801188             | 8.320259  |
| s   |       |           |           |       |       |                      |           |
|     | _cons | .5573333  | .1402154  | 3.97  | 0.000 | .2825162             | .8321504  |

### ► Example 3: Method d0

Method d0 evaluators receive  $\mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_E)$ , the coefficient vector, rather than the already evaluated  $\theta_1, \theta_2, \dots, \theta_E$ , and they are required to evaluate the overall log-likelihood  $\ln L$  rather than  $\ln \ell_j$ ,  $j = 1, \dots, N$ .

Use `mlevel` to produce the thetas from the coefficient vector.

Use `mlsum` to sum the components that enter into  $\ln L$ .

In the case of Weibull,  $\ln L = \sum \ln \ell_j$ , and our method d0 evaluator is

```
program weib0
 version 11
 args todo b lnf
 tempvar theta1 theta2
 mlevel 'theta1' = 'b', eq(1)
 mlevel 'theta2' = 'b', eq(2)
 local t "$ML_y1" // this is just for readability
 local d "$ML_y2"
 tempvar p M R
 quietly gen double 'p' = exp('theta2')
 quietly gen double 'M' = ('t'*exp(-'theta1'))^'p'
 quietly gen double 'R' = ln('t')-'theta1'
 mlsum 'lnf' = -'M' + 'd'*('theta2'-'theta1' + ('p'-1)*'R')
end
```

To fit our model using this evaluator, we would type

```
. ml model d0 weib0 (studytime died = i.drug age) /s
. ml maximize
```

◀

### □ Technical note

Method d0 does not require  $\ln L = \sum_j \ln \ell_j$ ,  $j = 1, \dots, N$ , as method lf does. Your likelihood function might have independent components only for groups of observations. Panel-data estimators have a log-likelihood value  $\ln L = \sum_i \ln L_i$ , where  $i$  indexes the panels, each of which contains multiple observations. Conditional logistic regression has  $\ln L = \sum_k \ln L_k$ , where  $k$  indexes the risk pools. Cox regression has  $\ln L = \sum_{(t)} \ln L_{(t)}$ , where  $(t)$  denotes the ordered failure times.

To evaluate such likelihood functions, first calculate the within-group log-likelihood contributions. This usually involves `generate` and `replace` statements prefixed with `by`, as in

```
tempvar sumd
by group: gen double 'sumd' = sum($ML_y1)
```

Structure your code so that the log-likelihood contributions are recorded in the last observation of each group. Say that a variable is named `'cont'`. To sum the contributions, code

```
tempvar last
quietly by group: gen byte 'last' = (_n==_N)
mlsum 'lnf' = 'cont' if 'last'
```

You must inform `mlsum` which observations contain log-likelihood values to be summed. First, you do not want to include intermediate results in the sum. Second, `mlsum` does not skip missing values. Rather, if `mlsum` sees a missing value among the contributions, it sets the overall result, `'lnf'`, to missing. That is how `ml maximize` is informed that the likelihood function could not be evaluated at the particular value of `b`. `ml maximize` will then take action to escape from what it thinks is an infeasible area of the likelihood function.

When the likelihood function violates the linear-form restriction  $\ln L = \sum_j \ln \ell_j$ ,  $j = 1, \dots, N$ , with  $\ln \ell_j$  being a function solely of values within the  $j$ th observation, use method d0. In the following examples, we will demonstrate methods d1 and d2 with likelihood functions that meet this linear-form restriction. The d1 and d2 methods themselves do not require the linear-form restriction, but the utility routines `mlvecsum` and `mlmatsum` do. Using method d1 or d2 when the restriction is violated is difficult; however, `mlmatbysum` may be of some help for method d2 evaluators.  $\square$

#### ► Example 4: Method d1

Method d1 evaluators are required to produce the gradient vector  $\mathbf{g} = \partial \ln L / \partial \mathbf{b}$ , as well as the overall log-likelihood value. Using `mlvecsum`, we can obtain  $\partial \ln L / \partial \mathbf{b}$  from  $\partial \ln L / \partial \theta_i$ ,  $i = 1, \dots, E$ . The derivatives of the Weibull log-likelihood function are

$$\frac{\partial \ln \ell_j}{\partial \theta_{1j}} = p_j (M_j - d_j)$$

$$\frac{\partial \ln \ell_j}{\partial \theta_{2j}} = d_j - R_j p_j (M_j - d_j)$$

The method d1 evaluator for this is

```

program weib1
 version 11
 args todo b lnf g // g is new
 tempvar t1 t2
 mlevel 't1' = 'b', eq(1)
 mlevel 't2' = 'b', eq(2)
 local t "$ML_y1"
 local d "$ML_y2"
 tempvar p M R
 quietly gen double 'p' = exp('t2')
 quietly gen double 'M' = ('t'*exp(-'t1'))^'p'
 quietly gen double 'R' = ln('t')-'t1'
 mlsum 'lnf' = -'M' + 'd'*('t2'-'t1' + ('p'-1)*'R')
 if ('todo'==0 | 'lnf'>=.) exit /* <-- new */
 tempname d1 d2 /* <-- new */
 mlvecsum 'lnf' 'd1' = 'p'*('M'-'d'), eq(1) /* <-- new */
 mlvecsum 'lnf' 'd2' = 'd' - 'R'*'p'*('M'-'d'), eq(2) /* <-- new */
 matrix 'g' = ('d1','d2') /* <-- new */
end

```

We obtained this code by starting with our method d0 evaluator and then adding the extra lines that method d1 requires. To fit our model using this evaluator, we could type

```

. ml model d1 weib1 (studytime died = drug2 drug3 age) /s
. ml maximize

```

but we recommend substituting method `d1debug` for method d1 and typing

```

. ml model d1debug weib1 (studytime died = drug2 drug3 age) /s
. ml maximize

```

Method `d1debug` will compare the derivatives we calculate with numerical derivatives and thus verify that our program is correct. Once we are certain the program is correct, then we would switch from method `d1debug` to method d1.  $\triangleleft$

### ► Example 5: Method d2

Method d2 evaluators are required to produce  $-\mathbf{H} = -\partial^2 \ln L / \partial \mathbf{b} \partial \mathbf{b}'$ , the negative Hessian matrix, as well as the gradient and log-likelihood value. `mlmatsum` will help calculate  $\partial^2 \ln L / \partial \mathbf{b} \partial \mathbf{b}'$  from the negative second derivatives with respect to  $\theta$ . For the Weibull model, these negative second derivatives are

$$\begin{aligned} -\frac{\partial^2 \ln \ell_j}{\partial \theta_{1j}^2} &= p_j^2 M_j \\ -\frac{\partial^2 \ln \ell_j}{\partial \theta_{1j} \partial \theta_{2j}} &= -p_j (M_j - d_j + R_j p_j M_j) \\ -\frac{\partial^2 \ln \ell_j}{\partial \theta_{2j}^2} &= p_j R_j (R_j p_j M_j + M_j - d_j) \end{aligned}$$

The method d2 evaluator is

```

program weib2
 version 11
 args todo b lnf g negH // negH added

 tempvar t1 t2
 mlevel 't1' = 'b', eq(1)
 mlevel 't2' = 'b', eq(2)

 local t "$ML_y1"
 local d "$ML_y2"

 tempvar p M R
 quietly gen double 'p' = exp('t2')
 quietly gen double 'M' = ('t'*exp(-'t1'))^'p'
 quietly gen double 'R' = ln('t')-'t1'

 mlsum 'lnf' = -'M' + 'd'*('t2'-'t1' + ('p'-1)*'R')
 if ('todo'==0 | 'lnf'>=.) exit

 tempname d1 d2
 mlvecsum 'lnf' 'd1' = 'p'*('M'-'d'), eq(1)
 mlvecsum 'lnf' 'd2' = 'd' - 'R'*'p'*('M'-'d'), eq(2)
 matrix 'g' = ('d1','d2')
 if ('todo'==1 | 'lnf'>=.) exit // new from here down

 tempname d11 d12 d22
 mlmatsum 'lnf' 'd11' = 'p'^2 * 'M', eq(1)
 mlmatsum 'lnf' 'd12' = -'p'*('M'-'d' + 'R'*'p'*'M'), eq(1,2)
 mlmatsum 'lnf' 'd22' = 'p'*'R'*('R'*'p'*'M' + 'M' - 'd'), eq(2)
 matrix 'negH' = ('d11','d12' \ 'd12','d22')

end

```

We started with our previous method d1 evaluator and added the lines that method d2 requires. We could now fit a model by typing

```

. ml model d2 weib2 (studytime died = drug2 drug3 age) /s
. ml maximize

```

but we would recommend substituting method d2debug for method d2 and typing

```

. ml model d2debug weib2 (studytime died = drug2 drug3 age) /s
. ml maximize

```

Method d2debug will compare the first and second derivatives we calculate with numerical derivatives and thus verify that our program is correct. Once we are certain the program is correct, then we would switch from method d2debug to method d2.

As we stated earlier, to produce the robust variance estimator with method lf, there is nothing to do except specify `vce(robust)`, `vce(cluster clustvar)`, or `pweight`. For method d0, these options do not work. For methods d1 and d2, these options will work if your likelihood function meets the linear-form restrictions and you fill in the equation scores. The equation scores are defined as

$$\frac{\partial \ln \ell_j}{\partial \theta_{1j}}, \frac{\partial \ln \ell_j}{\partial \theta_{2j}}, \dots$$

Your evaluator will be passed variables, one for each equation, which you fill in with the equation scores. For *both* method d1 and d2, these variables are passed in the sixth and subsequent positions of the argument list. That is, you must process the arguments as

```
args todo b lnf g negH g1 g2 ...
```

Note that for method d1, the ‘negH’ argument is not used; it is merely a placeholder.

### ► Example 6: Robust variance estimates

If you have used `mlvecsum` in your method d1 or d2 evaluator, it is easy to turn it into a program that allows the computation of the robust variance estimator. The expression that you specified on the right-hand side of `mlvecsum` is the equation score.

Here we turn the program that we gave earlier in the method d1 example into one that allows `vce(robust)`, `vce(cluster clustvar)`, or `pweight`.

```
program weib1
 version 11
 args todo b lnf g negH g1 g2 // negH, g1, and g2 are new
 tempvar t1 t2
 mlevel 't1' = 'b', eq(1)
 mlevel 't2' = 'b', eq(2)
 local t "$ML_y1"
 local d "$ML_y2"
 tempvar p M R
 quietly gen double 'p' = exp('t2')
 quietly gen double 'M' = ('t'*exp(-'t1'))^'p'
 quietly gen double 'R' = ln('t')-'t1'
 mlsum 'lnf' = -'M' + 'd'*('t2'-'t1' + ('p'-1)*'R')
 if ('todo'==0 | 'lnf'>=.) exit
 tempname d1 d2
 quietly replace 'g1' = 'p'*('M'-'d') /* <-- new */
 quietly replace 'g2' = 'd' - 'R'*'p'*('M'-'d') /* <-- new */
 mlvecsum 'lnf' 'd1' = 'g1', eq(1) /* <-- changed */
 mlvecsum 'lnf' 'd2' = 'g2', eq(2) /* <-- changed */
 matrix 'g' = ('d1','d2')
end
```

To fit our model and get the robust variance estimates, we type

```
. ml model d1 weib1 (studytime died = drug2 drug3 age) /s, vce(robust)
. ml maximize
```

## Survey options and ml

ml can handle stratification, poststratification, multiple stages of clustering, and finite population corrections. Specifying the svy option implies that the data come from a survey design and also implies that the survey linearized variance estimator is to be used; see [SVY] [variance estimation](#).

### ► Example 7

Suppose that we are interested in a probit analysis of data from a survey in which q1 is the answer to a yes/no question and d1, d2, d3 are demographic responses. The following is a d2 evaluator for the probit model that meets the requirements for vce(robust) (linear form and computes the scores).

```

program myd2probit
 version 11
 args todo b lnf g negH g1
 tempvar z Fz lnfj
 mlevel 'z' = 'b'
 quietly gen double 'Fz' = normal('z') if $ML_y1 == 1
 quietly replace 'Fz' = normal(-'z') if $ML_y1 == 0
 quietly gen double 'lnfj' = log('Fz')
 mlsun 'lnf' = 'lnfj'
 if ('todo'==0 | 'lnf' >= .) exit
 quietly replace 'g1' = normalden('z')/'Fz' if $ML_y1 == 1
 quietly replace 'g1' = -normalden('z')/'Fz' if $ML_y1 == 0
 mlevelsum 'lnf' 'g' = 'g1', eq(1)
 if ('todo'==1 | 'lnf' >= .) exit
 mlmatsum 'lnf' 'negH' = 'g1'*('g1'+ 'z'), eq(1,1)
end

```

To fit a model, we svyset the data, then use svy with ml.

```

. svyset psuid [pw=w], strata(strid)
. ml model d2 myd2probit (q1 = d1 d2 d3), svy
. ml maximize

```

We could also use the subpop() option to make inferences about the subpopulation identified by the variable sub:

```

. svyset psuid [pw=w], strata(strid)
. ml model d2 myd2probit (q1 = d1 d2 d3), svy subpop(sub)
. ml maximize

```

◀

## Saved results

For results saved by ml without the svy option, see [R] [maximize](#).

For results saved by ml with the svy option, see [SVY] [svy](#).

## Methods and formulas

ml is implemented using moptimize; see [M-5] [moptimize\(\)](#).



## References

- Gould, W. W., J. S. Pitblado, and W. M. Sribney. 2006. *Maximum Likelihood Estimation with Stata*. 3rd ed. College Station, TX: Stata Press.
- Korn, E. L., and B. I. Graubard. 1990. Simultaneous testing of regression coefficients with complex survey data: Use of Bonferroni  $t$  statistics. *American Statistician* 44: 270–276.
- Royston, P. 2007. Profile likelihood for estimation and confidence intervals. *Stata Journal* 7: 376–387.

## Also see

- [R] [maximize](#) — Details of iterative maximization
- [R] [nl](#) — Nonlinear least-squares estimation
- [M-5] [moptimize\(\)](#) — Model optimization
- [M-5] [optimize\(\)](#) — Function optimization

Syntax

```
mlogit depvar [indepvars] [if] [in] [weight] [, options]
```

| options                              | description                                                                                                                                       |
|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                |                                                                                                                                                   |
| <code>noconstant</code>              | suppress constant term                                                                                                                            |
| <code>baseoutcome(#)</code>          | value of <i>depvar</i> that will be the base outcome                                                                                              |
| <code>constraints(clist)</code>      | apply specified linear constraints; <i>clist</i> has the form <code>#[-#] [ , #[-#] ... ]</code>                                                  |
| <code>collinear</code>               | keep collinear variables                                                                                                                          |
| SE/Robust                            |                                                                                                                                                   |
| <code>vce(vctype)</code>             | <i>vcetype</i> may be <code>oim</code> , <code>robust</code> , <code>cluster clustvar</code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                            |                                                                                                                                                   |
| <code>level(#)</code>                | set confidence level; default is <code>level(95)</code>                                                                                           |
| <code>rrr</code>                     | report relative-risk ratios                                                                                                                       |
| <code>nocnsreport</code>             | do not display constraints                                                                                                                        |
| <code>display_options</code>         | control spacing and display of omitted variables and base and empty cells                                                                         |
| Maximization                         |                                                                                                                                                   |
| <code>maximize_options</code>        | control the maximization process; seldom used                                                                                                     |
| <sup>†</sup> <code>coeflegend</code> | display coefficients' legend instead of coefficient table                                                                                         |

<sup>†</sup>`coeflegend` does not appear in the dialog box.

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

`bootstrap`, `by`, `frapoly`, `jackknife`, `mfp`, `mi estimate`, `rolling`, `statsby`, and `svy` are allowed; see [U] 11.1.10 Prefix commands.

`vce(bootstrap)` and `vce(jackknife)` are not allowed with the `mi estimate` prefix.

Weights are not allowed with the `bootstrap` prefix.

`vce()` and weights are not allowed with the `svy` prefix.

`fweights`, `iweights`, and `pweights` are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Categorical outcomes > Multinomial logistic regression

## Description

`mlogit` fits maximum-likelihood multinomial logit models, also known as polytomous logistic regression. You can define constraints to perform constrained estimation. Some people refer to conditional logistic regression as multinomial logit. If you are one of them, see [R] [clogit](#).

See [R] [logistic](#) for a list of related estimation commands.

## Options

### Model

`noconstant`; see [R] [estimation options](#).

`baseoutcome(#)` specifies the value of *depvar* to be treated as the base outcome. The default is to choose the most frequent outcome.

`constraints(clist)`, `collinear`; see [R] [estimation options](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

If specifying `vce(bootstrap)` or `vce(jackknife)`, you must also specify `baseoutcome()`.

### Reporting

`level(#)`; see [R] [estimation options](#).

`rrr` reports the estimated coefficients transformed to relative-risk ratios, i.e.,  $e^b$  rather than  $b$ ; see [Description of the model](#) below for an explanation of this concept. Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated. `rrr` may be specified at estimation or when replaying previously estimated results.

`nocnsreport`; see [R] [estimation options](#).

*display\_options*: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

### Maximization

*maximize\_options*: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [R] [maximize](#). These options are seldom used.

The following option is available with `mlogit` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

Remarks are presented under the following headings:

*Description of the model*  
*Fitting unconstrained models*  
*Fitting constrained models*

`mlogit` fits maximum likelihood models with discrete dependent (left-hand-side) variables when the dependent variable takes on more than two outcomes and the outcomes have no natural ordering. If the dependent variable takes on only two outcomes, estimates are identical to those produced by `logistic` or `logit`; see [R] [logistic](#) or [R] [logit](#). If the outcomes are ordered, see [R] [ologit](#).

## Description of the model

For an introduction to multinomial logit models, see [Greene \(2008, 843–845\)](#), [Hosmer Jr. and Lemeshow \(2000, 260–287\)](#), [Long \(1997, chap. 6\)](#), [Long and Freese \(2006, chap. 6 and 7\)](#), and [Treiman \(2009, 336–341\)](#). For a description emphasizing the difference in assumptions and data requirements for conditional and multinomial logit, see [Davidson and MacKinnon \(1993\)](#).

Consider the outcomes  $1, 2, 3, \dots, m$  recorded in  $y$ , and the explanatory variables  $X$ . Assume that there are  $m = 3$  outcomes: “buy an American car”, “buy a Japanese car”, and “buy a European car”. The values of  $y$  are then said to be “unordered”. Even though the outcomes are coded 1, 2, and 3, the numerical values are arbitrary because  $1 < 2 < 3$  does not imply that outcome 1 (buy American) is less than outcome 2 (buy Japanese) is less than outcome 3 (buy European). This unordered categorical property of  $y$  distinguishes the use of `mlogit` from `regress` (which is appropriate for a continuous dependent variable), from `ologit` (which is appropriate for ordered categorical data), and from `logit` (which is appropriate for two outcomes, which can be thought of as ordered).

In the multinomial logit model, you estimate a set of coefficients,  $\beta^{(1)}$ ,  $\beta^{(2)}$ , and  $\beta^{(3)}$ , corresponding to each outcome:

$$\begin{aligned}\Pr(y = 1) &= \frac{e^{X\beta^{(1)}}}{e^{X\beta^{(1)}} + e^{X\beta^{(2)}} + e^{X\beta^{(3)}}} \\ \Pr(y = 2) &= \frac{e^{X\beta^{(2)}}}{e^{X\beta^{(1)}} + e^{X\beta^{(2)}} + e^{X\beta^{(3)}}} \\ \Pr(y = 3) &= \frac{e^{X\beta^{(3)}}}{e^{X\beta^{(1)}} + e^{X\beta^{(2)}} + e^{X\beta^{(3)}}}\end{aligned}$$

The model, however, is unidentified in the sense that there is more than one solution to  $\beta^{(1)}$ ,  $\beta^{(2)}$ , and  $\beta^{(3)}$  that leads to the same probabilities for  $y = 1$ ,  $y = 2$ , and  $y = 3$ . To identify the model, you arbitrarily set one of  $\beta^{(1)}$ ,  $\beta^{(2)}$ , or  $\beta^{(3)}$  to 0—it does not matter which. That is, if you arbitrarily set  $\beta^{(1)} = 0$ , the remaining coefficients  $\beta^{(2)}$  and  $\beta^{(3)}$  will measure the change relative to the  $y = 1$  group. If you instead set  $\beta^{(2)} = 0$ , the remaining coefficients  $\beta^{(1)}$  and  $\beta^{(3)}$  will measure the change relative to the  $y = 2$  group. The coefficients will differ because they have different interpretations, but the predicted probabilities for  $y = 1, 2$ , and  $3$  will still be the same. Thus either parameterization will be a solution to the same underlying model.

Setting  $\beta^{(1)} = 0$ , the equations become

$$\begin{aligned}\Pr(y = 1) &= \frac{1}{1 + e^{X\beta^{(2)}} + e^{X\beta^{(3)}}} \\ \Pr(y = 2) &= \frac{e^{X\beta^{(2)}}}{1 + e^{X\beta^{(2)}} + e^{X\beta^{(3)}}} \\ \Pr(y = 3) &= \frac{e^{X\beta^{(3)}}}{1 + e^{X\beta^{(2)}} + e^{X\beta^{(3)}}}\end{aligned}$$

The relative probability of  $y = 2$  to the base outcome is

$$\frac{\Pr(y = 2)}{\Pr(y = 1)} = e^{X\beta^{(2)}}$$

Let's call this ratio the relative risk, and let's further assume that  $X$  and  $\beta_k^{(2)}$  are vectors equal to  $(x_1, x_2, \dots, x_k)$  and  $(\beta_1^{(2)}, \beta_2^{(2)}, \dots, \beta_k^{(2)})'$ , respectively. The ratio of the relative risk for a one-unit change in  $x_i$  is then

$$\frac{e^{\beta_1^{(2)}x_1 + \dots + \beta_i^{(2)}(x_i+1) + \dots + \beta_k^{(2)}x_k}}{e^{\beta_1^{(2)}x_1 + \dots + \beta_i^{(2)}x_i + \dots + \beta_k^{(2)}x_k}} = e^{\beta_i^{(2)}}$$

Thus the exponentiated value of a coefficient is the relative-risk ratio for a one-unit change in the corresponding variable (risk is measured as the risk of the outcome relative to the base outcome).

## Fitting unconstrained models

### ► Example 1

We have data on the type of health insurance available to 616 psychologically depressed subjects in the United States (Tarlov et al. 1989; Wells et al. 1989). The insurance is categorized as either an indemnity plan (i.e., regular fee-for-service insurance, which may have a deductible or coinsurance rate) or a prepaid plan (a fixed up-front payment allowing subsequent unlimited use as provided, for instance, by an HMO). The third possibility is that the subject has no insurance whatsoever. We wish to explore the demographic factors associated with each subject's insurance choice. One of the demographic factors in our data is the race of the participant, coded as white or nonwhite:

```
. use http://www.stata-press.com/data/r11/sysdsn1
(Health insurance data)
. tabulate insure nonwhite, chi2 col
```

| Key       |                          |               |               |
|-----------|--------------------------|---------------|---------------|
|           | <i>frequency</i>         |               |               |
|           | <i>column percentage</i> |               |               |
| insure    | nonwhite                 |               | Total         |
|           | 0                        | 1             |               |
| Indemnity | 251<br>50.71             | 43<br>35.54   | 294<br>47.73  |
| Prepaid   | 208<br>42.02             | 69<br>57.02   | 277<br>44.97  |
| Uninsure  | 36<br>7.27               | 9<br>7.44     | 45<br>7.31    |
| Total     | 495<br>100.00            | 121<br>100.00 | 616<br>100.00 |

Pearson chi2(2) = 9.5599 Pr = 0.008

Although `insure` appears to take on the values `Indemnity`, `Prepaid`, and `Uninsure`, it actually takes on the values 1, 2, and 3. The words appear because we have associated a value label with the numeric variable `insure`; see [U] 12.6.3 Value labels.

When we fit a multinomial logit model, we can tell `mlogit` which outcome to use as the base outcome, or we can let `mlogit` choose. To fit a model of `insure` on `nonwhite`, letting `mlogit` choose the base outcome, we type

```
. mlogit insure nonwhite
Iteration 0: log likelihood = -556.59502
Iteration 1: log likelihood = -551.78935
Iteration 2: log likelihood = -551.78348
Iteration 3: log likelihood = -551.78348

Multinomial logistic regression Number of obs = 616
 LR chi2(2) = 9.62
 Prob > chi2 = 0.0081
Log likelihood = -551.78348 Pseudo R2 = 0.0086
```

| insure    | Coef.          | Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|-----------|----------------|-----------|--------|-------|----------------------|-----------|
| Indemnity | (base outcome) |           |        |       |                      |           |
| Prepaid   |                |           |        |       |                      |           |
| nonwhite  | .6608212       | .2157321  | 3.06   | 0.002 | .2379942             | 1.083648  |
| _cons     | -.1879149      | .0937644  | -2.00  | 0.045 | -.3716896            | -.0041401 |
| Uninsure  |                |           |        |       |                      |           |
| nonwhite  | .3779586       | .407589   | 0.93   | 0.354 | -.4209011            | 1.176818  |
| _cons     | -1.941934      | .1782185  | -10.90 | 0.000 | -2.291236            | -1.592632 |

`mlogit` chose the `indemnity` outcome as the base outcome and presented coefficients for the outcomes `prepaid` and `uninsured`. According to the model, the probability of `prepaid` for whites (`nonwhite = 0`) is

$$\Pr(\text{insure} = \text{Prepaid}) = \frac{e^{-.188}}{1 + e^{-.188} + e^{-1.942}} = 0.420$$

Similarly, for nonwhites, the probability of `prepaid` is

$$\Pr(\text{insure} = \text{Prepaid}) = \frac{e^{-.188+.661}}{1 + e^{-.188+.661} + e^{-1.942+.378}} = 0.570$$

These results agree with the column percentages presented by `tabulate` because the `mlogit` model is fully saturated. That is, there are enough terms in the model to fully explain the column percentage in each cell. The model chi-squared and the `tabulate` chi-squared are in almost perfect agreement; both test that the column percentages of `insure` are the same for both values of `nonwhite`.



► Example 2

By specifying the `baseoutcome()` option, we can control which outcome of the dependent variable is treated as the base. Left to its own, `mlogit` chose to make outcome 1, `indemnity`, the base outcome. To make outcome 2, `prepaid`, the base, we would type

```
. mlogit insure nonwhite, base(2)
Iteration 0: log likelihood = -556.59502
Iteration 1: log likelihood = -551.78935
Iteration 2: log likelihood = -551.78348
Iteration 3: log likelihood = -551.78348
Multinomial logistic regression
Log likelihood = -551.78348
```

|               |   |        |
|---------------|---|--------|
| Number of obs | = | 616    |
| LR chi2(2)    | = | 9.62   |
| Prob > chi2   | = | 0.0081 |
| Pseudo R2     | = | 0.0086 |

| insure                | Coef.          | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-----------------------|----------------|-----------|-------|-------|----------------------|-----------|
| Indemnity<br>nonwhite | -.6608212      | .2157321  | -3.06 | 0.002 | -1.083648            | -.2379942 |
| _cons                 | .1879149       | .0937644  | 2.00  | 0.045 | .0041401             | .3716896  |
| Prepaid               | (base outcome) |           |       |       |                      |           |
| Uninsure<br>nonwhite  | -.2828627      | .3977302  | -0.71 | 0.477 | -1.0624              | .4966742  |
| _cons                 | -1.754019      | .1805145  | -9.72 | 0.000 | -2.107821            | -1.400217 |

The `baseoutcome()` option requires that we specify the numeric value of the outcome, so we could not type `base(Prepaid)`.

Although the coefficients now appear to be different, the summary statistics reported at the top are identical. With this parameterization, the probability of prepaid insurance for whites is

$$\Pr(\text{insure} = \text{Prepaid}) = \frac{1}{1 + e^{1.188} + e^{-1.754}} = 0.420$$

This is the same answer we obtained previously.

◀

### ► Example 3

By specifying `rrr`, which we can do at estimation time or when we redisplay results, we see the model in terms of relative-risk ratios:

```
. mlogit, rrr
Multinomial logistic regression
Log likelihood = -551.78348
```

|               |   |        |
|---------------|---|--------|
| Number of obs | = | 616    |
| LR chi2(2)    | = | 9.62   |
| Prob > chi2   | = | 0.0081 |
| Pseudo R2     | = | 0.0086 |

| insure                | RRR            | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-----------------------|----------------|-----------|-------|-------|----------------------|----------|
| Indemnity<br>nonwhite | .516427        | .1114099  | -3.06 | 0.002 | .3383588             | .7882073 |
| Prepaid               | (base outcome) |           |       |       |                      |          |
| Uninsure<br>nonwhite  | .7536233       | .2997387  | -0.71 | 0.477 | .3456255             | 1.643247 |

Looked at this way, the relative risk of choosing an indemnity over a prepaid plan is 0.516 for nonwhites relative to whites.

To illustrate, from the output and discussions of examples 1 and 2 we find that

$$\Pr(\text{insure} = \text{Indemnity} \mid \text{white}) = \frac{1}{1 + e^{-.188} + e^{-1.942}} = 0.507$$

and thus the relative risk of choosing indemnity over prepaid (for whites) is

$$\frac{\Pr(\text{insure} = \text{Indemnity} \mid \text{white})}{\Pr(\text{insure} = \text{Prepaid} \mid \text{white})} = \frac{0.507}{0.420} = 1.207$$

For nonwhites,

$$\Pr(\text{insure} = \text{Indemnity} \mid \text{not white}) = \frac{1}{1 + e^{-.188+.661} + e^{-1.942+.378}} = 0.355$$

and thus the relative risk of choosing indemnity over prepaid (for nonwhites) is

$$\frac{\Pr(\text{insure} = \text{Indemnity} \mid \text{not white})}{\Pr(\text{insure} = \text{Prepaid} \mid \text{not white})} = \frac{0.355}{0.570} = 0.623$$

The ratio of these two relative risks, hence the name “relative-risk ratio”, is  $0.623/1.207 = 0.516$ , as given in the output under the heading “RRR”. ◀

#### □ Technical note

In models where only two categories are considered, the `mlogit` model reduces to standard `logit`. Consequently the exponentiated regression coefficients, labeled as RRR within `mlogit`, are equal to the odds ratios as given when the `or` option is specified under `logit`.

As such, always referring to `mlogit`’s exponentiated coefficients as odds ratios may be tempting. However, the discussion in [example 3](#) demonstrates that doing so would be incorrect. In general `mlogit` models, the exponentiated coefficients are ratios of relative risks, not ratios of odds. □

#### ▷ Example 4

One of the advantages of `mlogit` over `tabulate` is that we can include continuous variables and multiple categorical variables in the model. In examining the data on insurance choice, we decide that we want to control for age, gender, and site of study (the study was conducted in three sites):



```
. mlogit insure age male nonwhite i.site
Iteration 0: log likelihood = -555.85446
Iteration 1: log likelihood = -534.67443
Iteration 2: log likelihood = -534.36284
Iteration 3: log likelihood = -534.36165
Iteration 4: log likelihood = -534.36165

Multinomial logistic regression Number of obs = 615
 LR chi2(10) = 42.99
 Prob > chi2 = 0.0000
Log likelihood = -534.36165 Pseudo R2 = 0.0387
```

| insure    | Coef.          | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-----------|----------------|-----------|-------|-------|----------------------|-----------|
| Indemnity | (base outcome) |           |       |       |                      |           |
| Prepaid   |                |           |       |       |                      |           |
| age       | -.011745       | .0061946  | -1.90 | 0.058 | -.0238862            | .0003962  |
| male      | .5616934       | .2027465  | 2.77  | 0.006 | .1643175             | .9590693  |
| nonwhite  | .9747768       | .2363213  | 4.12  | 0.000 | .5115955             | 1.437958  |
| site      |                |           |       |       |                      |           |
| 2         | .1130359       | .2101903  | 0.54  | 0.591 | -.2989296            | .5250013  |
| 3         | -.5879879      | .2279351  | -2.58 | 0.010 | -1.034733            | -.1412433 |
| _cons     | .2697127       | .3284422  | 0.82  | 0.412 | -.3740222            | .9134476  |
| Uninsure  |                |           |       |       |                      |           |
| age       | -.0077961      | .0114418  | -0.68 | 0.496 | -.0302217            | .0146294  |
| male      | .4518496       | .3674867  | 1.23  | 0.219 | -.268411             | 1.17211   |
| nonwhite  | .2170589       | .4256361  | 0.51  | 0.610 | -.6171725            | 1.05129   |
| site      |                |           |       |       |                      |           |
| 2         | -1.211563      | .4705127  | -2.57 | 0.010 | -2.133751            | -.2893747 |
| 3         | -.2078123      | .3662926  | -0.57 | 0.570 | -.9257327            | .510108   |
| _cons     | -1.286943      | .5923219  | -2.17 | 0.030 | -2.447872            | -.1260134 |

These results suggest that the inclination of nonwhites to choose prepaid care is even stronger than it was without controlling. We also see that subjects in site 2 are less likely to be uninsured.

4

## Fitting constrained models

mlogit can fit models with subsets of coefficients constrained to be zero, with subsets of coefficients constrained to be equal both within and across equations, and with subsets of coefficients arbitrarily constrained to equal linear combinations of other estimated coefficients.

Before fitting a constrained model, you define the constraints with the `constraint` command; see [\[R\] constraint](#). Once the constraints are defined, you estimate using `mlogit`, specifying the `constraint()` option. Typing `constraint(4)` would use the constraint you previously saved as 4. Typing `constraint(1,4,6)` would use the previously stored constraints 1, 4, and 6. Typing `constraint(1-4,6)` would use the previously stored constraints 1, 2, 3, 4, and 6.

Sometimes you will not be able to specify the constraints without knowing the omitted outcome. In such cases, assume that the omitted outcome is whatever outcome is convenient for you, and include the `baseoutcome()` option when you specify the `mlogit` command.

## ► Example 5

We can use constraints to test hypotheses, among other things. In our insurance-choice model, let's test the hypothesis that there is no distinction between having indemnity insurance and being uninsured. Indemnity-style insurance was the omitted outcome, so we type

```
. test [Uninsure]
(1) [Uninsure]age = 0
(2) [Uninsure]male = 0
(3) [Uninsure]nonwhite = 0
(4) [Uninsure]1b.site = 0
(5) [Uninsure]2.site = 0
(6) [Uninsure]3.site = 0
 Constraint 4 dropped
 chi2(5) = 9.31
 Prob > chi2 = 0.0973
```

If indemnity had not been the omitted outcome, we would have typed `test [Uninsure=Indemnity]`.

The results produced by `test` are an approximation based on the estimated covariance matrix of the coefficients. Because the probability of being uninsured is low, the log likelihood may be nonlinear for the uninsured. Conventional statistical wisdom is not to trust the asymptotic answer under these circumstances but to perform a likelihood-ratio test instead.

To use Stata's `lrtest` (likelihood-ratio test) command, we must fit both the unconstrained and constrained models. The unconstrained model is the one we have previously fit. Following the instruction in [\[R\] lrtest](#), we first save the unconstrained model results:

```
. estimates store unconstrained
```

To fit the constrained model, we must refit our model with all the coefficients except the constant set to 0 in the `Uninsure` equation. We define the constraint and then refit:

```
. constraint 1 [Uninsure]
. mlogit insure age male nonwhite i.site, constraints(1)
Iteration 0: log likelihood = -555.85446
Iteration 1: log likelihood = -539.80523
Iteration 2: log likelihood = -539.75644
Iteration 3: log likelihood = -539.75643

Multinomial logistic regression Number of obs = 615
 Wald chi2(5) = 29.70
Log likelihood = -539.75643 Prob > chi2 = 0.0000

(1) [Uninsure]age = 0
(2) [Uninsure]male = 0
(3) [Uninsure]nonwhite = 0
(4) [Uninsure]2.site = 0
(5) [Uninsure]3.site = 0
```

| insure    | Coef.          | Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|-----------|----------------|-----------|--------|-------|----------------------|-----------|
| Indemnity | (base outcome) |           |        |       |                      |           |
| Prepaid   |                |           |        |       |                      |           |
| age       | -.0107025      | .0060039  | -1.78  | 0.075 | -.0224699            | .0010649  |
| male      | .4963616       | .1939683  | 2.56   | 0.010 | .1161907             | .8765324  |
| nonwhite  | .9421369       | .2252094  | 4.18   | 0.000 | .5007346             | 1.383539  |
| site      |                |           |        |       |                      |           |
| 2         | .2530912       | .2029465  | 1.25   | 0.212 | -.1446767            | .6508591  |
| 3         | -.5521773      | .2187237  | -2.52  | 0.012 | -.9808678            | -.1234869 |
| _cons     | .1792752       | .3171372  | 0.57   | 0.572 | -.4423023            | .8008527  |
| Uninsure  |                |           |        |       |                      |           |
| age       | (omitted)      |           |        |       |                      |           |
| male      | (omitted)      |           |        |       |                      |           |
| nonwhite  | (omitted)      |           |        |       |                      |           |
| site      |                |           |        |       |                      |           |
| 2         | (omitted)      |           |        |       |                      |           |
| 3         | (omitted)      |           |        |       |                      |           |
| _cons     | -1.87351       | .1601099  | -11.70 | 0.000 | -2.18732             | -1.5597   |

We can now perform the likelihood-ratio test:

```
. lrtest unconstrained .
Likelihood-ratio test LR chi2(5) = 10.79
(Assumption: . nested in unconstrained) Prob > chi2 = 0.0557
```

The likelihood-ratio chi-squared is 10.79 with 5 degrees of freedom—just slightly greater than the magic  $p = 0.05$  level—so we should not call this difference significant.

❑ Technical note

In certain circumstances, you should fit a multinomial logit model with conditional logit; see [R] **clogit**. With substantial data manipulation, **clogit** can handle the same class of models with some interesting additions. For example, if we had available the price and deductible of the most competitive insurance plan of each type, **mlogit** could not use this information, but **clogit** could.

## Saved results

`mlogit` saves the following in `e()`:

### Scalars

|                            |                                                                       |
|----------------------------|-----------------------------------------------------------------------|
| <code>e(N)</code>          | number of observations                                                |
| <code>e(k_out)</code>      | number of outcomes                                                    |
| <code>e(k)</code>          | number of parameters                                                  |
| <code>e(k_eq)</code>       | number of equations in <code>e(b)</code>                              |
| <code>e(k_eq_model)</code> | number of equations in model Wald test                                |
| <code>e(k_dv)</code>       | number of dependent variables                                         |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints                        |
| <code>e(df_m)</code>       | model degrees of freedom                                              |
| <code>e(r2_p)</code>       | pseudo- <i>R</i> -squared                                             |
| <code>e(ll)</code>         | log likelihood                                                        |
| <code>e(ll_0)</code>       | log likelihood, constant-only model                                   |
| <code>e(N_clust)</code>    | number of clusters                                                    |
| <code>e(chi2)</code>       | $\chi^2$                                                              |
| <code>e(p)</code>          | significance                                                          |
| <code>e(k_iform)</code>    | number of leading equations appropriate for <code>iform</code> output |
| <code>e(k_eq_base)</code>  | equation number of the base outcome                                   |
| <code>e(baseout)</code>    | the value of <i>devar</i> to be treated as the base outcome           |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                                             |
| <code>e(ic)</code>         | number of iterations                                                  |
| <code>e(rc)</code>         | return code                                                           |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                                           |

### Macros

|                                 |                                                                          |
|---------------------------------|--------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>mlogit</code>                                                      |
| <code>e(cmdline)</code>         | command as typed                                                         |
| <code>e(devar)</code>           | name of dependent variable                                               |
| <code>e(wtype)</code>           | weight type                                                              |
| <code>e(wexp)</code>            | weight expression                                                        |
| <code>e(title)</code>           | title in estimation output                                               |
| <code>e(clustvar)</code>        | name of cluster variable                                                 |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                  |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                           |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                            |
| <code>e(eqnames)</code>         | names of equations                                                       |
| <code>e(baselab)</code>         | value label corresponding to base outcome                                |
| <code>e(k_eq_model_skip)</code> | equations to skip in most reports                                        |
| <code>e(opt)</code>             | type of optimization                                                     |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>       | type of <code>ml</code> method                                           |
| <code>e(user)</code>            | name of likelihood-evaluator program                                     |
| <code>e(technique)</code>       | maximization technique                                                   |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular              |
| <code>e(crittype)</code>        | optimization criterion                                                   |
| <code>e(properties)</code>      | b V                                                                      |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                           |
| <code>e(marginsnotok)</code>    | predictions disallowed by <code>margins</code>                           |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>           |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>           |

|                              |                                              |
|------------------------------|----------------------------------------------|
| Matrices                     |                                              |
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(out)</code>          | outcome values                               |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(gradient)</code>     | gradient vector                              |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |
| Functions                    |                                              |
| <code>e(sample)</code>       | marks estimation sample                      |

## Methods and formulas

`mlogit` is implemented as an ado-file.

The multinomial logit model is described in [Greene \(2008, 843–845\)](#).

Suppose that there are  $k$  categorical outcomes and—without loss of generality—let the base outcome be 1. The probability that the response for the  $j$ th observation is equal to the  $i$ th outcome is

$$p_{ij} = \Pr(y_j = i) = \begin{cases} \frac{1}{1 + \sum_{m=2}^k \exp(\mathbf{x}_j \boldsymbol{\beta}_m)}, & \text{if } i = 1 \\ \frac{\exp(\mathbf{x}_j \boldsymbol{\beta}_i)}{1 + \sum_{m=2}^k \exp(\mathbf{x}_j \boldsymbol{\beta}_m)}, & \text{if } i > 1 \end{cases}$$

where  $\mathbf{x}_j$  is the row vector of observed values of the independent variables for the  $j$ th observation and  $\boldsymbol{\beta}_m$  is the coefficient vector for outcome  $m$ . The log pseudolikelihood is

$$\ln L = \sum_j w_j \sum_{i=1}^k I_i(y_j) \ln p_{ik}$$

where  $w_j$  is an optional weight and

$$I_i(y_j) = \begin{cases} 1, & \text{if } y_j = i \\ 0, & \text{otherwise} \end{cases}$$

Newton–Raphson maximum likelihood is used; see [\[R\]](#) [maximize](#).

For constrained equations, the set of constraints is orthogonalized, and a subset of maximizable parameters is selected. For example, a parameter that is constrained to zero is not a maximizable parameter. If two parameters are constrained to be equal to each other, only one is a maximizable parameter.

Let  $\mathbf{r}$  be the vector of maximizable parameters.  $\mathbf{r}$  is physically a subset of the solution parameters,  $\mathbf{b}$ . A matrix,  $\mathbf{T}$ , and a vector,  $\mathbf{m}$ , are defined as

$$\mathbf{b} = \mathbf{T}\mathbf{r} + \mathbf{m}$$

so that

$$\frac{\partial f}{\partial \mathbf{b}} = \frac{\partial f}{\partial \mathbf{r}} \mathbf{T}'$$

$$\frac{\partial^2 f}{\partial \mathbf{b}^2} = \mathbf{T} \frac{\partial^2 f}{\partial \mathbf{r}^2} \mathbf{T}'$$

$\mathbf{T}$  consists of a block form in which one part is a permutation of the identity matrix and the other part describes how to calculate the constrained parameters from the maximizable parameters.

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

`mlogit` also supports estimation with survey data. For details on VCEs with survey data, see [\[SVY\] variance estimation](#).

## References

- Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.
- Freese, J., and J. S. Long. 2000. [sg155: Tests for the multinomial logit model](#). *Stata Technical Bulletin* 58: 19–25. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 247–255. College Station, TX: Stata Press.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- Haan, P., and A. Uhlenborff. 2006. [Estimation of multinomial logit models with unobserved heterogeneity using maximum simulated likelihood](#). *Stata Journal* 6: 229–245.
- Hamilton, L. C. 1993. [sqv8: Interpreting multinomial logistic regression](#). *Stata Technical Bulletin* 13: 24–28. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 176–181. College Station, TX: Stata Press.
- . 2009. *Statistics with Stata (Updated for Version 10)*. Belmont, CA: Brooks/Cole.
- Hendrickx, J. 2000. [sbe37: Special restrictions in multinomial logistic regression](#). *Stata Technical Bulletin* 56: 18–26. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 93–103. College Station, TX: Stata Press.
- Hole, A. R. 2007. [Fitting mixed logit models by using maximum simulated likelihood](#). *Stata Journal* 7: 388–401.
- Hosmer Jr., D. W., and S. Lemeshow. 2000. *Applied Logistic Regression*. 2nd ed. New York: Wiley.
- Kleinbaum, D. G., and M. Klein. 2002. *Logistic Regression: A Self-Learning Text*. 2nd ed. New York: Springer.
- Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.
- Long, J. S., and J. Freese. 2006. *Regression Models for Categorical Dependent Variables Using Stata*. 2nd ed. College Station, TX: Stata Press.
- Tarlov, A. R., J. E. Ware Jr., S. Greenfield, E. C. Nelson, E. Perrin, and M. Zubkoff. 1989. The medical outcomes study. An application of methods for monitoring the results of medical care. *Journal of the American Medical Association* 262: 925–930.
- Treiman, D. J. 2009. *Quantitative Data Analysis: Doing Social Research to Test Ideas*. San Francisco, CA: Jossey-Bass.
- Wells, K. B., R. D. Hays, M. A. Burnam, W. H. Rogers, S. Greenfield, and J. E. Ware Jr. 1989. Detection of depressive disorder for patients receiving prepaid or fee-for-service care. Results from the Medical Outcomes Survey. *Journal of the American Medical Association* 262: 3298–3302.
- Xu, J., and J. S. Long. 2005. [Confidence intervals for predicted outcomes in regression models for categorical outcomes](#). *Stata Journal* 5: 537–559.

## Also see

- [R] **mlogit postestimation** — Postestimation tools for mlogit
- [R] **clogit** — Conditional (fixed-effects) logistic regression
- [R] **logistic** — Logistic regression, reporting odds ratios
- [R] **logit** — Logistic regression, reporting coefficients
- [R] **mlogit** — Multinomial (polytomous) logistic regression
- [R] **nlogit** — Nested logit regression
- [R] **ologit** — Ordered logistic regression
- [R] **rologit** — Rank-ordered logistic regression
- [R] **slogit** — Stereotype logistic regression
- [SVY] **svy estimation** — Estimation commands for survey data
- [U] **20 Estimation and postestimation commands**

Description

The following postestimation commands are available for `mlogit`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>hausman</code>             | Hausman's specification test                                                                        |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Syntax for predict

| <code>predict</code> <a href="#">[type]</a> { <i>stub*</i> <a href="#">newvar</a> <a href="#">newvarlist</a> } <a href="#">[if]</a> <a href="#">[in]</a> [ , <i>statistic</i> <code>outcome(outcome)</code> ] |                                                            |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|
| <code>predict</code> <a href="#">[type]</a> { <i>stub*</i> <a href="#">newvarlist</a> } <a href="#">[if]</a> <a href="#">[in]</a> , <code>scores</code>                                                       |                                                            |
| <i>statistic</i>                                                                                                                                                                                              | description                                                |
| Main                                                                                                                                                                                                          |                                                            |
| <code>pr</code>                                                                                                                                                                                               | probability of a positive outcome; the default             |
| <code>xb</code>                                                                                                                                                                                               | linear prediction                                          |
| <code>stdp</code>                                                                                                                                                                                             | standard error of the linear prediction                    |
| <code>stddp</code>                                                                                                                                                                                            | standard error of the difference in two linear predictions |



If you do not specify `outcome()`, `pr` (with one new variable specified), `xb`, and `stdp` assume `outcome(#1)`. You must specify `outcome()` with the `stddp` option.

You specify one or  $k$  new variables with `pr`, where  $k$  is the number of outcomes.

You specify one new variable with `xb`, `stdp`, and `stddp`.

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

### Main

`pr`, the default, calculates the probability of each of the categories of the dependent variable or the probability of the level specified in `outcome(outcome)`. If you specify the `outcome(outcome)` option, you need to specify only one new variable; otherwise, you must specify a new variable for each category of the dependent variable.

`xb` calculates the linear prediction. You must also specify the `outcome(outcome)` option.

`stdp` calculates the standard error of the linear prediction. You must also specify the `outcome(outcome)` option.

`stddp` calculates the standard error of the difference in two linear predictions. You must specify the `outcome(outcome)` option, and here you specify the two particular outcomes of interest inside the parentheses, for example, `predict sed, stddp outcome(1,3)`.

`outcome(outcome)` specifies the outcome for which the statistic is to be calculated. `equation()` is a synonym for `outcome()`: it does not matter which you use. `outcome()` or `equation()` can be specified using

`#1, #2, ...`, where `#1` means the first category of the dependent variable, `#2` means the second category, etc.;

the values of the dependent variable; or

the value labels of the dependent variable if they exist.

`scores` calculates equation-level score variables. The number of score variables created will be one less than the number of outcomes in the model. If the number of outcomes in the model were  $k$ , then

the first new variable will contain  $\partial \ln L / \partial (\mathbf{x}_j \beta_1)$ ;

the second new variable will contain  $\partial \ln L / \partial (\mathbf{x}_j \beta_2)$ ;

...

the  $(k - 1)$ th new variable will contain  $\partial \ln L / \partial (\mathbf{x}_j \beta_{k-1})$ .

## Remarks

Remarks are presented under the following headings:

*Obtaining predicted values*

*Calculating marginal effects*

*Testing hypotheses about coefficients*

## Obtaining predicted values

### ► Example 1

After estimation, we can use `predict` to obtain predicted probabilities, index values, and standard errors of the index, or differences in the index. For instance, in [example 4](#) of [\[R\] mlogit](#), we fit a model of insurance choice on various characteristics. We can obtain the predicted probabilities for outcome 1 by typing

```
. use http://www.stata-press.com/data/r11/sysdsn1
(Health insurance data)

. mlogit insure age i.male i.nonwhite i.site
(output omitted)

. predict p1 if e(sample), outcome(1)
(option pr assumed; predicted probability)
(29 missing values generated)

. summarize p1
```

| Variable | Obs | Mean     | Std. Dev. | Min      | Max    |
|----------|-----|----------|-----------|----------|--------|
| p1       | 615 | .4764228 | .1032279  | .1698142 | .71939 |

We added the `i.` prefix to the `male`, `nonwhite`, and `site` variables to explicitly identify them as factor variables. That makes no difference in the estimated results, but we will take advantage of it in later examples. We also included `if e(sample)` to restrict the calculation to the estimation sample. In [example 4](#) of [\[R\] mlogit](#), the multinomial logit model was fit on 615 observations, so there must be missing values in our dataset.

Although we typed `outcome(1)`, specifying 1 for the indemnity outcome, we could have typed `outcome(Indemnity)`. For instance, to obtain the probabilities for prepaid, we could type

```
. predict p2 if e(sample), outcome(prepaid)
(option pr assumed; predicted probability)
outcome prepaid not found
r(303);

. predict p2 if e(sample), outcome(Prepaid)
(option pr assumed; predicted probability)
(29 missing values generated)

. summarize p2
```

| Variable | Obs | Mean     | Std. Dev. | Min      | Max      |
|----------|-----|----------|-----------|----------|----------|
| p2       | 615 | .4504065 | .1125962  | .1964103 | .7885724 |

We must specify the label exactly as it appears in the underlying value label (or how it appears in the `mlogit` output), including capitalization.

Here we have used `predict` to obtain probabilities for the same sample on which we estimated. That is not necessary. We could use another dataset that had the independent variables defined (in our example, `age`, `male`, `nonwhite`, and `site`) and use `predict` to obtain predicted probabilities; here, we would not specify `if e(sample)`.

## ► Example 2

`predict` can also be used to obtain the index values—the  $\sum x_i \widehat{\beta}_i^{(k)}$ —as well as the probabilities:

```
. predict idx1, outcome(Indemnity) xb
(1 missing value generated)
. summarize idx1
```

| Variable | Obs | Mean | Std. Dev. | Min | Max |
|----------|-----|------|-----------|-----|-----|
| idx1     | 643 | 0    | 0         | 0   | 0   |

The indemnity outcome was our base outcome—the outcome for which all the coefficients were set to 0—so the index is always 0. For the prepaid and uninsured outcomes, we type

```
. predict idx2, outcome(Prepaid) xb
(1 missing value generated)
. predict idx3, outcome(Uninsure) xb
(1 missing value generated)
. summarize idx2 idx3
```

| Variable | Obs | Mean      | Std. Dev. | Min       | Max       |
|----------|-----|-----------|-----------|-----------|-----------|
| idx2     | 643 | -.0566113 | .4962973  | -1.298198 | 1.700719  |
| idx3     | 643 | -1.980747 | .6018139  | -3.112741 | -.8258458 |

We can obtain the standard error of the index by specifying the `stdp` option:

```
. predict se2, outcome(Prepaid) stdp
(1 missing value generated)
. list p2 idx2 se2 in 1/5
```

|    | p2       | idx2      | se2      |
|----|----------|-----------|----------|
| 1. | .3709022 | -.4831167 | .2437772 |
| 2. | .4977667 | .055111   | .1694686 |
| 3. | .4113073 | -.1712106 | .1793498 |
| 4. | .5424927 | .3788345  | .2513701 |
| 5. | .        | -.0925817 | .1452616 |

We obtained the probability, `p2`, in the previous example.

Finally, `predict` can calculate the standard error of the difference in the index values between two outcomes with the `stdpd` option:

```
. predict se_2_3, outcome(Prepaid,Uninsure) stdpd
(1 missing value generated)
. list idx2 idx3 se_2_3 in 1/5
```

|    | idx2      | idx3      | se_2_3   |
|----|-----------|-----------|----------|
| 1. | -.4831167 | -3.073253 | .5469354 |
| 2. | .055111   | -2.715986 | .4331918 |
| 3. | -.1712106 | -1.579621 | .3053815 |
| 4. | .3788345  | -1.462007 | .4492552 |
| 5. | -.0925817 | -2.814022 | .4024784 |

In the first observation, the difference in the indexes is  $-0.483 - (-3.073) = 2.59$ . The standard error of that difference is 0.547.

◀

### ► Example 3

It is more difficult to interpret the results from `mlogit` than those from `clogit` or `logit` because there are multiple equations. For example, suppose that one of the independent variables in our model takes on the values 0 and 1, and we are attempting to understand the effect of this variable. Assume that the coefficient on this variable for the second outcome,  $\beta^{(2)}$ , is positive. We might then be tempted to reason that the probability of the second outcome is higher if the variable is 1 rather than 0. Most of the time, that will be true, but occasionally we will be surprised. The probability of some other outcome could increase even more (say,  $\beta^{(3)} > \beta^{(2)}$ ), and thus the probability of outcome 2 would actually fall relative to that outcome. We can use `predict` to help interpret such results.

Continuing with our previously fit insurance-choice model, we wish to describe the model's predictions by race. For this purpose, we can use the method of predictive margins (also known as recycled predictions), in which we vary characteristics of interest across the whole dataset and average the predictions. That is, we have data on both whites and nonwhites, and our individuals have other characteristics as well. We will first pretend that all the people in our data are white but hold their other characteristics constant. We then calculate the probabilities of each outcome. Next we will pretend that all the people in our data are nonwhite, still holding their other characteristics constant. Again we calculate the probabilities of each outcome. The difference in those two sets of calculated probabilities, then, is the difference due to race, holding other characteristics constant.

```
. gen byte nonwhold = nonwhite // save real race
. replace nonwhite = 0 // make everyone white
(126 real changes made)

. predict wpsind, outcome(Indemnity) // predict probabilities
(option pr assumed; predicted probability)
(1 missing value generated)

. predict wpp, outcome(Prepaid)
(option pr assumed; predicted probability)
(1 missing value generated)

. predict wpsnoi, outcome(Uninsure)
(option pr assumed; predicted probability)
(1 missing value generated)

. replace nonwhite=1 // make everyone nonwhite
(644 real changes made)

. predict npsind, outcome(Indemnity)
(option pr assumed; predicted probability)
(1 missing value generated)

. predict npp, outcome(Prepaid)
(option pr assumed; predicted probability)
(1 missing value generated)

. predict npsnoi, outcome(Uninsure)
(option pr assumed; predicted probability)
(1 missing value generated)

. replace nonwhite=nonwhold // restore real race
(518 real changes made)
```

```
. summarize wp* nwp*, sep(3)
```

| Variable | Obs | Mean     | Std. Dev. | Min      | Max      |
|----------|-----|----------|-----------|----------|----------|
| wpind    | 643 | .5141673 | .0872679  | .3092903 | .71939   |
| wpp      | 643 | .4082052 | .0993286  | .1964103 | .6502247 |
| wponi    | 643 | .0776275 | .0360283  | .0273596 | .1302816 |
| nwpind   | 643 | .3112809 | .0817693  | .1511329 | .535021  |
| nwp      | 643 | .630078  | .0979976  | .3871782 | .8278881 |
| nwpni    | 643 | .0586411 | .0287185  | .0209648 | .0933874 |

In [example 1](#) of [\[R\] mlogit](#), we presented a cross-tabulation of insurance type and race. Those values were unadjusted. The means reported above are the values adjusted for age, sex, and site. Combining the results gives

|           | Unadjusted |          | Adjusted |          |
|-----------|------------|----------|----------|----------|
|           | white      | nonwhite | white    | nonwhite |
| Indemnity | 0.51       | 0.36     | 0.52     | 0.31     |
| Prepaid   | 0.42       | 0.57     | 0.41     | 0.63     |
| Uninsured | 0.07       | 0.07     | 0.08     | 0.06     |

We find, for instance, after adjusting for age, sex, and site, that although 57% of nonwhites in our data had prepaid plans, 63% of nonwhites chose prepaid plans.

Computing predictive margins by hand was instructive, but we can compute these values more easily using the `margins` command (see [\[R\] margins](#)). The two margins for the indemnity outcome can be estimated by typing

```
. margins nonwhite, predict(outcome(Indemnity)) noesample
Predictive margins Number of obs = 643
Model VCE : OIM
Expression : Pr(insure==Indemnity), predict(outcome(Indemnity))
```

|          | Delta-method |           |       |       |                      |
|----------|--------------|-----------|-------|-------|----------------------|
|          | Margin       | Std. Err. | z     | P> z  | [95% Conf. Interval] |
| nonwhite |              |           |       |       |                      |
| 0        | .5141673     | .0223485  | 23.01 | 0.000 | .470365 .5579695     |
| 1        | .3112809     | .0418049  | 7.45  | 0.000 | .2293448 .393217     |

`margins` also estimates the standard errors and confidence intervals of the margins. By default, `margins` uses only the estimation sample. We added the `noesample` option so that `margins` would use the entire sample and produce results comparable to our earlier analysis.

The margins for the other two outcomes can be computed by typing

```
. margins nonwhite, predict(outcome(Prepaid)) noesample
(output omitted)
. margins nonwhite, predict(outcome(Uninsure)) noesample
(output omitted)
```

□ Technical note

You can use `predict` to classify predicted values and compare them with the observed outcomes to interpret a multinomial logit model. This is a variation on the notions of sensitivity and specificity for logistic regression. Here we will classify indemnity and prepaid as definitely predicting indemnity, definitely predicting prepaid, and ambiguous.

```
. predict indem, outcome(Indemnity) index // obtain indexes
(1 missing value generated)

. predict prepaid, outcome(Prepaid) index
(1 missing value generated)

. gen diff = prepaid-indem // obtain difference
(1 missing value generated)

. predict sediff, outcome(Indemnity,Prepaid) stddp // & its standard error
(1 missing value generated)

. gen type = 1 if diff/sediff < -1.96 // definitely indemnity
(504 missing values generated)

. replace type = 3 if diff/sediff > 1.96 // definitely prepaid
(100 real changes made)

. replace type = 2 if type>=. & diff/sediff < . // ambiguous
(404 real changes made)

. label def type 1 "Def Ind" 2 "Ambiguous" 3 "Def Prep"

. label values type type // label results

. tabulate insure type
```

| insure    | type    |           |          | Total |
|-----------|---------|-----------|----------|-------|
|           | Def Ind | Ambiguous | Def Prep |       |
| Indemnity | 78      | 183       | 33       | 294   |
| Prepaid   | 44      | 177       | 56       | 277   |
| Uninsure  | 12      | 28        | 5        | 45    |
| Total     | 134     | 388       | 94       | 616   |

We can see that the predictive power of this model is modest. There are many misclassifications in both directions, though there are more correctly classified observations than misclassified observations.

Also the uninsured look overwhelmingly as though they might have come from the indemnity system rather than from the prepaid system.



Calculating marginal effects

▷ Example 4

We have already noted that the coefficients from multinomial logit can be difficult to interpret because they are relative to the base outcome. Another way to evaluate the effect of covariates is to examine the marginal effect of changing their values on the probability of observing an outcome.

The `margins` command can be used for this too. We can estimate the marginal effect of each covariate on the probability of observing the first outcome—indemnity insurance—by typing

```
. margins, dydx(*) predict(outcome(Indemnity))
Average marginal effects Number of obs = 615
Model VCE : OIM
Expression : Pr(insure==Indemnity), predict(outcome(Indemnity))
dy/dx w.r.t. : age 1.male 1.nonwhite 2.site 3.site
```

|            | Delta-method |           | z     | P> z  | [95% Conf. Interval] |           |
|------------|--------------|-----------|-------|-------|----------------------|-----------|
|            | dy/dx        | Std. Err. |       |       |                      |           |
| age        | .0026655     | .001399   | 1.91  | 0.057 | -.0000765            | .0054074  |
| 1.male     | -.1295734    | .0450943  | -2.87 | 0.004 | -.2179566            | -.0411903 |
| 1.nonwhite | -.2032404    | .0482554  | -4.21 | 0.000 | -.2978192            | -.1086616 |
| site       |              |           |       |       |                      |           |
| 2          | .0070995     | .0479993  | 0.15  | 0.882 | -.0869775            | .1011765  |
| 3          | .1216165     | .0505835  | 2.40  | 0.016 | .0224747             | .2207583  |

Note: dy/dx for factor levels is the discrete change from the base level.

By default, `margins` estimates the average marginal effect over the estimation sample, and that is what we see above. Being male decreases the average probability of having indemnity insurance by 0.130. We also see, from the note at the bottom of the table, that the marginal effect was computed as a discrete change in the probability of being male rather than female. That is why we made `male` a factor variable when fitting the model.

The `dydx(*)` option requested that `margins` estimate the marginal effect for each regressor, `dydx(age)` would have produced estimates only for the effect of `age`. `margins` has many options for controlling how the marginal effect is computed, including the ability to average over subgroups or to compute estimates for specified values of the regressors; see [\[R\] margins](#).

We could evaluate the marginal effects on the other two outcomes by typing

```
. margins, dydx(*) predict(outcome(Prepaid))
(output omitted)
. margins, dydx(*) predict(outcome(Uninsure))
(output omitted)
```

◀

## Testing hypotheses about coefficients

### ► Example 5

`test` tests hypotheses about the coefficients just as after any estimation command; see [\[R\] test](#). Note, however, `test`'s syntax for dealing with multiple-equation models. Because `test` bases its results on the estimated covariance matrix, we might prefer a likelihood-ratio test; see [example 5](#) in [\[R\] mlogit](#) for an example of `lrtest`.

If we simply list variables after the `test` command, we are testing that the corresponding coefficients are zero across all equations:

(Continued on next page)

```
. test 2.site 3.site
(1) [Indemnity]2.site = 0
(2) [Prepaid]2.site = 0
(3) [Uninsure]2.site = 0
(4) [Indemnity]3.site = 0
(5) [Prepaid]3.site = 0
(6) [Uninsure]3.site = 0
 Constraint 1 dropped
 Constraint 4 dropped
 chi2(4) = 19.74
 Prob > chi2 = 0.0006
```

We can test that all the coefficients (except the constant) in an equation are zero by simply typing the outcome in square brackets:

```
. test [Uninsure]
(1) [Uninsure]age = 0
(2) [Uninsure]male = 0
(3) [Uninsure]nonwhite = 0
(4) [Uninsure]1b.site = 0
(5) [Uninsure]2.site = 0
(6) [Uninsure]3.site = 0
 Constraint 4 dropped
 chi2(5) = 9.31
 Prob > chi2 = 0.0973
```

We specify the outcome just as we do with `predict`; we can specify the label if the outcome variable is labeled, or we can specify the numeric value of the outcome. We would have obtained the same test as above if we had typed `test [3]` because 3 is the value of `insure` for the outcome `uninsured`.

We can combine the two syntaxes. To test that the coefficients on the site variables are 0 in the equation corresponding to the outcome `prepaid`, we can type

```
. test [Prepaid]: 2.site 3.site
(1) [Prepaid]2.site = 0
(2) [Prepaid]3.site = 0
 chi2(2) = 10.78
 Prob > chi2 = 0.0046
```

We specified the outcome and then followed that with a colon and the variables we wanted to test.

We can also test that coefficients are equal across equations. To test that all coefficients except the constant are equal for the `prepaid` and `uninsured` outcomes, we can type

```
. test [Prepaid=Uninsure]
(1) [Prepaid]age - [Uninsure]age = 0
(2) [Prepaid]male - [Uninsure]male = 0
(3) [Prepaid]nonwhite - [Uninsure]nonwhite = 0
(4) [Prepaid]1b.site - [Uninsure]1b.site = 0
(5) [Prepaid]2.site - [Uninsure]2.site = 0
(6) [Prepaid]3.site - [Uninsure]3.site = 0
 Constraint 4 dropped
 chi2(5) = 13.80
 Prob > chi2 = 0.0169
```



To test that only the site variables are equal, we can type

```
. test [Prepaid=Uninsure]: 2.site 3.site
(1) [Prepaid]2.site - [Uninsure]2.site = 0
(2) [Prepaid]3.site - [Uninsure]3.site = 0
 chi2(2) = 12.68
 Prob > chi2 = 0.0018
```

Finally, we can test any arbitrary constraint by simply entering the equation and specifying the coefficients as described in [\[U\] 13.5 Accessing coefficients and standard errors](#). The following hypothesis is senseless but illustrates the point:

```
. test ([Prepaid]age+[Uninsure]2.site)/2 = 2-[Uninsure]nonwhite
(1) .5*[Prepaid]age + [Uninsure]nonwhite + .5*[Uninsure]2.site = 2
 chi2(1) = 22.45
 Prob > chi2 = 0.0000
```

See [\[R\] test](#) for more information about `test`. The information there about combining hypotheses across `test` commands (the `accum` option) also applies after `mlogit`.

◀

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[\[R\] mlogit](#) — Multinomial (polytomous) logistic regression

[\[U\] 20 Estimation and postestimation commands](#)

# Title

more — The —more— message

## Syntax

*Tell Stata to pause or not pause for —more— messages*

```
set more { on|off } [, permanently]
```

*Set number of lines between —more— messages*

```
set pagesize #
```

## Description

- set more on, which is the default, tells Stata to wait until you press a key before continuing when a —more— message is displayed.
- set more off tells Stata not to pause or display the —more— message.
- set pagesize # sets the number of lines between —more— messages. The permanently option is not allowed with set pagesize.

## Option

permanently specifies that, in addition to making the change right now, the more setting be remembered and become the default setting when you invoke Stata.

## Remarks

When you see —more— at the bottom of the screen,

| Press ...                       | and Stata ...                       |
|---------------------------------|-------------------------------------|
| letter <i>l</i> or <i>Enter</i> | displays the next line              |
| letter <i>q</i>                 | acts as if you pressed <i>Break</i> |
| Spacebar or any other key       | displays the next screen            |

You can also click on the **More** button or click on —more— to display the next screen.

- more— is Stata’s way of telling you that it has something more to show you but that showing it to you will cause the information on the screen to scroll off.
- If you type set more off, —more— conditions will never arise, and Stata’s output will scroll by at full speed.
- If you type set more on, —more— conditions will be restored at the appropriate places.
- Programmers should see [P] [more](#) for information on the more programming command.

## Also see

[R] [query](#) — Display system parameters

[P] [creturn](#) — Return c-class values

[P] [more](#) — Pause until key is pressed

[U] [7 —more— conditions](#)

Syntax

```
mprobit depvar [indepvars] [if] [in] [weight] [, options]
```

| <i>options</i>                               | description                                                                                                             |
|----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| Model                                        |                                                                                                                         |
| <u>no</u> constant                           | suppress constant terms                                                                                                 |
| <u>base</u> outcome( <i>#</i>   <i>lbl</i> ) | outcome used to normalize location                                                                                      |
| <u>probit</u> param                          | use the probit variance parameterization                                                                                |
| <u>con</u> straints( <i>constraints</i> )    | apply specified linear constraints                                                                                      |
| <u>col</u> linear                            | keep collinear variables                                                                                                |
| SE/Robust                                    |                                                                                                                         |
| vce( <i>vcetype</i> )                        | <i>vcetype</i> may be oim, <u>robust</u> , <u>cluster</u> <i>clustvar</i> , opg, <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting                                    |                                                                                                                         |
| <u>level</u> ( <i>#</i> )                    | set confidence level; default is level(95)                                                                              |
| <u>nocns</u> report                          | do not display constraints                                                                                              |
| <i>display_options</i>                       | control spacing and display of omitted variables and base and empty cells                                               |
| Integration                                  |                                                                                                                         |
| <u>int</u> points( <i>#</i> )                | number of quadrature points                                                                                             |
| Maximization                                 |                                                                                                                         |
| <i>maximize_options</i>                      | control the maximization process; seldom used                                                                           |
| <sup>†</sup> <u>coef</u> legend              | display coefficients' legend instead of coefficient table                                                               |

<sup>†</sup>coeflegend does not appear in the dialog box.

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

bootstrap, by, jackknife, mi estimate, rolling, statsby, and svy are allowed; see [U] 11.1.10 Prefix commands.

vce(bootstrap) and vce(jackknife) are not allowed with the mi estimate prefix.

Weights are not allowed with the bootstrap prefix.

vce() and weights are not allowed with the svy prefix.

fweights, iweights, and pweights are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Categorical outcomes > Independent multinomial probit

## Description

`mprobit` fits multinomial probit (MNP) models via maximum likelihood. *depvar* contains the outcome for each observation, and *indepvars* are the associated covariates. The error terms are assumed to be independent, standard normal, random variables. See [R] [asmprobit](#) for the case where the latent-variable errors are correlated or heteroskedastic and you have alternative-specific variables.

## Options

### Model

`noconstant` suppresses the  $J - 1$  constant terms.

`baseoutcome(# | lbl)` specifies the outcome used to normalize the location of the latent variable. The base outcome may be specified as a number or a label. The default is to use the most frequent outcome. The coefficients associated with the base outcome are zero.

`probitparam` specifies to use the probit variance parameterization by fixing the variance of the differenced latent errors between the scale and the base alternatives to be one. The default is to make the variance of the base and scale latent errors one, thereby making the variance of the difference to be two.

`constraints(constraints)`, `collinear`; see [R] [estimation options](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

If specifying `vce(bootstrap)` or `vce(jackknife)`, you must also specify `baseoutcome()`.

### Reporting

`level(#)`; see [R] [estimation options](#).

`nocnsreport`; see [R] [estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

### Integration

`intpoints(#)` specifies the number of Gaussian quadrature points to use in approximating the likelihood. The default is 15.

### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [R] [maximize](#). These options are seldom used.

Setting the optimization type to `technique(bhhh)` resets the default `vcetype` to `vce(opg)`.

The following option is available with `mprobit` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

The MNP model is used with discrete dependent variables that take on more than two outcomes that do not have a natural ordering. The stochastic error terms for this implementation of the model are assumed to have independent, standard normal distributions. To use `mprobit`, you must have one observation for each decision maker in the sample. See [R] [asmprobit](#) for another implementation of the MNP model that permits correlated and heteroskedastic errors and is suitable when you have data for each alternative that a decision maker faced.

The MNP model is frequently motivated using a latent-variable framework. The latent variable for the  $j$ th alternative,  $j = 1, \dots, J$ , is

$$\eta_{ij} = \mathbf{z}_i \boldsymbol{\alpha}_j + \xi_{ij}$$

where the  $1 \times q$  row vector  $\mathbf{z}_i$  contains the observed independent variables for the  $i$ th decision maker. Associated with  $\mathbf{z}_i$  are the  $J$  vectors of regression coefficients  $\boldsymbol{\alpha}_j$ . The  $\xi_{i,1}, \dots, \xi_{i,J}$  are distributed independently and identically standard normal. The decision maker chooses the alternative  $k$  such that  $\eta_{ik} \geq \eta_{im}$  for  $m \neq k$ .

Suppose that case  $i$  chooses alternative  $k$ , and take the difference between latent variable  $\eta_{ik}$  and the  $J - 1$  others:

$$\begin{aligned} v_{ijk} &= \eta_{ij} - \eta_{ik} \\ &= \mathbf{z}_i(\boldsymbol{\alpha}_j - \boldsymbol{\alpha}_k) + \xi_{ij} - \xi_{ik} \\ &= \mathbf{z}_i \boldsymbol{\gamma}_{j'} + \epsilon_{ij'} \end{aligned} \tag{1}$$

where  $j' = j$  if  $j < k$  and  $j' = j - 1$  if  $j > k$  so that  $j' = 1, \dots, J - 1$ .  $\text{Var}(\epsilon_{ij'}) = \text{Var}(\xi_{ij} - \xi_{ik}) = 2$  and  $\text{Cov}(\epsilon_{ij'}, \epsilon_{il'}) = 1$  for  $j' \neq l'$ . The probability that alternative  $k$  is chosen is

$$\begin{aligned} \Pr(i \text{ chooses } k) &= \Pr(v_{i1k} \leq 0, \dots, v_{i,J-1,k} \leq 0) \\ &= \Pr(\epsilon_{i1} \leq -\mathbf{z}_i \boldsymbol{\gamma}_1, \dots, \epsilon_{i,J-1} \leq -\mathbf{z}_i \boldsymbol{\gamma}_{J-1}) \end{aligned}$$

Hence, evaluating the likelihood function involves computing probabilities from the multivariate normal distribution. That all the covariances are equal simplifies the problem somewhat; see [Methods and formulas](#) for details.

In (1), not all  $J$  of the  $\boldsymbol{\alpha}_j$  are identifiable. To remove the indeterminacy,  $\boldsymbol{\alpha}_l$  is set to the zero vector, where  $l$  is the base outcome as specified in the `baseoutcome()` option. That fixes the  $l$ th latent variable to zero so that the remaining variables measure the attractiveness of the other alternatives relative to the base.

### ► Example 1

As discussed in [example 1](#) of [R] [mlogit](#), we have data on the type of health insurance available to 616 psychologically depressed subjects in the United States (Tarlov et al. 1989; Wells et al. 1989). Patients may have either an indemnity (fee-for-service) plan or a prepaid plan such as an HMO, or the patient may be uninsured. Demographic variables include age, gender, race, and site. Indemnity insurance is the most popular alternative, so `mprobit` will choose it as the base outcome by default.

```
. use http://www.stata-press.com/data/r11/sysdsn1
(Health insurance data)
```

```
. mprobit insure age male nonwhite i.site
```

```
Iteration 0: log likelihood = -535.89424
```

```
Iteration 1: log likelihood = -534.56173
```

```
Iteration 2: log likelihood = -534.52835
```

```
Iteration 3: log likelihood = -534.52833
```

```
Multinomial probit regression
```

```
Number of obs = 615
```

```
Wald chi2(10) = 40.18
```

```
Prob > chi2 = 0.0000
```

```
Log likelihood = -534.52833
```

| insure    | Coef.          | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-----------|----------------|-----------|-------|-------|----------------------|-----------|
| Indemnity | (base outcome) |           |       |       |                      |           |
| Prepaid   |                |           |       |       |                      |           |
| age       | -.0098536      | .0052688  | -1.87 | 0.061 | -.0201802            | .000473   |
| male      | .4774678       | .1718316  | 2.78  | 0.005 | .1406841             | .8142515  |
| nonwhite  | .8245003       | .1977582  | 4.17  | 0.000 | .4369013             | 1.212099  |
| site      |                |           |       |       |                      |           |
| 2         | .0973956       | .1794546  | 0.54  | 0.587 | -.2543289            | .4491201  |
| 3         | -.495892       | .1904984  | -2.60 | 0.009 | -.869262             | -.1225221 |
| _cons     | .22315         | .2792424  | 0.80  | 0.424 | -.324155             | .7704549  |
| Uninsure  |                |           |       |       |                      |           |
| age       | -.0050814      | .0075327  | -0.67 | 0.500 | -.0198452            | .0096823  |
| male      | .3332637       | .2432986  | 1.37  | 0.171 | -.1435929            | .8101203  |
| nonwhite  | .2485859       | .2767734  | 0.90  | 0.369 | -.29388              | .7910518  |
| site      |                |           |       |       |                      |           |
| 2         | -.6899485      | .2804497  | -2.46 | 0.014 | -1.23962             | -.1402771 |
| 3         | -.1788447      | .2479898  | -0.72 | 0.471 | -.6648957            | .3072063  |
| _cons     | -.9855917      | .3891873  | -2.53 | 0.011 | -1.748385            | -.2227986 |

◀

The likelihood function for `mprobit` is derived under the assumption that all decision-making units face the same choice set, which is the union of all outcomes observed in the dataset. If that is not true for your model, then an alternative is to use the `asmprobit` command, which does not require this assumption. To do that, you will need to expand the dataset so that each decision maker has  $k_i$  observations, where  $k_i$  is the number of alternatives in the choice set faced by decision maker  $i$ . You will also need to create a binary variable to indicate the choice made by each decision maker. Moreover, you will need to use the `correlation(independent)` and `stddev(homoskedastic)` options with `asmprobit` unless you have alternative-specific variables.

(Continued on next page)

## Saved results

`mprobit` saves the following in `e()`:

|                             |                                                         |
|-----------------------------|---------------------------------------------------------|
| Scalars                     |                                                         |
| <code>e(N)</code>           | number of observations                                  |
| <code>e(k_out)</code>       | number of outcomes                                      |
| <code>e(k_points)</code>    | number of quadrature points                             |
| <code>e(k)</code>           | number of parameters                                    |
| <code>e(k_eq)</code>        | number of equations                                     |
| <code>e(k_eq_model)</code>  | number of equations in model Wald test                  |
| <code>e(k_indvars)</code>   | number of independent variables                         |
| <code>e(k_dv)</code>        | number of dependent variables                           |
| <code>e(k_autoCns)</code>   | number of base, empty, and omitted constraints          |
| <code>e(df_m)</code>        | model degrees of freedom                                |
| <code>e(ll)</code>          | log simulated-likelihood                                |
| <code>e(N_clust)</code>     | number of clusters                                      |
| <code>e(chi2)</code>        | $\chi^2$                                                |
| <code>e(p)</code>           | significance                                            |
| <code>e(i_base)</code>      | base outcome index                                      |
| <code>e(const)</code>       | 0 if <code>noconstant</code> is specified, 1 otherwise  |
| <code>e(probitparam)</code> | 1 if <code>probitparam</code> is specified, 0 otherwise |
| <code>e(rank)</code>        | rank of <code>e(V)</code>                               |
| <code>e(ic)</code>          | number of iterations                                    |
| <code>e(rc)</code>          | return code                                             |
| <code>e(converged)</code>   | 1 if converged, 0 otherwise                             |



## Macros

|                    |                                                                          |
|--------------------|--------------------------------------------------------------------------|
| e(cmd)             | mprobit                                                                  |
| e(cmd2)            | mprobit                                                                  |
| e(cmdline)         | command as typed                                                         |
| e(depvar)          | name of dependent variable                                               |
| e(indvars)         | independent variables                                                    |
| e(wtype)           | weight type                                                              |
| e(wexp)            | weight expression                                                        |
| e(title)           | title in estimation output                                               |
| e(clustvar)        | name of cluster variable                                                 |
| e(chi2type)        | Wald, type of model $\chi^2$ test                                        |
| e(vce)             | <i>vcetype</i> specified in <code>vce()</code>                           |
| e(vcetype)         | title used to label Std. Err.                                            |
| e(outeqs)          | outcome equations                                                        |
| e(out#)            | outcome labels, #=1,...,e(k_out)                                         |
| e(opt)             | type of optimization                                                     |
| e(which)           | max or min; whether optimizer is to perform maximization or minimization |
| e(ml_method)       | type of ml method                                                        |
| e(user)            | name of likelihood-evaluator program                                     |
| e(technique)       | maximization technique                                                   |
| e(singularHmethod) | m-marquardt or hybrid; method used when Hessian is singular              |
| e(crittype)        | optimization criterion                                                   |
| e(properties)      | b V                                                                      |
| e(predict)         | program used to implement predict                                        |
| e(marginsnotok)    | predictions disallowed by margins                                        |
| e(asbalanced)      | factor variables <code>fvset</code> as <code>asbalanced</code>           |
| e(asobserved)      | factor variables <code>fvset</code> as <code>asobserved</code>           |

## Matrices

|                 |                                              |
|-----------------|----------------------------------------------|
| e(b)            | coefficient vector                           |
| e(outcomes)     | outcome values                               |
| e(Cns)          | constraints matrix                           |
| e(ilog)         | iteration log (up to 20 iterations)          |
| e(gradient)     | gradient vector                              |
| e(V)            | variance–covariance matrix of the estimators |
| e(V_modelbased) | model-based variance                         |

## Functions

|           |                         |
|-----------|-------------------------|
| e(sample) | marks estimation sample |
|-----------|-------------------------|

## Methods and formulas

mprobit is implemented as an ado-file.

See [Cameron and Trivedi \(2005, chap. 15\)](#) for a discussion of multinomial models, including multinomial probit. [Long and Freese \(2006, chap. 6\)](#) discuss the multinomial logistic, multinomial probit, and stereotype logistic regression models, with examples using Stata.

As discussed in [Remarks](#), the latent variables for a  $J$ -alternative model are  $\eta_{ij} = \mathbf{z}_i \boldsymbol{\alpha}_j + \xi_{ij}$ , for  $j = 1, \dots, J$ ,  $i = 1, \dots, n$ , and  $\{\xi_{i,1}, \dots, \xi_{i,J}\} \sim \text{i.i.d.} N(0, 1)$ . The experimenter observes alternative  $k$  for the  $i$ th observation if  $\eta_{ik} > \eta_{il}$  for  $l \neq k$ . For  $j \neq k$ , let

$$\begin{aligned}
v_{ij'} &= \eta_{ij} - \eta_{ik} \\
&= \mathbf{z}_i(\boldsymbol{\alpha}_j - \boldsymbol{\alpha}_k) + \xi_{ij} - \xi_{ik} \\
&= \mathbf{z}_i\boldsymbol{\gamma}_{j'} + \epsilon_{ij'}
\end{aligned}$$

where  $j' = j$  if  $j < k$  and  $j' = j - 1$  if  $j > k$  so that  $j' = 1, \dots, J - 1$ .  $\epsilon_i = (\epsilon_{i1}, \dots, \epsilon_{i,J-1}) \sim \text{MVN}(\mathbf{0}, \boldsymbol{\Sigma})$ , where

$$\boldsymbol{\Sigma} = \begin{pmatrix} 2 & 1 & 1 & \dots & 1 \\ 1 & 2 & 1 & \dots & 1 \\ 1 & 1 & 2 & \dots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 2 \end{pmatrix}$$

Denote the deterministic part of the model as  $\lambda_{ij'} = \mathbf{z}_i\boldsymbol{\gamma}_{j'}$ ; the probability that subject  $i$  chooses outcome  $k$  is

$$\begin{aligned}
\Pr(y_i = k) &= \Pr(v_{i1} \leq 0, \dots, v_{i,J-1} \leq 0) \\
&= \Pr(\epsilon_{i1} \leq -\lambda_{i1}, \dots, \epsilon_{i,J-1} \leq -\lambda_{i,J-1}) \\
&= \frac{1}{(2\pi)^{(J-1)/2} |\boldsymbol{\Sigma}|^{1/2}} \int_{-\infty}^{-\lambda_{i1}} \dots \int_{-\infty}^{-\lambda_{i,J-1}} \exp\left(-\frac{1}{2}\mathbf{z}'\boldsymbol{\Sigma}^{-1}\mathbf{z}\right) d\mathbf{z}
\end{aligned}$$

Because of the exchangeable correlation structure of  $\boldsymbol{\Sigma}$  ( $\rho_{ij} = 1/2$  for all  $i \neq j$ ), we can use Dunnett's (1989) result to reduce the multidimensional integral to one dimension:

$$\Pr(y_i = k) = \frac{1}{\sqrt{\pi}} \int_0^\infty \left\{ \prod_{j=1}^{J-1} \Phi\left(-z\sqrt{2} - \lambda_{ij}\right) + \prod_{j=1}^{J-1} \Phi\left(z\sqrt{2} - \lambda_{ij}\right) \right\} e^{-z^2} dz$$

Gaussian quadrature is used to approximate this integral, resulting in the  $K$ -point quadrature formula

$$\Pr(y_i = k) \approx \frac{1}{2} \sum_{k=1}^K w_k \left\{ \prod_{j=1}^{J-1} \Phi\left(-\sqrt{2x_k} - \lambda_{ij}\right) + \prod_{j=1}^{J-1} \Phi\left(\sqrt{2x_k} - \lambda_{ij}\right) \right\}$$

where  $w_k$  and  $x_k$  are the weights and roots of the Laguerre polynomial of order  $K$ . In `mprobit`,  $K$  is specified by the `intpoints()` option.

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

`mprobit` also supports estimation with survey data. For details on VCEs with survey data, see [\[SVY\] variance estimation](#).

## References

- Cameron, A. C., and P. K. Trivedi. 2005. *Microeconometrics: Methods and Applications*. New York: Cambridge University Press.
- Dunnett, C. W. 1989. Algorithm AS 251: Multivariate normal probability integrals with product correlation structure. *Journal of the Royal Statistical Society, Series C* 38: 564–579.
- Haan, P., and A. Uhlenborff. 2006. Estimation of multinomial logit models with unobserved heterogeneity using maximum simulated likelihood. *Stata Journal* 6: 229–245.
- Hole, A. R. 2007. Fitting mixed logit models by using maximum simulated likelihood. *Stata Journal* 7: 388–401.
- Long, J. S., and J. Freese. 2006. *Regression Models for Categorical Dependent Variables Using Stata*. 2nd ed. College Station, TX: Stata Press.
- Tarlov, A. R., J. E. Ware Jr., S. Greenfield, E. C. Nelson, E. Perrin, and M. Zubkoff. 1989. The medical outcomes study. An application of methods for monitoring the results of medical care. *Journal of the American Medical Association* 262: 925–930.
- Wells, K. B., R. D. Hays, M. A. Burnam, W. H. Rogers, S. Greenfield, and J. E. Ware Jr. 1989. Detection of depressive disorder for patients receiving prepaid or fee-for-service care. Results from the Medical Outcomes Survey. *Journal of the American Medical Association* 262: 3298–3302.

## Also see

- [R] **mprobit postestimation** — Postestimation tools for mprobit
- [R] **asmprobit** — Alternative-specific multinomial probit regression
- [R] **mlogit** — Multinomial (polytomous) logistic regression
- [R] **clogit** — Conditional (fixed-effects) logistic regression
- [R] **nlogit** — Nested logit regression
- [R] **ologit** — Ordered logistic regression
- [R] **oprobit** — Ordered probit regression
- [SVY] **svy estimation** — Estimation commands for survey data
- [U] **20 Estimation and postestimation commands**

Description

The following postestimation commands are available for `mprobit`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>hausman</code>             | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predicted probabilities, linear predictions, and standard errors                                    |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Syntax for predict

```
predict [type] { stub*|newvar|newvarlist } [if] [in] [, statistic outcome(outcome)]

predict [type] { stub*|newvarlist } [if] [in], scores
```

| statistic         | description                                    |
|-------------------|------------------------------------------------|
| Main              |                                                |
| <code>pr</code>   | probability of a positive outcome; the default |
| <code>xb</code>   | linear prediction                              |
| <code>stdp</code> | standard error of the linear prediction        |

If you do not specify `outcome()`, `pr` (with one new variable specified), `xb`, and `stdp` assume `outcome(#1)`.  
You specify one or  $k$  new variables with `pr`, where  $k$  is the number of outcomes.  
You specify one new variable with `xb` and `stdp`.  
These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

**pr**, the default, calculates the probability of each of the categories of the dependent variable or the probability of the level specified in `outcome(outcome)`. If you specify the `outcome(outcome)` option, you need to specify only one new variable; otherwise, you must specify a new variable for each category of the dependent variable.

**xb** calculates the linear prediction,  $\mathbf{x}_i\boldsymbol{\alpha}_j$ , for alternative  $j$  and individual  $i$ . The index,  $j$ , corresponds to the outcome specified in `outcome()`.

**stdp** calculates the standard error of the linear prediction.

`outcome(outcome)` specifies the outcome for which the statistic is to be calculated. `equation()` is a synonym for `outcome()`: it does not matter which you use. `outcome()` or `equation()` can be specified using

#1, #2, ..., where #1 means the first category of the dependent variable, #2 means the second category, etc.;

the values of the dependent variable; or

the value labels of the dependent variable if they exist.

**scores** calculates the equation-level score variables. The  $j$ th new variable will contain the scores for the  $j$ th fitted equation.

## Remarks

Once you have fit a multinomial probit model, you can use **predict** to obtain probabilities that an individual will choose each of the alternatives for the estimation sample, as well as other samples; see [\[U\] 20 Estimation and postestimation commands](#) and [\[R\] predict](#).

### ► Example 1

In [example 1](#) of [\[R\] mprobit](#), we fit the multinomial probit model to a dataset containing the type of health insurance available to 616 psychologically depressed subjects in the United States (Tarlo et al. 1989; Wells et al. 1989). We can obtain the predicted probabilities by typing

```
. use http://www.stata-press.com/data/r11/sysdsn1
(Health insurance data)
. mprobit insure age male nonwhite i.site
(output omitted)
. predict p1-p3
(option pr assumed; predicted probabilities)
```

```
. list p1-p3 insure in 1/10
```

|     | p1       | p2       | p3       | insure    |
|-----|----------|----------|----------|-----------|
| 1.  | .5961306 | .3741824 | .029687  | Indemnity |
| 2.  | .4719296 | .4972289 | .0308415 | Prepaid   |
| 3.  | .4896086 | .4121961 | .0981953 | Indemnity |
| 4.  | .3730529 | .5416623 | .0852848 | Prepaid   |
| 5.  | .5063069 | .4629773 | .0307158 | .         |
| 6.  | .4768125 | .4923548 | .0308327 | Prepaid   |
| 7.  | .5035672 | .4657016 | .0307312 | Prepaid   |
| 8.  | .3326361 | .5580404 | .1093235 | .         |
| 9.  | .4758165 | .4384811 | .0857024 | Uninsure  |
| 10. | .5734057 | .3316601 | .0949342 | Prepaid   |

`insure` contains a missing value for observations 5 and 8. Because of that, those two observations were not used in the estimation. However, because none of the independent variables is missing, `predict` can still calculate the probabilities. Had we typed

```
. predict p1-p3 if e(sample)
```

`predict` would have filled in missing values for `p1`, `p2`, and `p3` for those observations because they were not used in the estimation.



## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## References

Tarlov, A. R., J. E. Ware Jr., S. Greenfield, E. C. Nelson, E. Perrin, and M. Zubkoff. 1989. The medical outcomes study. An application of methods for monitoring the results of medical care. *Journal of the American Medical Association* 262: 925–930.

Wells, K. B., R. D. Hays, M. A. Burnam, W. H. Rogers, S. Greenfield, and J. E. Ware Jr. 1989. Detection of depressive disorder for patients receiving prepaid or fee-for-service care. Results from the Medical Outcomes Survey. *Journal of the American Medical Association* 262: 3298–3302.

## Also see

- [R] [mprobit](#) — Multinomial probit regression
- [U] [20 Estimation and postestimation commands](#)

Syntax

```
mvreg depvars = indepvars [if] [in] [weight] [, options]
```

| <i>options</i>                       | description                                                               |
|--------------------------------------|---------------------------------------------------------------------------|
| Model                                |                                                                           |
| <code>noconstant</code>              | suppress constant term                                                    |
| Reporting                            |                                                                           |
| <code>level(#)</code>                | set confidence level; default is <code>level(95)</code>                   |
| <code>corr</code>                    | report correlation matrix                                                 |
| <code>display_options</code>         | control spacing and display of omitted variables and base and empty cells |
| <sup>†</sup> <code>noheader</code>   | suppress header table from above coefficient table                        |
| <sup>†</sup> <code>notable</code>    | suppress coefficient table                                                |
| <sup>†</sup> <code>coeflegend</code> | display coefficients' legend instead of coefficient table                 |

<sup>†</sup> `noheader`, `notable`, and `coeflegend` are not shown in the dialog box.  
*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.  
*depvars* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.  
`bootstrap`, `by`, `jackknife`, `mi estimate`, `rolling`, and `statsby` are allowed; see [U] 11.1.10 Prefix commands.  
Weights are not allowed with the `bootstrap` prefix.  
`aweight`s are not allowed with the `jackknife` prefix.  
`aweight`s and `fweight`s are allowed; see [U] 11.1.6 weight.  
See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Linear models and related > Multiple-equation models > Multivariate regression

Description

`mvreg` fits multivariate regression models.

Options

|                                                                                                                                                                                                                                         |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                                                                                                                                                                                                                   |
| <code>noconstant</code> suppresses the constant term (intercept) in the model.                                                                                                                                                          |
| Reporting                                                                                                                                                                                                                               |
| <code>level(#)</code> specifies the confidence level, as a percentage, for confidence intervals. The default is <code>level(95)</code> or as set by <code>set level</code> ; see [U] 20.7 Specifying the width of confidence intervals. |

`corr` displays the correlation matrix of the residuals between the equations.

*display\_options*: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

The following options are available with `mvreg` but are not shown in the dialog box:

`noheader` suppresses display of the table reporting  $F$  statistics,  $R$ -squared, and root mean squared error above the coefficient table.

`notable` suppresses display of the coefficient table.

`coeflegend`; see [R] [estimation options](#).

## Remarks

Multivariate regression differs from multiple regression in that *several* dependent variables are jointly regressed on the same independent variables. Multivariate regression is related to Zellner's seemingly unrelated regression (see [R] [sureg](#)), but because the same set of independent variables is used for each dependent variable, the syntax is simpler, and the calculations are faster.

The individual coefficients and standard errors produced by `mvreg` are identical to those that would be produced by `regress` estimating each equation separately. The difference is that `mvreg`, being a joint estimator, also estimates the between-equation covariances, so you can test coefficients across equations and, in fact, the `test` syntax makes such tests more convenient.

### ► Example 1

Using the automobile data, we fit a multivariate regression for space variables (`headroom`, `trunk`, and `turn`) in terms of a set of other variables, including three performance variables (`displacement`, `gear_ratio`, and `mpg`):



```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
```

```
. mvreg headroom trunk turn = price mpg displ gear_ratio length weight
```

| Equation | Obs | Parms | RMSE     | "R-sq" | F        | P      |
|----------|-----|-------|----------|--------|----------|--------|
| headroom | 74  | 7     | .7390205 | 0.2996 | 4.777213 | 0.0004 |
| trunk    | 74  | 7     | 3.052314 | 0.5326 | 12.7265  | 0.0000 |
| turn     | 74  | 7     | 2.132377 | 0.7844 | 40.62042 | 0.0000 |

|              | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|--------------|-----------|-----------|-------|-------|----------------------|-----------|
| headroom     |           |           |       |       |                      |           |
| price        | -.0000528 | .000038   | -1.39 | 0.168 | -.0001286            | .0000229  |
| mpg          | -.0093774 | .0260463  | -0.36 | 0.720 | -.061366             | .0426112  |
| displacement | .0031025  | .0024999  | 1.24  | 0.219 | -.0018873            | .0080922  |
| gear_ratio   | .2108071  | .3539588  | 0.60  | 0.553 | -.4956976            | .9173119  |
| length       | .015886   | .012944   | 1.23  | 0.224 | -.0099504            | .0417223  |
| weight       | -.0000868 | .0004724  | -0.18 | 0.855 | -.0010296            | .0008561  |
| _cons        | -.4525117 | 2.170073  | -0.21 | 0.835 | -4.783995            | 3.878972  |
| trunk        |           |           |       |       |                      |           |
| price        | .0000445  | .0001567  | 0.28  | 0.778 | -.0002684            | .0003573  |
| mpg          | -.0220919 | .1075767  | -0.21 | 0.838 | -.2368159            | .1926322  |
| displacement | .0032118  | .0103251  | 0.31  | 0.757 | -.0173971            | .0238207  |
| gear_ratio   | -.2271321 | 1.461926  | -0.16 | 0.877 | -3.145149            | 2.690885  |
| length       | .170811   | .0534615  | 3.20  | 0.002 | .0641014             | .2775206  |
| weight       | -.0015944 | .001951   | -0.82 | 0.417 | -.0054885            | .0022997  |
| _cons        | -13.28253 | 8.962868  | -1.48 | 0.143 | -31.17249            | 4.607429  |
| turn         |           |           |       |       |                      |           |
| price        | -.0002647 | .0001095  | -2.42 | 0.018 | -.0004833            | -.0000462 |
| mpg          | -.0492948 | .0751542  | -0.66 | 0.514 | -.1993031            | .1007136  |
| displacement | .0036977  | .0072132  | 0.51  | 0.610 | -.0106999            | .0180953  |
| gear_ratio   | -.1048432 | 1.021316  | -0.10 | 0.919 | -2.143399            | 1.933712  |
| length       | .072128   | .0373487  | 1.93  | 0.058 | -.0024204            | .1466764  |
| weight       | .0027059  | .001363   | 1.99  | 0.051 | -.0000145            | .0054264  |
| _cons        | 20.19157  | 6.261549  | 3.22  | 0.002 | 7.693467             | 32.68968  |

We should have specified the `corr` option so that we would also see the correlations between the residuals of the equations. We can correct our omission because `mvreg`—like all estimation commands—typed without arguments redisplay results. The `noheader` and `notable` (read “no-table”) options suppress redisplaying the output we have already seen:

```
. mvreg, notable noheader corr
```

```
Correlation matrix of residuals:
```

|          | headroom | trunk   | turn   |
|----------|----------|---------|--------|
| headroom | 1.0000   |         |        |
| trunk    | 0.4986   | 1.0000  |        |
| turn     | -0.1090  | -0.0628 | 1.0000 |

```
Breusch-Pagan test of independence: chi2(3) = 19.566, Pr = 0.0002
```

The Breusch–Pagan test is significant, so the residuals of these three space variables are not independent of each other.

The three performance variables among our independent variables are `mpg`, `displacement`, and `gear_ratio`. We can jointly test the significance of these three variables in all the equations by typing

```

. test mpg displacement gear_ratio
(1) [headroom]mpg = 0
(2) [trunk]mpg = 0
(3) [turn]mpg = 0
(4) [headroom]displacement = 0
(5) [trunk]displacement = 0
(6) [turn]displacement = 0
(7) [headroom]gear_ratio = 0
(8) [trunk]gear_ratio = 0
(9) [turn]gear_ratio = 0

 F(9, 67) = 0.33
 Prob > F = 0.9622

```

These three variables are not, as a group, significant. We might have suspected this from their individual significance in the individual regressions, but this multivariate test provides an overall assessment with one  $p$ -value.

We can also perform a test for the joint significance of all three equations:

```

. test [headroom]
(output omitted)
. test [trunk], accum
(output omitted)
. test [turn], accum
(1) [headroom]price = 0
(2) [headroom]mpg = 0
(3) [headroom]displacement = 0
(4) [headroom]gear_ratio = 0
(5) [headroom]length = 0
(6) [headroom]weight = 0
(7) [trunk]price = 0
(8) [trunk]mpg = 0
(9) [trunk]displacement = 0
(10) [trunk]gear_ratio = 0
(11) [trunk]length = 0
(12) [trunk]weight = 0
(13) [turn]price = 0
(14) [turn]mpg = 0
(15) [turn]displacement = 0
(16) [turn]gear_ratio = 0
(17) [turn]length = 0
(18) [turn]weight = 0

 F(18, 67) = 19.34
 Prob > F = 0.0000

```

The set of variables as a whole is strongly significant. We might have suspected this, too, from the individual equations.



## □ Technical note

The `mvreg` command provides a good way to deal with multiple comparisons. If we wanted to assess the effect of `length`, we might be dissuaded from interpreting any of its coefficients except that in the `trunk` equation. `[trunk]length`—the coefficient on `length` in the `trunk` equation—has a  $p$ -value of 0.002, but in the other two equations, it has  $p$ -values of only 0.224 and 0.058.

A conservative statistician might argue that there are 18 tests of significance in `mvreg`'s output (not counting those for the intercept), so  $p$ -values more than  $0.05/18 = 0.0028$  should be declared

insignificant at the 5% level. A more aggressive but, in our opinion, reasonable approach would be to first note that the three equations are jointly significant, so we are justified in making some interpretation. Then we would work through the individual variables using `test`, possibly using  $0.05/6 = 0.0083$  (6 because there are six independent variables) for the 5% significance level. For instance, examining `length`:

```
. test length
(1) [headroom]length = 0
(2) [trunk]length = 0
(3) [turn]length = 0
 F(3, 67) = 4.94
 Prob > F = 0.0037
```

The reported significance level of 0.0037 is less than 0.0083, so we will declare this variable significant. `[trunk]length` is certainly significant with its  $p$ -value of 0.002, but what about in the remaining two equations with  $p$ -values 0.224 and 0.058? We perform a joint test:

```
. test [headroom]length [turn]length
(1) [headroom]length = 0
(2) [turn]length = 0
 F(2, 67) = 2.91
 Prob > F = 0.0613
```

At this point, reasonable statisticians could disagree. The 0.06 significance value suggests no interpretation, but these were the two least-significant values out of three, so we would expect the  $p$ -value to be a little high. Perhaps an equivocal statement is warranted: there seems to be an effect, but chance cannot be excluded.

□

*(Continued on next page)*

## Saved results

`mvreg` saves the following in `e()`:

### Scalars

|                         |                                                           |
|-------------------------|-----------------------------------------------------------|
| <code>e(N)</code>       | number of observations                                    |
| <code>e(k)</code>       | number of parameters (including constant)                 |
| <code>e(k_eq)</code>    | number of equations                                       |
| <code>e(df_r)</code>    | residual degrees of freedom                               |
| <code>e(chi2)</code>    | Breusch–Pagan $\chi^2$ (corr only)                        |
| <code>e(df_chi2)</code> | degrees of freedom for Breusch–Pagan $\chi^2$ (corr only) |
| <code>e(rank)</code>    | rank of <code>e(V)</code>                                 |

### Macros

|                              |                                                                |
|------------------------------|----------------------------------------------------------------|
| <code>e(cmd)</code>          | <code>mvreg</code>                                             |
| <code>e(cmdline)</code>      | command as typed                                               |
| <code>e(depvar)</code>       | names of dependent variables                                   |
| <code>e(eqnames)</code>      | names of equations                                             |
| <code>e(wtype)</code>        | weight type                                                    |
| <code>e(wexp)</code>         | weight expression                                              |
| <code>e(r2)</code>           | $R$ -squared for each equation                                 |
| <code>e(rmse)</code>         | RMSE for each equation                                         |
| <code>e(F)</code>            | $F$ statistic for each equation                                |
| <code>e(p_F)</code>          | significance of $F$ for each equation                          |
| <code>e(properties)</code>   | <code>b V</code>                                               |
| <code>e(estat_cmd)</code>    | program used to implement <code>estat</code>                   |
| <code>e(predict)</code>      | program used to implement <code>predict</code>                 |
| <code>e(marginsok)</code>    | predictions allowed by <code>margins</code>                    |
| <code>e(marginsnotok)</code> | predictions disallowed by <code>margins</code>                 |
| <code>e(asbalanced)</code>   | factor variables <code>fvset</code> as <code>asbalanced</code> |
| <code>e(asobserved)</code>   | factor variables <code>fvset</code> as <code>asobserved</code> |

### Matrices

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>e(b)</code>     | coefficient vector                           |
| <code>e(Sigma)</code> | $\Sigma$ matrix                              |
| <code>e(V)</code>     | variance–covariance matrix of the estimators |

### Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`mvreg` is implemented as an ado-file.

Given  $q$  equations and  $p$  independent variables (including the constant), the parameter estimates are given by the  $p \times q$  matrix

$$\mathbf{B} = (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{Y}$$

where  $\mathbf{Y}$  is an  $n \times q$  matrix of dependent variables and  $\mathbf{X}$  is a  $n \times p$  matrix of independent variables.  $\mathbf{W}$  is a weighting matrix equal to  $\mathbf{I}$  if no weights are specified. If weights are specified, let  $\mathbf{v}$ :  $1 \times n$  be the specified weights. If `fweight` frequency weights are specified,  $\mathbf{W} = \text{diag}(\mathbf{v})$ . If `aweight` analytic weights are specified,  $\mathbf{W} = \text{diag}\{\mathbf{v}/(\mathbf{1}'\mathbf{v})(\mathbf{1}'\mathbf{1})\}$ , meaning that the weights are normalized to sum to the number of observations.

The residual covariance matrix is

$$\mathbf{R} = \{\mathbf{Y}'\mathbf{W}\mathbf{Y} - \mathbf{B}'(\mathbf{X}'\mathbf{W}\mathbf{X})\mathbf{B}\}/(n - p)$$

The estimated covariance matrix of the estimates is  $\mathbf{R} \otimes (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}$ . These results are identical to those produced by `sureg` when the same list of independent variables is specified repeatedly; see [R] [sureg](#).

The [Breusch and Pagan \(1980\)](#)  $\chi^2$  statistic—a Lagrange multiplier statistic—is given by

$$\lambda = n \sum_{i=1}^q \sum_{j=1}^{i-1} r_{ij}^2$$

where  $r_{ij}$  is the estimated correlation between the residuals of the equations and  $n$  is the number of observations. It is distributed as  $\chi^2$  with  $q(q-1)/2$  degrees of freedom.

## Reference

Breusch, T. S., and A. R. Pagan. 1980. The Lagrange multiplier test and its applications to model specification in econometrics. *Review of Economic Studies* 47: 239–253.

## Also see

- [R] [mvreg postestimation](#) — Postestimation tools for mvreg
- [MV] [manova](#) — Multivariate analysis of variance and covariance
- [R] [nlsur](#) — Estimation of nonlinear systems of equations
- [R] [reg3](#) — Three-stage estimation for systems of simultaneous equations
- [R] [regress](#) — Linear regression
- [R] [regress postestimation](#) — Postestimation tools for regress
- [R] [sureg](#) — Zellner’s seemingly unrelated regression
- [U] [20 Estimation and postestimation commands](#)

## Description

The following postestimation commands are available for `mvreg`:

| command                | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>     | VCE and estimation sample summary                                                                   |
| <code>estimates</code> | cataloging estimation results                                                                       |
| <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>margins</code>   | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>   | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code> | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details.

## Syntax for predict

`predict` [*type*] *newvar* [*if*] [*in*] [, equation(*eqno* [, *eqno*]) *statistic*]

| <i>statistic</i>               | description                                                |
|--------------------------------|------------------------------------------------------------|
| Main                           |                                                            |
| <code>xb</code>                | linear prediction; the default                             |
| <code>stdp</code>              | standard error of the linear prediction                    |
| <code><u>r</u>esiduals</code>  | residuals                                                  |
| <code><u>d</u>ifference</code> | difference between the linear predictions of two equations |
| <code><u>s</u>tddp</code>      | standard error of the difference in linear predictions     |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

`equation(eqno [ , eqno ])` specifies the equation to which you are referring.

`equation()` is filled in with one *eqno* for the `xb`, `stdp`, and `residuals` options. `equation(#1)` would mean the calculation is to be made for the first equation, `equation(#2)` would mean the second, and so on. You could also refer to the equations by their names. `equation(income)` would refer to the equation named `income` and `equation(hours)`, to the equation named `hours`.

If you do not specify `equation()`, results are the same as if you specified `equation(#1)`.

`difference` and `stdp` refer to between-equation concepts. To use these options, you must specify two equations, e.g., `equation(#1,#2)` or `equation(income,hours)`. When two equations must be specified, `equation()` is required. With `equation(#1,#2)`, `difference` computes the prediction of `equation(#1)` minus the prediction of `equation(#2)`.

`xb`, the default, calculates the fitted values—the prediction of  $\mathbf{x}_j\mathbf{b}$  for the specified equation.

`stdp` calculates the standard error of the prediction for the specified equation (the standard error of the predicted expected value or mean for the observation's covariate pattern). The standard error of the prediction is also referred to as the standard error of the fitted value.

`residuals` calculates the residuals.

`difference` calculates the difference between the linear predictions of two equations in the system.

`stdp` is allowed only after you have previously fit a multiple-equation model. The standard error of the difference in linear predictions ( $\mathbf{x}_{1j}\mathbf{b} - \mathbf{x}_{2j}\mathbf{b}$ ) between equations 1 and 2 is calculated.

For more information on using `predict` after multiple-equation estimation commands, see [\[R\] predict](#).

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[\[R\] mvreg](#) — Multivariate regression

[\[U\] 20 Estimation and postestimation commands](#)

Syntax

Negative binomial regression model

```
nbreg depvar [indepvars] [if] [in] [weight] [, nbreg_options]
```

Generalized negative binomial model

```
gnbreg depvar [indepvars] [if] [in] [weight] [, gnbreg_options]
```

| <i>nbreg_options</i>                           | description                                                                                                                                                          |
|------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                          |                                                                                                                                                                      |
| <u>noconstant</u>                              | suppress constant term                                                                                                                                               |
| <u>dispersion</u> ( <u>mean</u> )              | parameterization of dispersion; <code>dispersion(mean)</code> is the default                                                                                         |
| <u>dispersion</u> ( <u>constant</u> )          | constant dispersion for all observations                                                                                                                             |
| <u>exposure</u> ( <i>varname<sub>e</sub></i> ) | include $\ln(\text{varname}_e)$ in model with coefficient constrained to 1                                                                                           |
| <u>offset</u> ( <i>varname<sub>o</sub></i> )   | include <i>varname<sub>o</sub></i> in model with coefficient constrained to 1                                                                                        |
| <u>constraints</u> ( <i>constraints</i> )      | apply specified linear constraints                                                                                                                                   |
| <u>collinear</u>                               | keep collinear variables                                                                                                                                             |
| SE/Robust                                      |                                                                                                                                                                      |
| <u>vce</u> ( <i>vcetype</i> )                  | <i>vcetype</i> may be <code>oim</code> , <code>robust</code> , <code>cluster clustvar</code> , <code>opg</code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                                      |                                                                                                                                                                      |
| <u>level</u> (#)                               | set confidence level; default is <code>level(95)</code>                                                                                                              |
| <u>nolrtest</u>                                | suppress likelihood-ratio test                                                                                                                                       |
| <u>irr</u>                                     | report incidence-rate ratios                                                                                                                                         |
| <u>nocnsreport</u>                             | do not display constraints                                                                                                                                           |
| <u>display_options</u>                         | control spacing and display of omitted variables and base and empty cells                                                                                            |
| Maximization                                   |                                                                                                                                                                      |
| <u>maximize_options</u>                        | control the maximization process; seldom used                                                                                                                        |
| <sup>†</sup> <u>coeflegend</u>                 | display coefficients' legend instead of coefficient table                                                                                                            |



| <i>gnbreg_options</i>                             | description                                                                                                                                                                 |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                             |                                                                                                                                                                             |
| <code>noconstant</code>                           | suppress constant term                                                                                                                                                      |
| <code>lnalpha(<i>varlist</i>)</code>              | dispersion model variables                                                                                                                                                  |
| <code>exposure(<i>varname<sub>e</sub></i>)</code> | include $\ln(\textit{varname}_e)$ in model with coefficient constrained to 1                                                                                                |
| <code>offset(<i>varname<sub>o</sub></i>)</code>   | include $\textit{varname}_o$ in model with coefficient constrained to 1                                                                                                     |
| <code>constraints(<i>constraints</i>)</code>      | apply specified linear constraints                                                                                                                                          |
| <code>collinear</code>                            | keep collinear variables                                                                                                                                                    |
| SE/Robust                                         |                                                                                                                                                                             |
| <code>vce(<i>vcetype</i>)</code>                  | <i>vcetype</i> may be <code>oim</code> , <code>robust</code> , <code>cluster <i>clustvar</i></code> , <code>opg</code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                                         |                                                                                                                                                                             |
| <code>level(#)</code>                             | set confidence level; default is <code>level(95)</code>                                                                                                                     |
| <code>irr</code>                                  | report incidence-rate ratios                                                                                                                                                |
| <code>nocnsreport</code>                          | do not display constraints                                                                                                                                                  |
| <code>display_options</code>                      | control spacing and display of omitted variables and base and empty cells                                                                                                   |
| Maximization                                      |                                                                                                                                                                             |
| <code>maximize_options</code>                     | control the maximization process; seldom used                                                                                                                               |
| <sup>†</sup> <code>coeflegend</code>              | display coefficients' legend instead of coefficient table                                                                                                                   |

<sup>†</sup>`coeflegend` does not appear in the dialog box.

*indepvars* and *varlist* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar*, *indepvars*, *varname<sub>e</sub>*, and *varname<sub>o</sub>* may contain time-series operators (nbreg only); see

[U] 11.4.4 Time-series varlists.

`bootstrap`, `by` (nbreg only), `fracpoly` (nbreg only), `jackknife`, `mfp` (nbreg only), `mi estimate`, `nestreg` (nbreg only), `rolling`, `statsby`, `stepwise`, and `svy` are allowed; see [U] 11.1.10 Prefix commands.

`vce(bootstrap)` and `vce(jackknife)` are not allowed with the `mi estimate` prefix.

Weights are not allowed with the `bootstrap` prefix.

`vce()` and weights are not allowed with the `svy` prefix.

`fweights`, `iweights`, and `pweights` are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

## Menu

### nbreg

Statistics > Count outcomes > Negative binomial regression

### gnbreg

Statistics > Count outcomes > Generalized negative binomial regression

## Description

**nbreg** fits a negative binomial regression model of *depvar* on *indepvars*, where *depvar* is a nonnegative count variable. In this model, the count variable is believed to be generated by a Poisson-like process, except that the variation is greater than that of a true Poisson. This extra variation is referred to as overdispersion. See [R] [poisson](#) before reading this entry.

**gnbreg** fits a generalization of the negative binomial mean-dispersion model; the shape parameter  $\alpha$  may also be parameterized.

If you have panel data, see [XT] [xtnbreg](#).

## Options for nbreg

### Model

**noconstant**; see [R] [estimation options](#).

**dispersion(mean | constant)** specifies the parameterization of the model. **dispersion(mean)**, the default, yields a model with dispersion equal to  $1 + \alpha \exp(\mathbf{x}_j\beta + \text{offset}_j)$ ; that is, the dispersion is a function of the expected mean:  $\exp(\mathbf{x}_j\beta + \text{offset}_j)$ . **dispersion(constant)** has dispersion equal to  $1 + \delta$ ; that is, it is a constant for all observations.

**exposure(varname<sub>e</sub>)**, **offset(varname<sub>o</sub>)**, **constraints(constraints)**, **collinear**; see [R] [estimation options](#).

### SE/Robust

**vce(vcetype)** specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

### Reporting

**level(#)**; see [R] [estimation options](#).

**nolrttest** suppresses fitting the Poisson model. Without this option, a comparison Poisson model is fit, and the likelihood is used in a likelihood-ratio test of the null hypothesis that the dispersion parameter is zero.

**irr** reports estimated coefficients transformed to incidence-rate ratios, that is,  $e^{\beta_i}$  rather than  $\beta_i$ . Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated or stored. **irr** may be specified at estimation or when replaying previously estimated results.

**nocnsreport**; see [R] [estimation options](#).

**display\_options**: **noomitted**, **vsquish**, **noemptycells**, **baselevels**, **allbaselevels**; see [R] [estimation options](#).

### Maximization

**maximize\_options**: **difficult**, **technique(algorithm\_spec)**, **iterate(#)**, **[no]log**, **trace**, **gradient**, **showstep**, **hessian**, **showtolerance**, **tolerance(#)**, **ltolerance(#)**, **nrtolerance(#)**, **nonrntolerance**, **from(init\_specs)**; see [R] [maximize](#). These options are seldom used.

Setting the optimization type to **technique(bhhh)** resets the default **vcetype** to **vce(opg)**.

The following option is available with `nbreg` but is not shown in the dialog box:  
`coeflegend`; see [R] [estimation options](#).

## Options for `gnbreg`

### Model

`noconstant`; see [R] [estimation options](#).

`lnalpha(varlist)` allows you to specify a linear equation for  $\ln\alpha$ . Specifying `lnalpha(male old)` means that  $\ln\alpha = \gamma_0 + \gamma_1\text{male} + \gamma_2\text{old}$ , where  $\gamma_0$ ,  $\gamma_1$ , and  $\gamma_2$  are parameters to be estimated along with the other model coefficients. If this option is not specified, `gnbreg` and `nbreg` will produce the same results because the shape parameter will be parameterized as a constant.

`exposure(varnamee)`, `offset(varnameo)`, `constraints(constraints)`, `collinear`; see [R] [estimation options](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

### Reporting

`level(#)`; see [R] [estimation options](#).

`irr` reports estimated coefficients transformed to incidence-rate ratios, that is,  $e^{\beta_i}$  rather than  $\beta_i$ . Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated or stored. `irr` may be specified at estimation or when replaying previously estimated results.

`nocnsreport`; see [R] [estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [R] [maximize](#). These options are seldom used.

Setting the optimization type to `technique(bhhh)` resets the default `vcetype` to `vce(opg)`.

The following option is available with `gnbreg` but is not shown in the dialog box:  
`coeflegend`; see [R] [estimation options](#).

## Remarks

Remarks are presented under the following headings:

*Introduction to negative binomial regression*  
*nbreg*  
*gnbreg*

## Introduction to negative binomial regression

Negative binomial regression models the number of occurrences (counts) of an event when the event has extra-Poisson variation, that is, when it has overdispersion. The Poisson regression model is

$$y_j \sim \text{Poisson}(\mu_j)$$

where

$$\mu_j = \exp(\mathbf{x}_j\boldsymbol{\beta} + \text{offset}_j)$$

for observed counts  $y_j$  with covariates  $\mathbf{x}_j$  for the  $j$ th observation. One derivation of the negative binomial mean-dispersion model is that individual units follow a Poisson regression model, but there is an omitted variable  $\nu_j$ , such that  $e^{\nu_j}$  follows a gamma distribution with mean 1 and variance  $\alpha$ :

$$y_j \sim \text{Poisson}(\mu_j^*)$$

where

$$\mu_j^* = \exp(\mathbf{x}_j\boldsymbol{\beta} + \text{offset}_j + \nu_j)$$

and

$$e^{\nu_j} \sim \text{Gamma}(1/\alpha, \alpha)$$

With this parameterization, a  $\text{Gamma}(a, b)$  distribution will have expectation  $ab$  and variance  $ab^2$ .

We refer to  $\alpha$  as the overdispersion parameter. The larger  $\alpha$  is, the greater the overdispersion. The Poisson model corresponds to  $\alpha = 0$ . `nbreg` parameterizes  $\alpha$  as  $\ln\alpha$ . `gnbreg` allows  $\ln\alpha$  to be modeled as  $\ln\alpha_j = \mathbf{z}_j\boldsymbol{\gamma}$ , a linear combination of covariates  $\mathbf{z}_j$ .

`nbreg` will fit two different parameterizations of the negative binomial model. The default, described above and also given by the `dispersion(mean)` option, has dispersion for the  $j$ th observation equal to  $1 + \alpha \exp(\mathbf{x}_j\boldsymbol{\beta} + \text{offset}_j)$ . This is seen by noting that the above implies that

$$\mu_j^* \sim \text{Gamma}(1/\alpha, \alpha\mu_j)$$

and thus

$$\begin{aligned} \text{Var}(y_j) &= E\{\text{Var}(y_j|\mu_j^*)\} + \text{Var}\{E(y_j|\mu_j^*)\} \\ &= E(\mu_j^*) + \text{Var}(\mu_j^*) \\ &= \mu_j(1 + \alpha\mu_j) \end{aligned}$$

The alternative parameterization, given by the `dispersion(constant)` option, has dispersion equal to  $1 + \delta$ ; that is, it is constant for all observations. This is so because the constant-dispersion model assumes instead that

$$\mu_j^* \sim \text{Gamma}(\mu_j/\delta, \delta)$$

and thus  $\text{Var}(y_j) = \mu_j(1 + \delta)$ . The Poisson model corresponds to  $\delta = 0$ .

For detailed derivations of both models, see [Cameron and Trivedi \(1998, 70–77\)](#). In particular, note that the mean-dispersion model is known as the NB2 model in their terminology, whereas the constant-dispersion model is referred to as the NB1 model.

See [Long and Freese \(2006\)](#) and [Cameron and Trivedi \(2009, chap. 17\)](#) for a discussion of the negative binomial regression model with Stata examples and for a discussion of other regression models for count data.

[Hilbe \(2007\)](#) provides an exclusive review of the negative binomial model and its variations, using Stata examples.

## nbreg

It is not uncommon to posit a Poisson regression model and observe a lack of model fit. The following data appeared in [Rodríguez \(1993\)](#):

```
. use http://www.stata-press.com/data/r11/rod93
. list
```

|     | cohort | age_mos | deaths | exposure |
|-----|--------|---------|--------|----------|
| 1.  | 1      | 0.5     | 168    | 278.4    |
| 2.  | 1      | 2.0     | 48     | 538.8    |
| 3.  | 1      | 4.5     | 63     | 794.4    |
| 4.  | 1      | 9.0     | 89     | 1,550.8  |
| 5.  | 1      | 18.0    | 102    | 3,006.0  |
| 6.  | 1      | 42.0    | 81     | 8,743.5  |
| 7.  | 1      | 90.0    | 40     | 14,270.0 |
| 8.  | 2      | 0.5     | 197    | 403.2    |
| 9.  | 2      | 2.0     | 48     | 786.0    |
| 10. | 2      | 4.5     | 62     | 1,165.3  |
| 11. | 2      | 9.0     | 81     | 2,294.8  |
| 12. | 2      | 18.0    | 97     | 4,500.5  |
| 13. | 2      | 42.0    | 103    | 13,201.5 |
| 14. | 2      | 90.0    | 39     | 19,525.0 |
| 15. | 3      | 0.5     | 195    | 495.3    |
| 16. | 3      | 2.0     | 55     | 956.7    |
| 17. | 3      | 4.5     | 58     | 1,381.4  |
| 18. | 3      | 9.0     | 85     | 2,604.5  |
| 19. | 3      | 18.0    | 87     | 4,618.5  |
| 20. | 3      | 42.0    | 70     | 9,814.5  |
| 21. | 3      | 90.0    | 10     | 5,802.5  |

```
. generate logexp = ln(exposure)
. poisson deaths i.cohort, offset(logexp)
Iteration 0: log likelihood = -2160.0544
Iteration 1: log likelihood = -2159.5162
Iteration 2: log likelihood = -2159.5159
Iteration 3: log likelihood = -2159.5159
```

```
Poisson regression Number of obs = 21
 LR chi2(2) = 49.16
 Prob > chi2 = 0.0000
Log likelihood = -2159.5159 Pseudo R2 = 0.0113
```

| deaths | Coef.     | Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|--------|-----------|-----------|--------|-------|----------------------|-----------|
| cohort |           |           |        |       |                      |           |
| 2      | -.3020405 | .0573319  | -5.27  | 0.000 | -.4144089            | -.1896721 |
| 3      | .0742143  | .0589726  | 1.26   | 0.208 | -.0413698            | .1897983  |
| _cons  | -3.899488 | .0411345  | -94.80 | 0.000 | -3.98011             | -3.818866 |
| logexp | (offset)  |           |        |       |                      |           |

```
. estat gof
 Goodness-of-fit chi2 = 4190.689
 Prob > chi2(18) = 0.0000
```

The extreme significance of the goodness-of-fit  $\chi^2$  indicates that the Poisson regression model is inappropriate, suggesting to us that we should try a negative binomial model:

```
. nbreg deaths i.cohort, offset(logexp) nolog
Negative binomial regression Number of obs = 21
 LR chi2(2) = 0.40
Dispersion = mean Prob > chi2 = 0.8171
Log likelihood = -131.3799 Pseudo R2 = 0.0015
```

| deaths   | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| cohort   |           |           |       |       |                      |           |
| 2        | -.2676187 | .7237203  | -0.37 | 0.712 | -1.686084            | 1.150847  |
| 3        | -.4573957 | .7236651  | -0.63 | 0.527 | -1.875753            | .9609618  |
| _cons    | -2.086731 | .511856   | -4.08 | 0.000 | -3.08995             | -1.083511 |
| logexp   | (offset)  |           |       |       |                      |           |
| /lnalpha | .5939963  | .2583615  |       |       | .0876171             | 1.100376  |
| alpha    | 1.811212  | .4679475  |       |       | 1.09157              | 3.005295  |

Likelihood-ratio test of alpha=0:   chibar2(01) = 4056.27 Prob>=chibar2 = 0.000

Our original Poisson model is a special case of the negative binomial—it corresponds to  $\alpha = 0$ . **nbreg**, however, estimates  $\alpha$  indirectly, estimating instead  $\ln\alpha$ . In our model,  $\ln\alpha = 0.594$ , meaning that  $\alpha = 1.81$  (**nbreg** undoes the transformation for us at the bottom of the output).

To test  $\alpha = 0$  (equivalent to  $\ln\alpha = -\infty$ ), **nbreg** performs a likelihood-ratio test. The staggering  $\chi^2$  value of 4,056 asserts that the probability that we would observe these data conditional on  $\alpha = 0$  is virtually zero, that is, conditional on the process being Poisson. The data are not Poisson. It is not accidental that this  $\chi^2$  value is close to the goodness-of-fit statistic from the Poisson regression itself.

❑ **Technical note**

The usual Gaussian test of  $\alpha = 0$  is omitted because this test occurs on the boundary, invalidating the usual theory associated with such tests. However, the likelihood-ratio test of  $\alpha = 0$  has been modified to be valid on the boundary. In particular, the null distribution of the likelihood-ratio test statistic is not the usual  $\chi^2_1$ , but rather a 50:50 mixture of a  $\chi^2_0$  (point mass at zero) and a  $\chi^2_1$ , denoted as  $\bar{\chi}^2_{01}$ . See [Gutierrez, Carter, and Drukker \(2001\)](#) for more details.



❑ **Technical note**

The negative binomial model deals with cases in which there is more variation than would be expected if the process were Poisson. The negative binomial model is not helpful if there is less than Poisson variation—if the variance of the count variable is less than its mean. However, underdispersion is uncommon. Poisson models arise because of independently generated events. Overdispersion comes about if some of the parameters (causes) of the Poisson processes are unknown. To obtain underdispersion, the sequence of events somehow would have to be regulated; that is, events would not be independent but controlled based on past occurrences.



# gnbreg

`gnbreg` is a generalization of `nbreg`, `dispersion(mean)`. Whereas in `nbreg`, one  $\ln\alpha$  is estimated, `gnbreg` allows  $\ln\alpha$  to vary, observation by observation, as a linear combination of another set of covariates:  $\ln\alpha_j = \mathbf{z}_j\boldsymbol{\gamma}$ .

We will assume that the number of deaths is a function of age, whereas the  $\ln\alpha$  parameter is a function of cohort. To fit the model, we type

```
. gnbreg deaths age_mos, lnalpna(i.cohort) offset(logexp)
Fitting constant-only model:
Iteration 0: log likelihood = -187.067 (not concave)
Iteration 1: log likelihood = -138.13047
Iteration 2: log likelihood = -136.05314
Iteration 3: log likelihood = -131.70022
Iteration 4: log likelihood = -131.58001
Iteration 5: log likelihood = -131.57948
Iteration 6: log likelihood = -131.57948
Fitting full model:
Iteration 0: log likelihood = -124.34327
Iteration 1: log likelihood = -117.74238
Iteration 2: log likelihood = -117.56306
Iteration 3: log likelihood = -117.56164
Iteration 4: log likelihood = -117.56164
Generalized negative binomial regression Number of obs = 21
 LR chi2(1) = 28.04
 Prob > chi2 = 0.0000
 Pseudo R2 = 0.1065

Log likelihood = -117.56164
```

| deaths  | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|---------|-----------|-----------|-------|-------|----------------------|-----------|
| deaths  |           |           |       |       |                      |           |
| age_mos | -.0516657 | .0051747  | -9.98 | 0.000 | -.061808             | -.0415233 |
| _cons   | -1.867225 | .2227944  | -8.38 | 0.000 | -2.303894            | -1.430556 |
| logexp  | (offset)  |           |       |       |                      |           |
| lnalpha |           |           |       |       |                      |           |
| cohort  |           |           |       |       |                      |           |
| 2       | .0939546  | .7187748  | 0.13  | 0.896 | -1.314818            | 1.502727  |
| 3       | .0815279  | .7365478  | 0.11  | 0.912 | -1.362079            | 1.525135  |
| _cons   | -.475958  | .5156503  | -0.92 | 0.356 | -1.486614            | .534698   |

We find that age is a significant determinant of the number of deaths. The standard errors for the variables in the  $\ln\alpha$  equation suggest that the overdispersion parameter does not vary across cohorts. We can test this assertion by typing

```
. test 2.cohort 3.cohort
(1) [lnalpha]2.cohort = 0
(2) [lnalpha]3.cohort = 0
 chi2(2) = 0.02
 Prob > chi2 = 0.9904
```

There is no evidence of variation by cohort in these data.

## □ Technical note

Note the intentional absence of a likelihood-ratio test for  $\alpha = 0$  in **gnbreg**. The test is affected by the same boundary condition that affects the comparison test in **nbreg**; however, when  $\alpha$  is parameterized by more than a constant term, the null distribution becomes intractable. For this reason, we recommend using **nbreg** to test for overdispersion and, if you have reason to believe that overdispersion exists, only then modeling the overdispersion using **gnbreg**.



## Saved results

**nbreg** and **gnbreg** save the following in **e()**:

### Scalars

|                      |                                                |
|----------------------|------------------------------------------------|
| <b>e(N)</b>          | number of observations                         |
| <b>e(k)</b>          | number of parameters                           |
| <b>e(k_aux)</b>      | number of auxiliary parameters                 |
| <b>e(k_eq)</b>       | number of equations                            |
| <b>e(k_eq_model)</b> | number of equations in model Wald test         |
| <b>e(k_dv)</b>       | number of dependent variables                  |
| <b>e(k_autoCns)</b>  | number of base, empty, and omitted constraints |
| <b>e(df_m)</b>       | model degrees of freedom                       |
| <b>e(r2_p)</b>       | pseudo- <i>R</i> -squared                      |
| <b>e(ll)</b>         | log likelihood                                 |
| <b>e(ll_0)</b>       | log likelihood, constant-only mode             |
| <b>e(ll_c)</b>       | log likelihood, comparison model               |
| <b>e(alpha)</b>      | value of alpha                                 |
| <b>e(delta)</b>      | value of delta                                 |
| <b>e(N_clust)</b>    | number of clusters                             |
| <b>e(chi2)</b>       | $\chi^2$                                       |
| <b>e(chi2_c)</b>     | $\chi^2$ for comparison test                   |
| <b>e(p)</b>          | significance                                   |
| <b>e(rank)</b>       | rank of <b>e(V)</b>                            |
| <b>e(rank0)</b>      | rank of <b>e(V)</b> for constant-only model    |
| <b>e(ic)</b>         | number of iterations                           |
| <b>e(rc)</b>         | return code                                    |
| <b>e(converged)</b>  | 1 if converged, 0 otherwise                    |



## Macros

|                                 |                                                                                 |
|---------------------------------|---------------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>nbreg</code> or <code>gnbreg</code>                                       |
| <code>e(cmdline)</code>         | command as typed                                                                |
| <code>e(depvar)</code>          | name of dependent variable                                                      |
| <code>e(wtype)</code>           | weight type                                                                     |
| <code>e(wexp)</code>            | weight expression                                                               |
| <code>e(title)</code>           | title in estimation output                                                      |
| <code>e(clustvar)</code>        | name of cluster variable                                                        |
| <code>e(offset)</code>          | offset ( <code>nbreg</code> )                                                   |
| <code>e(offset1)</code>         | offset ( <code>gnbreg</code> )                                                  |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                         |
| <code>e(chi2_ct)</code>         | Wald or LR; type of model $\chi^2$ test corresponding to <code>e(chi2_c)</code> |
| <code>e(dispers)</code>         | mean or constant                                                                |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                                  |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                                   |
| <code>e(diparm#)</code>         | display transformed parameter #                                                 |
| <code>e(diparm_opt2)</code>     | options for displaying transformed parameters                                   |
| <code>e(opt)</code>             | type of optimization                                                            |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization        |
| <code>e(ml_method)</code>       | type of ml method                                                               |
| <code>e(user)</code>            | name of likelihood-evaluator program                                            |
| <code>e(technique)</code>       | maximization technique                                                          |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular                     |
| <code>e(crittype)</code>        | optimization criterion                                                          |
| <code>e(properties)</code>      | b V                                                                             |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                                  |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>                  |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>                  |

## Matrices

|                              |                                                                               |
|------------------------------|-------------------------------------------------------------------------------|
| <code>e(b)</code>            | coefficient vector                                                            |
| <code>e(Cns)</code>          | constraints matrix                                                            |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)                                           |
| <code>e(gradient)</code>     | gradient vector                                                               |
| <code>e(V)</code>            | variance–covariance matrix of the estimators                                  |
| <code>e(V_modelbased)</code> | model-based variance                                                          |
| <code>e(ml_h)</code>         | derivative tolerance, $(\text{abs}(b)+1e-3)*1e-3$ ( <code>gnbreg</code> only) |
| <code>e(ml_scale)</code>     | derivative scale factor ( <code>gnbreg</code> only)                           |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`nbreg` and `gnbreg` are implemented as ado-files.

See [R] [poisson](#) and Johnson, Kemp, and Kotz (2005, chap. 4) for an introduction to the Poisson distribution.

Methods and formulas are presented under the following headings:

*Mean-dispersion model*  
*Constant-dispersion model*

## Mean-dispersion model

A negative binomial distribution can be regarded as a gamma mixture of Poisson random variables. The number of times something occurs,  $y_j$ , is distributed as  $\text{Poisson}(\nu_j \mu_j)$ . That is, its conditional likelihood is

$$f(y_j | \nu_j) = \frac{(\nu_j \mu_j)^{y_j} e^{-\nu_j \mu_j}}{\Gamma(y_j + 1)}$$

where  $\mu_j = \exp(\mathbf{x}_j \boldsymbol{\beta} + \text{offset}_j)$  and  $\nu_j$  is an unobserved parameter with a  $\text{Gamma}(1/\alpha, \alpha)$  density:

$$g(\nu) = \frac{\nu^{(1-\alpha)/\alpha} e^{-\nu/\alpha}}{\alpha^{1/\alpha} \Gamma(1/\alpha)}$$

This gamma distribution has mean 1 and variance  $\alpha$ , where  $\alpha$  is our ancillary parameter.

The unconditional likelihood for the  $j$ th observation is therefore

$$f(y_j) = \int_0^\infty f(y_j | \nu) g(\nu) d\nu = \frac{\Gamma(m + y_j)}{\Gamma(y_j + 1) \Gamma(m)} p_j^m (1 - p_j)^{y_j}$$

where  $p_j = 1/(1 + \alpha \mu_j)$  and  $m = 1/\alpha$ . Solutions for  $\alpha$  are handled by searching for  $\ln \alpha$  because  $\alpha$  is required to be greater than zero.

The log likelihood (with weights  $w_j$  and offsets) is given by

$$m = 1/\alpha \quad p_j = 1/(1 + \alpha \mu_j) \quad \mu_j = \exp(\mathbf{x}_j \boldsymbol{\beta} + \text{offset}_j)$$

$$\begin{aligned} \ln L = \sum_{j=1}^n w_j & \left[ \ln\{\Gamma(m + y_j)\} - \ln\{\Gamma(y_j + 1)\} \right. \\ & \left. - \ln\{\Gamma(m)\} + m \ln(p_j) + y_j \ln(1 - p_j) \right] \end{aligned}$$

For `gnbreg`,  $\alpha$  can vary across the observations according to the parameterization  $\ln \alpha_j = \mathbf{z}_j \boldsymbol{\gamma}$ .

## Constant-dispersion model

The constant-dispersion model assumes that  $y_j$  is conditionally distributed as  $\text{Poisson}(\mu_j^*)$ , where  $\mu_j^* \sim \text{Gamma}(\mu_j/\delta, \delta)$  for some dispersion parameter  $\delta$  (by contrast, the mean-dispersion model assumes that  $\mu_j^* \sim \text{Gamma}(1/\alpha, \alpha \mu_j)$ ). The log likelihood is given by

$$m_j = \mu_j/\delta \quad p = 1/(1 + \delta)$$

$$\begin{aligned} \ln L = \sum_{j=1}^n w_j & \left[ \ln\{\Gamma(m_j + y_j)\} - \ln\{\Gamma(y_j + 1)\} \right. \\ & \left. - \ln\{\Gamma(m_j)\} + m_j \ln(p) + y_j \ln(1 - p) \right] \end{aligned}$$

with everything else defined as before in the calculations for the mean-dispersion model.

Maximization for `gnbreg` is done via the `lf` linear-form method, and for `nbreg` it is done via the `e2` method described in [R] [ml](#).

`nbreg` and `gnbreg` support the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

These commands also support estimation with survey data. For details on VCEs with survey data, see [SVY] [variance estimation](#).

## References

- Cameron, A. C., and P. K. Trivedi. 1998. *Regression Analysis of Count Data*. Cambridge: Cambridge University Press.
- . 2009. *Microeconometrics Using Stata*. College Station, TX: Stata Press.
- Deb, P., and P. K. Trivedi. 2006. Maximum simulated likelihood estimation of a negative binomial regression model with multinomial endogenous treatment. *Stata Journal* 6: 246–255.
- Gutierrez, R. G., S. Carter, and D. M. Drukker. 2001. [sg160: On boundary-value likelihood-ratio tests](#). *Stata Technical Bulletin* 60: 15–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 269–273. College Station, TX: Stata Press.
- Hilbe, J. M. 1998. [sg91: Robust variance estimators for MLE Poisson and negative binomial regression](#). *Stata Technical Bulletin* 45: 26–28. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 177–180. College Station, TX: Stata Press.
- . 1999. [sg102: Zero-truncated Poisson and negative binomial regression](#). *Stata Technical Bulletin* 47: 37–40. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 233–236. College Station, TX: Stata Press.
- . 2007. *Negative Binomial Regression*. New York: Cambridge University Press.
- Johnson, N. L., A. W. Kemp, and S. Kotz. 2005. *Univariate Discrete Distributions*. 3rd ed. New York: Wiley.
- Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.
- Long, J. S., and J. Freese. 2001. Predicted probabilities for count models. *Stata Journal* 1: 51–57.
- . 2006. *Regression Models for Categorical Dependent Variables Using Stata*. 2nd ed. College Station, TX: Stata Press.
- Miranda, A., and S. Rabe-Hesketh. 2006. Maximum likelihood estimation of endogenous switching and sample selection models for binary, ordinal, and count variables. *Stata Journal* 6: 285–308.
- Rodríguez, G. 1993. [sbe10: An improvement to poisson](#). *Stata Technical Bulletin* 11: 11–14. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 94–98. College Station, TX: Stata Press.
- Rogers, W. H. 1991. [sbe1: Poisson regression with rates](#). *Stata Technical Bulletin* 1: 11–12. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 62–64. College Station, TX: Stata Press.
- . 1993. [sg16.4: Comparison of nbreg and glm for negative binomial](#). *Stata Technical Bulletin* 16: 7. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 82–84. College Station, TX: Stata Press.

## Also see

- [R] [nbreg postestimation](#) — Postestimation tools for `nbreg` and `gnbreg`
- [R] [glm](#) — Generalized linear models
- [R] [poisson](#) — Poisson regression
- [R] [zinb](#) — Zero-inflated negative binomial regression
- [R] [ztnb](#) — Zero-truncated negative binomial regression
- [SVY] [svy estimation](#) — Estimation commands for survey data
- [XT] [xtnbreg](#) — Fixed-effects, random-effects, & population-averaged negative binomial models
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `nbreg` and `gnbreg`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>            | link test for model specification                                                                   |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic noffset]

predict [type] { stub*|newvarreg newvardisp } [if] [in] , scores
```

| statistic         | description                                                          |
|-------------------|----------------------------------------------------------------------|
| Main              |                                                                      |
| <code>n</code>    | number of events; the default                                        |
| <code>ir</code>   | incidence rate (equivalent to <code>predict ... , n noffset</code> ) |
| <code>xb</code>   | linear prediction                                                    |
| <code>stdp</code> | standard error of the linear prediction                              |

In addition, relevant only after `gnbreg` are the following:

| <i>statistic</i>     | <i>description</i>                        |
|----------------------|-------------------------------------------|
| Main                 |                                           |
| <code>alpha</code>   | predicted values of $\alpha_j$            |
| <code>lnalpha</code> | predicted values of $\ln\alpha_j$         |
| <code>stdplna</code> | standard error of predicted $\ln\alpha_j$ |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

`n`, the default, calculates the predicted number of events, which is  $\exp(\mathbf{x}_j\beta)$  if neither `offset(varnameo)` nor `exposure(varnamee)` was specified when the model was fit;  $\exp(\mathbf{x}_j\beta + \text{offset}_j)$  if `offset()` was specified; or  $\exp(\mathbf{x}_j\beta) \times \text{exposure}_j$  if `exposure()` was specified.

`ir` calculates the incidence rate  $\exp(\mathbf{x}_j\beta)$ , which is the predicted number of events when exposure is 1. This is equivalent to specifying both `n` and `nooffset` options.

`xb` calculates the linear prediction, which is  $\mathbf{x}_j\beta$  if neither `offset()` nor `exposure()` was specified;  $\mathbf{x}_j\beta + \text{offset}_j$  if `offset()` was specified; or  $\mathbf{x}_j\beta + \ln(\text{exposure}_j)$  if `exposure()` was specified; see `nooffset` below.

`stdp` calculates the standard error of the linear prediction.

`alpha`, `lnalpha`, and `stdplna` are relevant after `gnbreg` estimation only; they produce the predicted values of  $\alpha_j$ ,  $\ln\alpha_j$ , and the standard error of the predicted  $\ln\alpha_j$ , respectively.

`nooffset` is relevant only if you specified `offset()` or `exposure()` when you fit the model. It modifies the calculations made by `predict` so that they ignore the offset or exposure variable; the linear prediction is treated as  $\mathbf{x}_j\beta$  rather than as  $\mathbf{x}_j\beta + \text{offset}_j$  or  $\mathbf{x}_j\beta + \ln(\text{exposure}_j)$ . Specifying `predict ...`, `nooffset` is equivalent to specifying `predict ...`, `ir`.

`scores` calculates equation-level score variables.

The first new variable will contain  $\partial \ln L / \partial (\mathbf{x}_j\beta)$ .

The second new variable will contain  $\partial \ln L / \partial (\ln\alpha_j)$  for `dispersion(mean)` and `gnbreg`.

The second new variable will contain  $\partial \ln L / \partial (\ln\delta)$  for `dispersion(constant)`.

Remarks

After nbreg and gnbreg, predict returns the predicted number of events:

```
. use http://www.stata-press.com/data/r11/rod93
. nbreg deaths i.cohort, nolog
Negative binomial regression
 Number of obs = 21
 LR chi2(2) = 0.14
Dispersion = mean Prob > chi2 = 0.9307
Log likelihood = -108.48841 Pseudo R2 = 0.0007
```

| deaths   | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| cohort   |           |           |       |       |                      |           |
| 2        | .0591305  | .2978419  | 0.20  | 0.843 | -.5246289            | .64289    |
| 3        | -.0538792 | .2981621  | -0.18 | 0.857 | -.6382662            | .5305077  |
| _cons    | 4.435906  | .2107213  | 21.05 | 0.000 | 4.0229               | 4.848912  |
| /lnalpha | -1.207379 | .3108622  |       |       | -1.816657            | -.5980999 |
| alpha    | .29898    | .0929416  |       |       | .1625683             | .5498555  |

Likelihood-ratio test of alpha=0:  chibar2(01) =  434.62 Prob>=chibar2 = 0.000

```
. predict count
(option n assumed; predicted number of events)
. summarize deaths count
```

| Variable | Obs | Mean     | Std. Dev. | Min | Max      |
|----------|-----|----------|-----------|-----|----------|
| deaths   | 21  | 84.66667 | 48.84192  | 10  | 197      |
| count    | 21  | 84.66667 | 4.00773   | 80  | 89.57143 |

Methods and formulas

All postestimation commands listed above are implemented as ado-files.

In the following, we use the same notation as in [R] [nbreg](#).

Methods and formulas are presented under the following headings:

- [Mean-dispersion model](#)
- [Constant-dispersion model](#)

Mean-dispersion model

The equation-level scores are given by

score( $\mathbf{x}\beta$ ) $_j$  =  $p_j(y_j - \mu_j)$

score( $\tau$ ) $_j$  =  $-m \left\{ \frac{\alpha_j(\mu_j - y_j)}{1 + \alpha_j\mu_j} - \ln(1 + \alpha_j\mu_j) + \psi(y_j + m) - \psi(m) \right\}$

where  $\tau_j = \ln\alpha_j$  and  $\psi(z)$  is the digamma function.

## Constant-dispersion model

The equation-level scores are given by

$$\begin{aligned}\text{score}(\mathbf{x}\beta)_j &= m_j \{ \psi(y_j + m_j) - \psi(m_j) + \ln(p) \} \\ \text{score}(\tau)_j &= y_j - (y_j + m_j)(1 - p) - \text{score}(\mathbf{x}\beta)_j\end{aligned}$$

where  $\tau_j = \ln \delta_j$ .

## Also see

[R] [nbreg](#) — Negative binomial regression

[U] [20 Estimation and postestimation commands](#)

Syntax

Standard estimation command syntax

```
nestreg [, options]: command_name depvar (varlist) [(varlist) ...]
[if] [in] [weight] [command_options]
```

Survey estimation command syntax

```
nestreg [, options]: svy [vcetype] [, svy_options]: command_name depvar
(varlist) [(varlist) ...] [if] [in] [, command_options]
```

| <i>options</i>                  | description                                                         |
|---------------------------------|---------------------------------------------------------------------|
| Reporting                       |                                                                     |
| <code>waldtable</code>          | report Wald test results; the default                               |
| <code>lrtable</code>            | report likelihood-ratio test results                                |
| <code>quietly</code>            | suppress any output from <i>command_name</i>                        |
| <code>store(<i>stub</i>)</code> | store nested estimation results in <code>_est_</code> <i>stub</i> # |

by is allowed; see [U] 11.1.10 Prefix commands.  
Weights are allowed if *command\_name* allows them; see [U] 11.1.6 weight.  
A *varlist* in parentheses indicates that this list of variables is to be considered as a block. Each variable in a *varlist* not bound in parentheses will be treated as its own block.  
All postestimation commands behave as they would after *command\_name* without the `nestreg` prefix; see the postestimation manual entry for *command\_name*.

Menu

Statistics > Other > Nested model statistics

Description

`nestreg` fits nested models by sequentially adding blocks of variables and then reports comparison tests between the nested models.

Options

|                                                                                                                                                                                                                                                                                                                                  |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Reporting                                                                                                                                                                                                                                                                                                                        |
| <code>waldtable</code> specifies that the table of Wald test results be reported. <code>waldtable</code> is the default.                                                                                                                                                                                                         |
| <code>lrtable</code> specifies that the table of likelihood-ratio tests be reported. This option is not allowed if <code>pweights</code> , the <code>vce(robust)</code> option, or the <code>vce(cluster <i>clustvar</i>)</code> option is specified. <code>lrtable</code> is also not allowed with the <code>svy</code> prefix. |
| <code>quietly</code> suppresses the display of any output from <i>command_name</i> .                                                                                                                                                                                                                                             |



`store(stub)` specifies that each model fit by `nestreg` be stored under the name `_est_stub#`, where `#` is the nesting order from first to last.

## Remarks

Remarks are presented under the following headings:

[Estimation commands](#)  
[Wald tests](#)  
[Likelihood-ratio tests](#)  
[Programming for nestreg](#)

## Estimation commands

`nestreg` removes collinear predictors and observations with missing values from the estimation sample before calling `command_name`.

The following Stata commands are supported by `nestreg`:

|                       |                      |                      |
|-----------------------|----------------------|----------------------|
| <code>clogit</code>   | <code>nbreg</code>   | <code>regress</code> |
| <code>cloglog</code>  | <code>ologit</code>  | <code>scobit</code>  |
| <code>glm</code>      | <code>oprobit</code> | <code>stcox</code>   |
| <code>intreg</code>   | <code>poisson</code> | <code>stcrreg</code> |
| <code>logistic</code> | <code>probit</code>  | <code>streg</code>   |
| <code>logit</code>    | <code>qreg</code>    | <code>tobit</code>   |

You do not supply a `depvar` for `streg`, `stcox`, or `stcrreg`; otherwise, `depvar` is required. You must supply two `depvars` for `intreg`.

## Wald tests

Use `nestreg` to test the significance of blocks of predictors, building the regression model one block at a time. Using the data from [example 1](#) of [\[R\] test](#), we wish to test the significance of the following predictors of birth rate: `medage`, `medagesq`, and `region` (already partitioned into four indicator variables: `reg1`, `reg2`, `reg3`, and `reg4`).

```
. use http://www.stata-press.com/data/r11/census4
(birth rate, median age)
. nestreg: regress brate (medage) (medagesq) (reg2-reg4)
```

Block 1: medage

| Source   | SS         | df | MS         | Number of obs = 50     |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | 32675.1044 | 1  | 32675.1044 | F( 1, 48) = 164.72     |  |  |
| Residual | 9521.71561 | 48 | 198.369075 | Prob > F = 0.0000      |  |  |
|          |            |    |            | R-squared = 0.7743     |  |  |
|          |            |    |            | Adj R-squared = 0.7696 |  |  |
| Total    | 42196.82   | 49 | 861.159592 | Root MSE = 14.084      |  |  |

| brate  | Coef.     | Std. Err. | t      | P> t  | [95% Conf. Interval] |           |
|--------|-----------|-----------|--------|-------|----------------------|-----------|
| medage | -15.24893 | 1.188141  | -12.83 | 0.000 | -17.63785            | -12.86002 |
| _cons  | 618.3935  | 35.15416  | 17.59  | 0.000 | 547.7113             | 689.0756  |

|                   |            |           |            |                        |                      |          |
|-------------------|------------|-----------|------------|------------------------|----------------------|----------|
| Block 2: medagesq |            |           |            |                        |                      |          |
| Source            | SS         | df        | MS         | Number of obs = 50     |                      |          |
| Model             | 36755.8524 | 2         | 18377.9262 | F( 2, 47) = 158.75     |                      |          |
| Residual          | 5440.96755 | 47        | 115.765267 | Prob > F = 0.0000      |                      |          |
|                   |            |           |            | R-squared = 0.8711     |                      |          |
|                   |            |           |            | Adj R-squared = 0.8656 |                      |          |
| Total             | 42196.82   | 49        | 861.159592 | Root MSE = 10.759      |                      |          |
| brate             | Coef.      | Std. Err. | t          | P> t                   | [95% Conf. Interval] |          |
| medage            | -109.8925  | 15.96663  | -6.88      | 0.000                  | -142.0132            | -77.7718 |
| medagesq          | 1.607332   | .2707228  | 5.94       | 0.000                  | 1.062708             | 2.151956 |
| _cons             | 2007.071   | 235.4316  | 8.53       | 0.000                  | 1533.444             | 2480.698 |

|                         |            |           |            |                        |                      |           |
|-------------------------|------------|-----------|------------|------------------------|----------------------|-----------|
| Block 3: reg2 reg3 reg4 |            |           |            |                        |                      |           |
| Source                  | SS         | df        | MS         | Number of obs = 50     |                      |           |
| Model                   | 38803.419  | 5         | 7760.68381 | F( 5, 44) = 100.63     |                      |           |
| Residual                | 3393.40095 | 44        | 77.1227489 | Prob > F = 0.0000      |                      |           |
|                         |            |           |            | R-squared = 0.9196     |                      |           |
|                         |            |           |            | Adj R-squared = 0.9104 |                      |           |
| Total                   | 42196.82   | 49        | 861.159592 | Root MSE = 8.782       |                      |           |
| brate                   | Coef.      | Std. Err. | t          | P> t                   | [95% Conf. Interval] |           |
| medage                  | -109.0957  | 13.52452  | -8.07      | 0.000                  | -136.3526            | -81.83886 |
| medagesq                | 1.635208   | .2290536  | 7.14       | 0.000                  | 1.173581             | 2.096835  |
| reg2                    | 15.00284   | 4.252068  | 3.53       | 0.001                  | 6.433365             | 23.57233  |
| reg3                    | 7.366435   | 3.953336  | 1.86       | 0.069                  | -.6009898            | 15.33386  |
| reg4                    | 21.39679   | 4.650602  | 4.60       | 0.000                  | 12.02412             | 30.76946  |
| _cons                   | 1947.61    | 199.8405  | 9.75       | 0.000                  | 1544.858             | 2350.362  |

| Block | Block F | Block df | Residual df | Pr > F | R2     | Change in R2 |
|-------|---------|----------|-------------|--------|--------|--------------|
| 1     | 164.72  | 1        | 48          | 0.0000 | 0.7743 |              |
| 2     | 35.25   | 1        | 47          | 0.0000 | 0.8711 | 0.0967       |
| 3     | 8.85    | 3        | 44          | 0.0001 | 0.9196 | 0.0485       |

This single call to nestreg ran regress three times, adding a block of predictors to the model for each run as in

|                        |            |           |            |                        |                      |           |
|------------------------|------------|-----------|------------|------------------------|----------------------|-----------|
| . regress brate medage |            |           |            |                        |                      |           |
| Source                 | SS         | df        | MS         | Number of obs = 50     |                      |           |
| Model                  | 32675.1044 | 1         | 32675.1044 | F( 1, 48) = 164.72     |                      |           |
| Residual               | 9521.71561 | 48        | 198.369075 | Prob > F = 0.0000      |                      |           |
|                        |            |           |            | R-squared = 0.7743     |                      |           |
|                        |            |           |            | Adj R-squared = 0.7696 |                      |           |
| Total                  | 42196.82   | 49        | 861.159592 | Root MSE = 14.084      |                      |           |
| brate                  | Coef.      | Std. Err. | t          | P> t                   | [95% Conf. Interval] |           |
| medage                 | -15.24893  | 1.188141  | -12.83     | 0.000                  | -17.63785            | -12.86002 |
| _cons                  | 618.3935   | 35.15416  | 17.59      | 0.000                  | 547.7113             | 689.0756  |

```
. regress brate medage medagesq
```

| Source   | SS         | df | MS         | Number of obs = 50     |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | 36755.8524 | 2  | 18377.9262 | F( 2, 47) = 158.75     |  |  |
| Residual | 5440.96755 | 47 | 115.765267 | Prob > F = 0.0000      |  |  |
|          |            |    |            | R-squared = 0.8711     |  |  |
|          |            |    |            | Adj R-squared = 0.8656 |  |  |
| Total    | 42196.82   | 49 | 861.159592 | Root MSE = 10.759      |  |  |

| brate    | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|----------|-----------|-----------|-------|-------|----------------------|----------|
| medage   | -109.8925 | 15.96663  | -6.88 | 0.000 | -142.0132            | -77.7718 |
| medagesq | 1.607332  | .2707228  | 5.94  | 0.000 | 1.062708             | 2.151956 |
| _cons    | 2007.071  | 235.4316  | 8.53  | 0.000 | 1533.444             | 2480.698 |

```
. regress brate medage medagesq reg2-reg4
```

| Source   | SS         | df | MS         | Number of obs = 50     |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | 38803.419  | 5  | 7760.68381 | F( 5, 44) = 100.63     |  |  |
| Residual | 3393.40095 | 44 | 77.1227489 | Prob > F = 0.0000      |  |  |
|          |            |    |            | R-squared = 0.9196     |  |  |
|          |            |    |            | Adj R-squared = 0.9104 |  |  |
| Total    | 42196.82   | 49 | 861.159592 | Root MSE = 8.782       |  |  |

| brate    | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| medage   | -109.0957 | 13.52452  | -8.07 | 0.000 | -136.3526            | -81.83886 |
| medagesq | 1.635208  | .2290536  | 7.14  | 0.000 | 1.173581             | 2.096835  |
| reg2     | 15.00284  | 4.252068  | 3.53  | 0.001 | 6.433365             | 23.57233  |
| reg3     | 7.366435  | 3.953336  | 1.86  | 0.069 | -.6009898            | 15.33386  |
| reg4     | 21.39679  | 4.650602  | 4.60  | 0.000 | 12.02412             | 30.76946  |
| _cons    | 1947.61   | 199.8405  | 9.75  | 0.000 | 1544.858             | 2350.362  |

`nestreg` collected the  $F$  statistic for the corresponding block of predictors and the model  $R^2$  statistic from each model fit.

The  $F$  statistic for the first block, 164.72, is for a test of the joint significance of the first block of variables; it is simply the  $F$  statistic from the regression of `brate` on `medage`. The  $F$  statistic for the second block, 35.25, is for a test of the joint significance of the second block of variables in a regression of both the first and second blocks of variables. In our example, it is an  $F$  test of `medagesq` in the regression of `brate` on `medage` and `medagesq`. Similarly, the third block's  $F$  statistic of 8.85 corresponds to a joint test of `reg2`, `reg3`, and `reg4` in the final regression.

## Likelihood-ratio tests

The `nestreg` command provides a simple syntax for performing likelihood-ratio tests for nested model specifications; also see `lrtest`. Using the data from [example 1](#) of [\[R\] lrtest](#), we wish to jointly test the significance of the following predictors of low birthweight: `age`, `lwt`, `ptl`, and `ht`.

```
. use http://www.stata-press.com/data/r11/lbw
(Hosmer & Lemeshow data)
. xi: nestreg, lr: logistic low (i.race smoke ui) (age lwt ptl ht)
i.race _Irace_1-3 (naturally coded; _Irace_1 omitted)
Block 1: _Irace_2 _Irace_3 smoke ui
Logistic regression
Number of obs = 189
LR chi2(4) = 18.80
Prob > chi2 = 0.0009
Pseudo R2 = 0.0801
Log likelihood = -107.93404
```

| low      | Odds Ratio | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|----------|------------|-----------|------|-------|----------------------|----------|
| _Irace_2 | 3.052746   | 1.498087  | 2.27 | 0.023 | 1.166747             | 7.987382 |
| _Irace_3 | 2.922593   | 1.189229  | 2.64 | 0.008 | 1.316457             | 6.488285 |
| smoke    | 2.945742   | 1.101838  | 2.89 | 0.004 | 1.415167             | 6.131715 |
| ui       | 2.419131   | 1.047359  | 2.04 | 0.041 | 1.035459             | 5.651788 |

```
Block 2: age lwt ptl ht
Logistic regression
Number of obs = 189
LR chi2(8) = 33.22
Prob > chi2 = 0.0001
Pseudo R2 = 0.1416
Log likelihood = -100.724
```

| low      | Odds Ratio | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|----------|------------|-----------|-------|-------|----------------------|----------|
| _Irace_2 | 3.534767   | 1.860737  | 2.40  | 0.016 | 1.259736             | 9.918406 |
| _Irace_3 | 2.368079   | 1.039949  | 1.96  | 0.050 | 1.001356             | 5.600207 |
| smoke    | 2.517698   | 1.00916   | 2.30  | 0.021 | 1.147676             | 5.523162 |
| ui       | 2.1351     | .9808153  | 1.65  | 0.099 | .8677528             | 5.2534   |
| age      | .9732636   | .0354759  | -0.74 | 0.457 | .9061578             | 1.045339 |
| lwt      | .9849634   | .0068217  | -2.19 | 0.029 | .9716834             | .9984249 |
| ptl      | 1.719161   | .5952579  | 1.56  | 0.118 | .8721455             | 3.388787 |
| ht       | 6.249602   | 4.322408  | 2.65  | 0.008 | 1.611152             | 24.24199 |

| Block | LL       | LR    | df | Pr > LR | AIC      | BIC      |
|-------|----------|-------|----|---------|----------|----------|
| 1     | -107.934 | 18.80 | 4  | 0.0009  | 225.8681 | 242.0768 |
| 2     | -100.724 | 14.42 | 4  | 0.0061  | 219.448  | 248.6237 |

The estimation results from the full model are left in `e()`, so we can later use `estat` and other postestimation commands.

```
. estat gof
Logistic model for low, goodness-of-fit test
number of observations = 189
number of covariate patterns = 182
Pearson chi2(173) = 179.24
Prob > chi2 = 0.3567
```

Programming for nestreg

If you want your user-written command (*command\_name*) to work with `nestreg`, it must follow standard Stata syntax and allow the `if` qualifier. Furthermore, *command\_name* must have `sw` or `swml` as a program property; see [\[P\] program properties](#). If *command\_name* has `swml` as a property, *command\_name* must save the log-likelihood value in `e(ll)` and the model degrees of freedom in `e(df_m)`.

## Saved results

`nestreg` saves the following in `r()`:

Matrices

|                      |                                                    |
|----------------------|----------------------------------------------------|
| <code>r(wald)</code> | matrix corresponding to the Wald table             |
| <code>r(lr)</code>   | matrix corresponding to the likelihood-ratio table |

## Methods and formulas

`nestreg` is implemented as an ado-file.

## Acknowledgment

We thank Paul H. Bern, Princeton University, for developing the hierarchical regression command that inspired `nestreg`.

## Reference

Acock, A. C. 2008. *A Gentle Introduction to Stata*. 2nd ed. College Station, TX: Stata Press.

## Also see

[P] [program properties](#) — Properties of user-defined programs

# Title

**net** — Install and manage user-written additions from the Internet

## Syntax

*Set current location for net*

```
net from directory_or_url
```

*Change to a different net directory*

```
net cd path_or_url
```

*Change to a different net site*

```
net link linkname
```

*Search for installed packages*

```
net search (see \[R\] net search)
```

*Report current net location*

```
net
```

*Describe a package*

```
net describe pkgname [, from(directory_or_url)]
```

*Set location where packages will be installed*

```
net set ado dirname
```

*Set location where ancillary files will be installed*

```
net set other dirname
```

*Report net ‘from’, ‘ado’, and ‘other’ settings*

```
net query
```

*Install ado-files and help files from a package*

```
net install pkgname [, all replace force from(directory_or_url)]
```

*Install ancillary files from a package*

```
net get pkgname [, all replace force from(directory_or_url)]
```

Shortcut to access Stata Journal (SJ) net site

```
net sj vol-issue [insert]
```

Shortcut to access Stata Technical Bulletin (STB) net site

```
net stb issue [insert]
```

List installed packages

```
ado [, find(string) from(dirname)]
ado dir [pkgid] [, find(string) from(dirname)]
```

Describe installed packages

```
ado describe [pkgid] [, find(string) from(dirname)]
```

Uninstall an installed package

```
ado uninstall pkgid [, from(dirname)]
```

where

```
pkgname is name of a package
pkgid is name of a package
 or a number in square brackets: [#]
dirname is a directory name
 or PLUS (default)
 or PERSONAL
 or SITE
```

## Description

**net** downloads and installs additions to Stata. The additions can be obtained from the Internet or from media. The additions can be ado-files (new commands), help files, or even datasets. Collections of files are bound together into *packages*. For instance, the package named *zz49* might add the *xyz* command to Stata. At a minimum, such a package would contain *xyz.ado*, the code to implement the new command, and *xyz.sthlp*, the online help to describe it. That the package contains two files is a detail: you use **net** to download the package *zz49*, regardless of the number of files.

**ado** manages the packages you have installed by using **net**. The **ado** command lets you list and uninstall previously installed packages.

You can also access the **net** and **ado** features by selecting **Help > SJ and User-written Programs**; this is the recommended method to find and install additions to Stata.

## Options

**all** is used with **net install** and **net get**. Typing it with either one makes the command equivalent to typing **net install** followed by **net get**.

`replace` is for use with `net install` and `net get`. It specifies that the downloaded files replace existing files if any of the files already exists.

`force` specifies that the downloaded files replace existing files if any of the files already exists, even if Stata thinks all the files are the same. `force` implies `replace`.

`find(string)` is for use with `ado`, `ado dir`, and `ado describe`. It specifies that the descriptions of the packages installed on your computer be searched, and that the package descriptions containing `string` be listed.

`from(dirname)`, when used with `ado`, specifies where the packages are installed. The default is `from(PLUS)`. `PLUS` is a code word that Stata understands to correspond to a particular directory on your computer that was set at installation time. On Windows computers, `PLUS` probably means the directory `c:\ado\plus`, but it might mean something else. You can find out what it means by typing `sysdir`, but doing so is irrelevant if you use the defaults.

`from(directory_or_url)`, when used with `net`, specifies the directory or URL where installable packages may be found. The directory or URL is the same as the one that would have been specified with `net from`.

## Remarks

For an introduction to using `net` and `ado`, see [U] [28 Using the Internet to keep up to date](#). The purpose of this documentation is

- to briefly, but accurately, describe `net` and `ado` and all their features and
- to provide documentation to those who wish to set up their own sites to distribute additions to Stata.

Remarks are presented under the following headings:

*Definition of a package*

*The purpose of the net and ado commands*

*Content pages*

*Package-description pages*

*Where packages are installed*

*A summary of the net command*

*A summary of the ado command*

*Relationship of net and ado to the point-and-click interface*

*Creating your own site*

*Format of content and package-description files*

*Example 1*

*Example 2*

*Additional package directives*

*SMCL in content and package-description files*

*Error-free file delivery*

## Definition of a package

A *package* is a collection of files—typically, `.ado` and `.sthlp` files—that together provide a new feature in Stata. Packages contain additions that you wish had been part of Stata at the outset. We write such additions, and so do other users.

One source of these additions is the *Stata Journal*, a printed and electronic journal with corresponding software. If you want the journal, you must subscribe, but the software is available for free from our web site.



## The purpose of the net and ado commands

The `net` command makes it easy to distribute and install packages. The goal is to get you quickly to a package-description page that summarizes the addition, for example,

---

```
. net describe rte_stat, from(http://www.wemakeitupaswego.edu/faculty/sgazer/)
```

---

```
package rte_stat from http://www.wemakeitupaswego.edu/faculty/sgazer/
```

---

### TITLE

rte\_stat. The robust-to-everything statistic; update.

### DESCRIPTION/AUTHOR(S)

S. Gazer, Dept. of Applied Theoretical Mathematics, WMIUAWG Univ. Aleph-0 100% confidence intervals proved too conservative for some applications; Aleph-1 confidence intervals have been substituted. The new robust-to-everything supplants the previous robust-to-everything-conceivable statistic. See "Inference in the absence of data" (forthcoming). After installation, see help `rte`.

### INSTALLATION FILES

(type `net install rte_stat`)

rte.ado  
rte.sthlp  
nullset.ado  
random.ado

---

If you decide that the addition might prove useful, `net` makes the installation easy:

```
. net install rte_stat
checking rte_stat consistency and verifying not already installed...
installing into c:\ado\plus\ ...
installation complete.
```

The `ado` command helps you manage packages installed with `net`. Perhaps you remember that you installed a package that calculates the robust-to-everything statistic, but you cannot remember the command's name. You could use `ado` to search what you have previously installed for the `rte` command,

```
. ado
[1] package sg145 from http://www.stata.com/stb/stb56
 STB-56 sg145. Scalar measures of fit for regression models.
 (output omitted)
[15] package rte_stat from http://www.wemakeitupaswego.edu/faculty/sgazer/
 rte_stat. The robust-to-everything statistic; update.
 (output omitted)
[21] package st0119 from http://www.stata-journal.com/software/sj7-1
 SJ7-1 st0119. Rasch analysis
```

or you might type

```
. ado, find("robust-to-everything")
[15] package rte_stat from http://www.wemakeitupaswego.edu/faculty/sgazer/
 rte_stat. The robust-to-everything statistic; update.
```

Perhaps you decide that `rte`, despite the author's claims, is not worth the disk space it occupies. You can use `ado` to erase it:

```
. ado uninstall rte_stat
package rte_stat from http://www.wemakeitupaswego.edu/faculty/sgazer/
 rte_stat. The robust-to-everything statistic; update.
(package uninstalled)
```

`ado uninstall` is easier than erasing the files by hand because `ado uninstall` erases every file associated with the package, and, moreover, `ado` knows where on your computer `rte_stat` is installed; you would have to hunt for these files.

## Content pages

There are two types of pages displayed by `net`: content pages and package-description pages. When you type `net from`, `net cd`, `net link`, or `net` without arguments, Stata goes to the specified place and displays the content page:

```
. net from http://www.stata.com
```

---

```
http://www.stata.com/
StataCorp
```

---

```
Welcome to StataCorp.
```

```
Below we provide links to sites providing additions to Stata, including
the Stata Journal, STB, and Statalist. These are NOT THE OFFICIAL UPDATES;
you fetch and install the official updates by typing -update-.
```

```
PLACES you could -net link- to:
```

```
 sj The Stata Journal
```

```
DIRECTORIES you could -net cd- to:
```

```
 stb materials published in the Stata Technical Bulletin
 users materials written by various people, including StataCorp
 employees
 meetings software packages from Stata Users Group meetings
 links links to other locations providing additions to Stata
```

---

A content page tells you about other content pages and package-description pages. The example above lists other content pages only. Below we follow one of the links for the *Stata Journal*:

```
. net link sj
```

---

```
http://www.stata-journal.com/
The Stata Journal
```

---

```
The Stata Journal is a refereed, quarterly journal containing articles
of interest to Stata users. For more details and subscription information,
visit the Stata Journal web site at http://www.stata-journal.com.
```

```
PLACES you could -net link- to:
```

```
 stata StataCorp web site
```

```
DIRECTORIES you could -net cd- to:
```

```
 production Files for authors of the Stata Journal
 software Software associated with Stata Journal articles
```

---

---

```
. net cd software
```

---

```
http://www.stata-journal.com/software/
The Stata Journal
```

---

```
PLACES you could -net link- to:
```

|       |                                                 |
|-------|-------------------------------------------------|
| stata | StataCorp web site                              |
| stb   | Stata Technical Bulletin (STB) software archive |

```
DIRECTORIES you could -net cd- to:
```

|                  |                   |
|------------------|-------------------|
| (output omitted) |                   |
| sj7-1            | volume 7, issue 1 |
| (output omitted) |                   |
| sj1-1            | volume 1, issue 1 |

---

```
. net cd sj7-1
```

---

```
http://www.stata-journal.com/software/sj7-1/
Stata Journal volume 7, issue 1
```

---

```
DIRECTORIES you could -net cd- to:
```

|    |                      |
|----|----------------------|
| .. | Other Stata Journals |
|----|----------------------|

```
PACKAGES you could -net describe-:
```

|        |                                                                                             |
|--------|---------------------------------------------------------------------------------------------|
| dm0027 | File filtering in Stata: handling complex data formats and navigating log files efficiently |
| st0119 | Rasch analysis                                                                              |
| st0120 | Multivariable regression spline models                                                      |
| st0121 | mhbounds - Sensitivity Analysis for Average Treatment Effects                               |

---

dm0027, st0119, ..., st0121 are links to package-description pages.

The links for the *Stata Technical Bulletin* (STB) follow steps similar to those in the *Stata Journal* example above.

```
. net from http://www.stata.com
(output omitted)
. net cd stb
```

---

```
http://www.stata.com/stb/
The Stata Technical Bulletin
```

---

```
PLACES you could -net link- to:
```

|       |                    |
|-------|--------------------|
| stata | StataCorp web site |
|-------|--------------------|

```
DIRECTORIES you could -net cd- to:
```

|                  |                    |
|------------------|--------------------|
| (output omitted) |                    |
| stb54            | STB-54, March 2000 |
| (output omitted) |                    |

---

```
. net cd stb54
```

---

```
http://www.stata.com/stb/stb54/
STB-54 March 2000
```

---

```
DIRECTORIES you could -net cd- to:
```

|    |            |
|----|------------|
| .. | Other STBs |
|----|------------|

```
PACKAGES you could -net describe-:
```

```
(output omitted)
```

---

1. When you type `net from`, you follow that with a location to display the location's content page.
  - a. The location could be a URL, such as <http://www.stata.com>. The content page at that location would then be listed.
  - b. The location could be `e:` on a Windows computer or a mounted volume on a Mac computer. The content page on that source would be listed. That would work if you had special media obtained from StataCorp or special media prepared by another user.
  - c. The location could even be a directory on your computer, but that would work only if that directory contained the right kind of files.
2. Once you have specified a location, typing `net cd` will take you into subdirectories of that location, if there are any. Typing

```
. net from http://www.stata-journal.com
. net cd software
```

is equivalent to typing

```
. net from http://www.stata-journal.com/software
```

Typing `net cd` displays the content page from that location.

3. Typing `net` without arguments redisplay the current content page, which is the content page last displayed.
4. `net link` is similar to `net cd` in that the result is to change the location, but rather than changing to subdirectories of the current location, `net link` jumps to another location:

```
. net from http://www.stata-journal.com
```

---

```
http://www.stata-journal.com/
The Stata Journal
```

---

The Stata Journal is a refereed, quarterly journal containing articles of interest to Stata users. For more details and subscription information, visit the Stata Journal web site at <http://www.stata-journal.com>.

PLACES you could **-net link-** to:

|       |                    |
|-------|--------------------|
| stata | StataCorp web site |
|-------|--------------------|

DIRECTORIES you could **-net cd-** to:

|            |                                                 |
|------------|-------------------------------------------------|
| production | Files for authors of the Stata Journal          |
| software   | Software associated with Stata Journal articles |

---

Typing `net link stata` would jump to <http://www.stata.com>:

```
. net link stata
```

---

```
http://www.stata.com/
StataCorp
```

---

Welcome to StataCorp.  
(output omitted)

---

## Package-description pages

Package-description pages describe what could be installed:

```
. net from http://www.stata-journal.com/software/sj7-1
```

---

```
http://www.stata-journal.com/software/sj7-1/
(output omitted)
```

```
. net describe st0119
```

---

```
package st0119 from http://www.stata-journal.com/software/sj7-1
```

---

### TITLE

SJ7-1 st0119. Rasch analysis

### DESCRIPTION/AUTHOR(S)

Rasch analysis

by Jean-Benoit Hardouin, University of Nantes, France

Support: jean-benoit.hardouin@univ-nantes.fr

After installation, type help **gammasy**, **gauss****hermite**,  
**geekel2d**, **raschtest**, and **raschtestv7**

### INSTALLATION FILES

(type **net install st0119**)

```
st0119/raschtest.ado
st0119/raschtest.hlp
st0119/raschtestv7.ado
st0119/raschtestv7.hlp
st0119/gammasy.ado
st0119/gammasy.hlp
st0119/gausshermite.ado
st0119/gausshermite.hlp
st0119/geekel2d.ado
st0119/geekel2d.hlp
```

### ANCILLARY FILES

(type **net get st0119**)

```
st0119/data.dta
st0119/outrasch.do
```

---

A package-description page describes the package and tells you how to install the component files. Package-description pages potentially describe two types of files:

1. Installation files: files that you type **net install** to install and that are required to make the addition work.
2. Ancillary files: additional files that you might want to install—you type **net get** to install them—but that you can ignore. Ancillary files are typically datasets that are useful for demonstration purposes. Ancillary files are not really installed in the sense of being copied to an official place for use by Stata itself. They are merely copied into the current directory so that you may use them if you wish.

You install the official files by typing **net install** followed by the package name. For example, to install **st0119**, you would type

```
. net install st0119
checking st0119 consistency and verifying not already installed...
installing into c:\ado\plus\ ...
installation complete.
```

You get the ancillary files—if there are any and if you want them—by typing **net get** followed by the package name:

```
. net get st0119
checking st0119 consistency and verifying not already installed...
copying into current directory...
 copying data.dta
 copying outrasch.do
ancillary files successfully copied.
```

Most users ignore the ancillary files.

Once you have installed a package—by typing `net install`—use `ado` to redisplay the package-description page whenever you wish:

```
. ado describe st0119
```

---

```
[1] package st0119 from http://www.stata-journal.com/software/sj7-1
```

---

#### **TITLE**

SJ7-1 st0119. Rasch analysis

#### **DESCRIPTION/AUTHOR(S)**

Rasch analysis  
by Jean-Benoit Hardouin, University of Nantes, France  
Support: [jean-benoit.hardouin@univ-nantes.fr](mailto:jean-benoit.hardouin@univ-nantes.fr)  
After installation, type help **gammasy**, **gausshermite**,  
**geekel2d**, **raschtest**, and **raschtestv7**

#### **INSTALLATION FILES**

r/raschtest.ado  
r/raschtest.hlp  
r/raschtestv7.ado  
r/raschtestv7.hlp  
g/gammasy.ado  
g/gammasy.hlp  
g/gausshermite.ado  
g/gausshermite.hlp  
g/geekel2d.ado  
g/geekel2d.hlp

#### **INSTALLED ON**

24 Apr 2007

---

The package-description page shown by `ado` includes the location from which we got the package and when we installed it. It does not mention the ancillary files that were originally part of this package because they are not tracked by `ado`.

## Where packages are installed

Packages should be installed in `PLUS` or `SITE`, which are code words that Stata understands and that correspond to some real directories on your computer. Typing `sysdir` will tell you where these are, if you care.

```
. sysdir
STATA: C:\Program Files\Stata11\
UPDATES: C:\Program Files\Stata11\ado\updates\
BASE: C:\Program Files\Stata11\ado\base\
SITE: C:\Program Files\Stata11\ado\site\
PLUS: c:\ado\plus\
PERSONAL: c:\ado\personal\
OLDPLACE: c:\ado\
```

If you type `sysdir`, you may obtain different results.

By default, `net` installs in the `PLUS` directory, and `ado` tells you about what is installed there. If you are on a multiple-user system, you may wish to install some packages in the `SITE` directory. This way, they will be available to other Stata users. To do that, before using `net install`, type

```
. net set ado SITE
```

and when reviewing what is installed or removing packages, redirect `ado` to that directory:

```
. ado ..., from(SITE)
```

In both cases, you type `SITE` because Stata will understand that `SITE` means the site `ado`-directory as defined by `sysdir`. To install into `SITE`, you must have write access to that directory.

If you reset where `net` installs and then, in the same session, wish to install into your private `ado`-directory, type

```
. net set ado PLUS
```

That is how things were originally. If you are confused as to where you are, type `net query`.

## A summary of the `net` command

The `net` command displays content pages and package-description pages. Such pages are provided over the Internet, and most users get them there. We recommend that you start at <http://www.stata.com> and work out from there. We also recommend using `net search` to find packages of interest to you; see [R] [net search](#).

`net from` moves you to a location and displays the content page.

`net cd` and `net link` change from your current location to other locations. `net cd` enters subdirectories of the original location. `net link` jumps from one location to another, depending on the code on the content page.

`net describe` lists a package-description page. Packages are named, and you type `net describe pkgname`.

`net install` installs a package into your copy of Stata. `net get` copies any additional files (ancillary files) to your current directory.

`net sj` and `net stb` simplify loading files from the *Stata Journal* and its predecessor, the *Stata Technical Bulletin*.

```
net sj vol-issue
```

is a synonym for typing

```
net from http://www.stata-journal.com/software/sjvol-issue
```

whereas

```
net sj vol-issue insert
```

is a synonym for typing

```
net from http://www.stata-journal.com/software/sjvol-issue
net describe insert
```

`net set` controls where `net` installs files. By default, `net` installs in the `PLUS` directory; see [P] [sysdir](#). `net set ado SITE` would cause subsequent `net` commands to install in the `SITE` directory. `net set other` sets where ancillary files, such as `.dta` files, are installed. The default is the current directory.

`net query` displays the current `net from`, `net set ado`, and `net set other` settings.

## A summary of the ado command

The `ado` command lists the package descriptions of previously installed packages.

Typing `ado` without arguments is the same as typing `ado dir`. Both list the names and titles of the packages you have installed.

`ado describe` lists full package-description pages.

`ado uninstall` removes packages from your computer.

Because you can install packages from a variety of sources, the package names may not always be unique. Thus the packages installed on your computer are numbered sequentially, and you may refer to them by name or by number. For instance, say that you wanted to get rid of the robust-to-everything statistic command you installed. Type

```
. ado, find("robust-to-everything")
[15] package rte_stat from http://www.wemakeitupaswego.edu/faculty/sgazer
rte_stat. The robust-to-everything statistic; update.
```

You could then type

```
. ado uninstall rte_stat
```

or

```
. ado uninstall [15]
```

Typing `ado uninstall rte_stat` would work only if the name `rte_stat` were unique; otherwise, `ado` would refuse, and you would have to type the number.

The `find()` option is allowed with `ado dir` and `ado describe`. It searches the package description for the word or phrase you specify, ignoring case (alpha matches Alpha). The complete package description is searched, including the author's name and the name of the files. Thus if `rte` was the name of a command that you wanted to eliminate, but you could not remember the name of the package, you could type

```
. ado, find(rte)
[15] package rte_stat from http://www.wemakeitupaswego.edu/faculty/sgazer
rte_stat. The robust-to-everything statistic; update.
```

## Relationship of net and ado to the point-and-click interface

Users may instead select **Help > SJ and User-written Programs**. There are advantages and disadvantages:

1. Flipping through content and package-description pages is easier; it is much like a browser. See [GS] **19 Updating and extending Stata—Internet functionality** ([GSM](#), [GSU](#), or [GSW](#)).
2. When browsing a product-description page, note that the `.sthlp` files are highlighted. You may click on `.sthlp` files to review them before installing the package.
3. You may not redirect from where `ado` searches for files.



## Creating your own site

The rest of this entry concerns how to create your own site to distribute additions to Stata. The idea is that you have written additions for use with Stata—say, `xyz.ado` and `xyz.sthlp`—and you wish to put them out so that coworkers or researchers at other institutions can easily install them. Or, perhaps you just have a dataset that you and others want to share.

In any case, all you need is a homepage. You place the files that you want to distribute on your homepage (or in a subdirectory), and you add two more files—a content file and a package-description file—and you are done.

## Format of content and package-description files

The content file describes the content page. It must be named `stata.toc`:

---

```

begin stata.toc
OFF (to make site unavailable temporarily)
* lines starting with * are comments; they are ignored
* blank lines are ignored, too
* v indicates version—specify v 3; old-style toc files do not have this
v 3
* d lines display description text
* the first d line is the title, and the remaining ones are text
* blank d lines display a blank line
d title
d text
d text
d
...
* l lines display links
l word-to-show path-or-url [description]
l word-to-show path-or-url [description]
...
* t lines display other directories within the site
t path [description]
t path [description]
...
* p lines display packages
p pkgname [description]
p pkgname [description]
...
end stata.toc

```

---

Package files describe packages and are named `pkgname.pkg`:

---

```

* lines starting with * are comments; they are ignored
* blank lines are ignored, too
* v indicates version—specify v 3; old-style pkg files do not have this
v 3
* d lines display package description text
* the first d line is the title, and the remaining ones are text
* blank d lines display a blank line
d title
d text
d Distribution-Date: date
d text
d
...
* f identifies the component files
f [path/]filename [description]
f [path/]filename [description]
...
* e line is optional; it means stop reading
e

```

---

Note the Distribution-Date description line. This line is optional but recommended. Stata can look for updates to user-written programs with the `adoupdate` command if the package files from which those programs were installed contain a Distribution-Date description line.

## Example 1

Say that we want the user to see the following:

---

```

. net from http://www.university.edu/~me

```

---

```

http://www.university.edu/~me
Chris Farrar, Uni University

```

---

```

PACKAGES you could -net describe-:
 xyz interval-truncated survival
. net describe xyz

```

---

```

package xyz from http://www.university.edu/~me

```

---

```

TITLE
 xyz. interval-truncated survival.
DESCRIPTION/AUTHOR(S)
 C. Farrar, Uni University.
INSTALLATION FILES (type net install xyz)
 xyz.ado
 xyz.sthlp
ANCILLARY FILES (type net get xyz)
 sample.dta

```

---

The files needed to do this would be

---

```

v 3
d Chris Farrar, Uni University
p xyz interval-truncated survival

```

---

---

```

v 3
d xyz. interval-truncated survival.
d C. Farrar, Uni University.
f xyz.ado
f xyz.sthlp
f sample.dta

```

---

```

end xyz.pkg

```

---

On his homepage, Chris would place the following files:

|            |                                                             |
|------------|-------------------------------------------------------------|
| stata.toc  | (shown above)                                               |
| xyz.pkg    | (shown above)                                               |
| xyz.ado    | file to be delivered (for use by <code>net install</code> ) |
| xyz.sthlp  | file to be delivered (for use by <code>net install</code> ) |
| sample.dta | file to be delivered (for use by <code>net get</code> )     |

Chris does nothing to distinguish ancillary files from installation files.

## Example 2

S. Gazer wants to create a more complex site:

```
. net from http://www.wemakeitupaswego.edu/faculty/sgazer
```

---

```
http://www.wemakeitupaswego.edu/faculty/sgazer
```

```
Data-free inference materials
```

---

S. Gazer, Department of Applied Theoretical Mathematics

Also see my homepage for the preprint of "Irrefutable inference".

PLACES you could **-net link-** to:

|       |                    |
|-------|--------------------|
| stata | StataCorp web site |
|-------|--------------------|

DIRECTORIES you could **-net cd-** to:

|    |                                                   |
|----|---------------------------------------------------|
| ir | irrefutable inference programs (work in progress) |
|----|---------------------------------------------------|

PACKAGES you could **-net describe-**:

|      |                                            |
|------|--------------------------------------------|
| rtec | Robust-to-everything-conceivable statistic |
| rte  | Robust-to-everything statistic             |

*(Continued on next page)*

---

```
. net describe rte
```

---

```
package rte from http://www.wemakeitupaswego.edu/faculty/sgazer/
```

---

**TITLE**

```
rte. The robust-to-everything statistic; update.
```

**DESCRIPTION/AUTHOR(S)**

```
S. Gazer, Dept. of Applied Theoretical Mathematics, WMIUAWG Univ.
Aleph-0 100% confidence intervals proved too conservative for some
applications; Aleph-1 confidence intervals have been substituted.
The new robust-to-everything supplants the previous robust-to-
everything-conceivable statistic. See "Inference in the absence
of data" (forthcoming). After installation, see help rte.
```

```
Distribution-Date: 20070420
```

```
Support: email sgazer@wemakeitupaswego.edu
```

**INSTALLATION FILES**

```
(type net install rte_stat)
```

```
rte.ado
rte.sthlp
nullset.ado
random.ado
```

**ANCILLARY FILES**

```
(type net get rte_stat)
```

```
empty.dta
```

---

The files needed to do this would be

---

```
begin stata.toc
v 3
d Data-free inference materials
d S. Gazer, Department of Applied Theoretical Mathematics
d
d Also see my homepage for the preprint of "Irrefutable inference".
l stata http://www.stata.com
t ir irrefutable inference programs (work in progress)
p rtec Robust-to-everything-conceivable statistic
p rte Robust-to-everything statistic
end stata.toc
```

---



---

```
begin rte.pkg
v 3
d rte. The robust-to-everything statistic; update.
d {bf:S. Gazer, Dept. of Applied Theoretical Mathematics, WMIUAWG Univ.}
d Aleph-0 100% confidence intervals proved too conservative for some
d applications; Aleph-1 confidence intervals have been substituted.
d The new robust-to-everything supplants the previous robust-to-
d everything-conceivable statistic. See "Inference in the absence
d of data" (forthcoming). After installation, see help {bf:rte}.
d
d Distribution-Date: 20070420
d
d Support: email sgazer@wemakeitupaswego.edu
f rte.ado
f rte.sthlp
f nullset.ado
f random.ado
f empty.dta
end rte.pkg
```

---

On his homepage, Mr. Gazer would place the following files:

|                           |                                                                  |
|---------------------------|------------------------------------------------------------------|
| <code>stata.toc</code>    | (shown above)                                                    |
| <code>rte.pkg</code>      | (shown above)                                                    |
| <code>rte.ado</code>      | (file to be delivered)                                           |
| <code>rte.sthlp</code>    | (file to be delivered)                                           |
| <code>nullset.ado</code>  | (file to be delivered)                                           |
| <code>random.ado</code>   | (file to be delivered)                                           |
| <code>empty.dta</code>    | (file to be delivered)                                           |
| <code>rtec.pkg</code>     | the other package referred to in <code>stata.toc</code>          |
| <code>rtec.ado</code>     | the corresponding files to be delivered                          |
| <code>rtec.sthlp</code>   |                                                                  |
| <code>ir/stata.toc</code> | the contents file for when the user types <code>net cd ir</code> |
| <code>ir/...</code>       | whatever other <code>.pkg</code> files are referred to           |
| <code>ir/...</code>       | whatever other files are to be delivered                         |

If Mr. Gazer later updated the `rte` package, he could change the Distribution-Date description line in his package. Then, if someone who had previously installed the `rte` packaged wanted to obtain the latest version, that person could use the `adoupdate` command; see [\[R\] adoupdate](#).

For complex sites, a different structure may prove more convenient:

|                              |                                                                  |
|------------------------------|------------------------------------------------------------------|
| <code>stata.toc</code>       | (shown above)                                                    |
| <code>rte.pkg</code>         | (shown above)                                                    |
| <code>rtec.pkg</code>        | the other package referred to in <code>stata.toc</code>          |
| <code>rte/</code>            | directory containing <code>rte</code> files to be delivered:     |
| <code>rte/rte.ado</code>     | (file to be delivered)                                           |
| <code>rte/rte.sthlp</code>   | (file to be delivered)                                           |
| <code>rte/nullset.ado</code> | (file to be delivered)                                           |
| <code>rte/random.ado</code>  | (file to be delivered)                                           |
| <code>rte/empty.dta</code>   | (file to be delivered)                                           |
| <code>rtec/</code>           | directory containing <code>rtec</code> files to be delivered:    |
| <code>rtec/...</code>        | (files to be delivered)                                          |
| <code>ir/stata.toc</code>    | the contents file for when the user types <code>net cd ir</code> |
| <code>ir/*.pkg</code>        | whatever other package files are referred to                     |
| <code>ir/*/...</code>        | whatever other files are to be delivered                         |

If you prefer this structure, it is simply a matter of changing the bottom of the `rte.pkg` from

```
f rte.ado
f rte.sthlp
f nullset.ado
f random.ado
f empty.dta
```

to

```
f rte/rte.ado
f rte/rte.sthlp
f rte/nullset.ado
f rte/random.ado
f rte/empty.dta
```

In writing paths and files, the directory separator forward slash (/) is used, regardless of operating system, because this is what the Internet uses.

It does not matter whether the files you put out are in Windows, Mac, or Unix format (how lines end is recorded differently). When Stata reads the files over the Internet, it will figure out the file format on its own and will automatically translate the files to what is appropriate for the receiver.

## Additional package directives

**F** *filename* is similar to **f** *filename*, except that, when the file is installed, it will always be copied to the system directories (and not the current directory).

With **f** *filename*, the file is installed into a directory according to the file's suffix. For instance, `xyz.ado` would be installed in the system directories, whereas `xyz.dta` would be installed in the current directory.

Coding **F** `xyz.ado` would have the same result as coding **f** `xyz.ado`.

Coding **F** `xyz.dta`, however, would state that `xyz.dta` is to be installed in the system directories.

**g** *platformname filename* is also a variation on **f** *filename*. It specifies that the file be installed only if the user's operating system is of type *platformname*; otherwise, the file is ignored. The platform names are WIN (32-bit x86), WIN64A (64-bit x86-64), and WIN64I (64-bit Itanium) for Windows; MACINTEL (32-bit Intel, GUI), OSX.X86 (32-bit Intel, console), MACINTEL64 (64-bit Intel, GUI), OSX.X8664 (64-bit Intel, console), MAC (32-bit PowerPC), and OSX.PPC (32-bit PowerPC), for Mac; and AIX, LINUX (32-bit x86), LINUX64 (64-bit x86-64), LINUX64I (64-bit Itanium), SOL64, and SOLX8664 (64-bit x86-64) for Unix.

**G** *platformname filename* is a variation on **F** *filename*. The file, if not ignored, is to be installed in the system directories.

**g** *platformname filename1 filename2* is a more detailed version of **g** *platformname filename*. In this case, *filename1* is the name of the file on the server (the file to be copied), and *filename2* is to be the name of the file on the user's system; e.g., you might code

```
g WIN mydll.forwin mydll.plugin
g LINUX mydll.forlinux mydll.plugin
```

When you specify one *filename*, the result is the same as specifying two identical *filenames*.

**G** *platformname filename1 filename2* is the install-in-system-directories version of **g** *platformname filename1 filename2*.

**h** *filename* asserts that *filename* must be loaded, or this package is not to be installed; e.g., you might code

```
g WIN mydll.forwin mydll.plugin
g LINUX mydll.forlinux mydll.plugin
h mydll.plugin
```

if you were offering the plugin `mydll.plugin` for Windows and Linux only.

## SMCL in content and package-description files

The text listed on the second and subsequent **d** lines in both `stata.toc` and `pkgname.pkg` may contain SMCL as long as you include **v** 3; see [\[P\] smcl](#).

Thus, in `rte.pkg`, S. Gazer coded the third line as

```
d {bf:S. Gazer, Dept. of Applied Theoretical Mathematics, WMIUAWG Univ.}
```

## Error-free file delivery

Most people transport files over the Internet and never worry about the file being corrupted in the process because corruption rarely occurs. If, however, the files must be delivered perfectly or not at all, you can include checksum files in the directory.

For instance, say that `big.dta` is included in your package and that it must be sent perfectly. First, use Stata to make the checksum file for `big.dta`

```
. checksum big.dta, save
```

That command creates a small file called `big.sum`; see [\[D\] checksum](#). Then copy both `big.dta` and `big.sum` to your homepage. If `set checksum` is on (the default is off), whenever Stata reads *filename.whatever* over the net, it also looks for *filename.sum*. If it finds such a file, it uses the information recorded in it to verify that what was copied was error free.

If you do this, be cautious. If you put `big.dta` and `big.sum` on your homepage and then later change `big.dta` without changing `big.sum`, people will think that there are transmission errors when they try to download `big.dta`.

## References

- Baum, C. F., and N. J. Cox. 1999. [ip29: Metadata for user-written contributions to the Stata programming language](#). *Stata Technical Bulletin* 52: 10–12. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 121–124. College Station, TX: Stata Press.
- Cox, N. J., and C. F. Baum. 2000. [ip29.1: Metadata for user-written contributions to the Stata programming language](#). *Stata Technical Bulletin* 54: 21–22. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 124–126. College Station, TX: Stata Press.

## Also see

- [\[R\] adoupdate](#) — Update user-written ado-files
- [\[R\] net search](#) — Search the Internet for installable packages
- [\[R\] search](#) — Search Stata documentation
- [\[R\] sj](#) — Stata Journal and STB installation instructions
- [\[R\] ssc](#) — Install and uninstall packages from SSC
- [\[D\] checksum](#) — Calculate checksum of file
- [\[P\] smcl](#) — Stata Markup and Control Language
- [\[R\] update](#) — Update Stata
- [\[GSM\] 19 Updating and extending Stata—Internet functionality](#)
- [\[GSU\] 19 Updating and extending Stata—Internet functionality](#)
- [\[GSW\] 19 Updating and extending Stata—Internet functionality](#)
- [\[U\] 28 Using the Internet to keep up to date](#)

Syntax

```
net search word [word ...] [, options]
```

| options    | description                                                    |
|------------|----------------------------------------------------------------|
| or         | list packages that contain any of the keywords; default is all |
| nosj       | search non-SJ and non-STB sources                              |
| tocpkg     | search both tables of contents and packages; the default       |
| toc        | search tables of contents only                                 |
| pkg        | search packages only                                           |
| everywhere | search packages for match                                      |
| filenames  | search filenames associated with package for match             |
| errnone    | make return code 111 instead of 0 when no matches found        |

Description

`net search` searches the Internet for user-written additions to Stata, including, but not limited to, user-written additions published in the *Stata Journal* (SJ) and in the *Stata Technical Bulletin* (STB). `net search` lists the available additions that contain the specified keywords.

The user-written materials found are available for immediate download by using the `net` command or by clicking on the link.

In addition to typing `net search`, you may select **Help > Search...** and choose **Search net resources**. This is the recommended way to search for user-written additions to Stata.

Options

`or` is relevant only when multiple keywords are specified. By default, `net search` lists only packages that include all the keywords. `or` changes the command to list packages that contain any of the keywords.

`nosj` specifies that `net search` not list matches that were published in the SJ or in the STB.

`tocpkg`, `toc`, and `pkg` determine what is searched. `tocpkg` is the default, meaning that both tables of contents (tocs) and packages (pkgs) are searched. `toc` restricts the search to tables of contents. `pkg` restricts the search to packages.

`everywhere` and `filenames` determine where in packages `net search` looks for *keywords*. The default is `everywhere`. `filenames` restricts `net search` to search for matches only in the filenames associated with a package. Specifying `everywhere` implies `pkg`.

`errnone` is a programmer’s option that causes the return code to be 111 instead of 0 when no matches are found.



## Remarks

`net search` searches the Internet for user-written additions to Stata. If you want to search the Stata documentation for a particular topic, command, or author, see [\[R\] search](#). `net search word [word ...]` (without options) is equivalent to typing `search word [word ...]`, `net`.

Remarks are presented under the following headings:

*Topic searches*  
*Author searches*  
*Command searches*  
*Where does net search look?*  
*How does net search work?*

## Topic searches

### Example: Find what is available about random effects

```
. net search random effect
```

Comments:

- It is best to search using the singular form of a word. `net search random effect` will find both “random effect” and “random effects”.
- `net search random effect` will also find “random-effect” because `net search` performs a string search and not a word search.
- `net search random effect` lists all packages containing the words “random” and “effect”, not necessarily used together.
- If you wanted all packages containing the word “random” or the word “effect”, you would type `net search random effect, or`.

## Author searches

### Example: Find what is available by author Jeroen Weesie

```
. net search weesie
```

Comments:

- You could type `net search jeroen weesie`, but that might list fewer results because sometimes the last name is used without the first.
- You could type `net search Weesie`, but it would not matter. Capitalization is ignored in the search.

### Example: Find what is available by Jeroen Weesie, excluding SJ and STB materials

```
. net search weesie, nosj
```

- The SJ and the STB tend to dominate search results because so much has been published in them. If you know that what you are looking for is not in the SJ or in the STB, specifying the `nosj` option will narrow the search.
- `net search weesie` lists everything that `net search weesie, nosj` lists, and more. If you just type `net search weesie`, look down the list. SJ and STB materials are listed first, and non-SJ and non-STB materials are listed last.

## Command searches

### Example: Find the user-written command `kursus`

```
. net search kursus, file
```

- You could just type `net search kursus`, and that will list everything `net search kursus`, `file` lists, and more. Because you know `kursus` is a command, however, there must be a `kursus.ado` file associated with the package. Typing `net search kursus, file` narrows the search.
- You could also type `net search kursus.ado, file` to narrow the search even more.

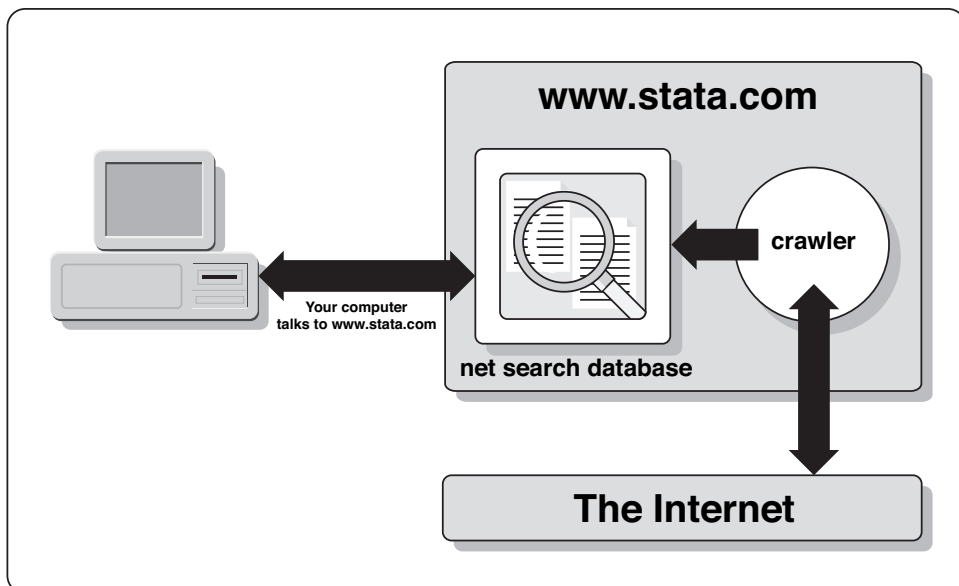
## Where does net search look?

`net search` looks everywhere, not just at <http://www.stata.com>.

`net search` begins by looking at <http://www.stata.com>, but then follows every link, which takes it to other places, and then follows every link again, which takes it to even more places, and so on.

Authors: Please let us know if you have a site that we should include in our search by sending an email to [webmaster@stata.com](mailto:webmaster@stata.com). We will then link to your site from ours to ensure that `net search` finds your materials. That is not strictly necessary, however, as long as your site is directly or indirectly linked from some site that is linked to ours.

## How does net search work?



Our website maintains a database of Stata resources. When you use `net search`, it contacts <http://www.stata.com> with your request, <http://www.stata.com> searches its database, and Stata returns the results to you.

Another part of the system is called the crawler, which searches the web for new Stata resources to add to the `net search` database and verifies that the resources already found are still available. When a new resource becomes available, the crawler takes about 2 days to add it to the database, and, similarly, if a resource disappears, the crawler takes roughly 2 days to remove it from the database.

## References

- Baum, C. F., and N. J. Cox. 1999. [ip29: Metadata for user-written contributions to the Stata programming language](#). *Stata Technical Bulletin* 52: 10–12. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 121–124. College Station, TX: Stata Press.
- Cox, N. J., and C. F. Baum. 2000. [ip29.1: Metadata for user-written contributions to the Stata programming language](#). *Stata Technical Bulletin* 54: 21–22. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 124–126. College Station, TX: Stata Press.
- Gould, W. W., and A. R. Riley. 2000. [stata55: Search web for installable packages](#). *Stata Technical Bulletin* 54: 4–6. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 10–13. College Station, TX: Stata Press.

## Also see

- [R] [net](#) — Install and manage user-written additions from the Internet
- [R] [ssc](#) — Install and uninstall packages from SSC
- [R] [sj](#) — Stata Journal and STB installation instructions
- [R] [hsearch](#) — Search help files
- [R] [search](#) — Search Stata documentation
- [R] [adoupdate](#) — Update user-written ado-files
- [R] [update](#) — Update Stata

## Syntax

*Turn on or off the use of a proxy server*

```
set httpproxy {on | off} [, init]
```

*Set proxy host name*

```
set httpproxyhost ["name"]
```

*Set the proxy port number*

```
set httpproxyport #
```

*Turn on or off proxy authorization*

```
set httpproxyauth {on | off}
```

*Set proxy authorization user ID*

```
set httpproxyuser ["name"]
```

*Set proxy authorization password*

```
set httpproxypw ["password"]
```

*Set time limit for establishing initial connection*

```
set timeout1 #seconds [, permanently]
```

*Set time limit for data transfer*

```
set timeout2 #seconds [, permanently]
```

## Description

Several commands (e.g., `net`, `news`, and `update`) are designed specifically for use over the Internet. Many other Stata commands that read a file (e.g., `copy`, `type`, and `use`) can also read directly from a URL. All these commands will usually work without your ever needing to concern yourself with the `set` commands discussed here. These `set` commands provide control over network system parameters.

If you experience problems when using Stata's network features, ask your system administrator if your site uses a proxy. A proxy is a server between your computer and the rest of the Internet, and your computer may need to communicate with other computers on the Internet through this proxy. If your site uses a proxy, your system administrator can provide you with its host name and the port your computer can use to communicate with it. If your site's proxy requires you to log in to it before it will respond, your system administrator will provide you with a user ID and password.

`set httpproxyhost` sets the name of the host to be used as a proxy server. `set httpproxyport` sets the port number. `set httpproxy` turns on or off the use of a proxy server, leaving the proxy host name and port intact, even when not in use.

Under the Mac and Windows operating systems, when you `set httpproxy on`, Stata will attempt to obtain the values of `httpproxyhost` and `httpproxyport` from the operating system if they have not been previously set. `set httpproxy on`, `init` attempts to obtain these values from the operating system, even if they have been previously set.

If the proxy requires authorization (user ID and password), set authorization on via `set http-proxyauth on`. The proxy user and proxy password must also be set to the appropriate user ID and password by using `set httpproxyuser` and `set httpproxypw`.

Stata remembers the various proxy settings between sessions and does not need a `permanently` option.

`set timeout1` changes the time limit in seconds that Stata imposes for establishing the initial connection with a remote host. The default value is 120. `set timeout2` changes the time limit in seconds that Stata imposes for subsequent data transfer with the host. The default value is 300. If these time limits are exceeded, a “connection timed out” message and error code 2 are produced. You should seldom need to change these settings.

## Options

`init` specifies that `set httpproxy on` attempts to initialize `httpproxyhost` and `httpproxyport` from the operating system (Mac and Windows only).

`permanently` specifies that, in addition to making the change right now, the `timeout1` and `timeout2` settings be remembered and become the default setting when you invoke Stata.

The various `httpproxy` settings do not have a `permanently` option because `permanently` is implied.

## Remarks

If you receive an error message, see <http://www.stata.com/support/faqs/web/> for the latest information.

### 1. remote connection failed r(677);

If you see

```
remote connection failed
r(677);
```

then you asked for something to be done over the web, and Stata tried but could not contact the specified host. Stata was able to talk over the network and look up the host but was not able to establish a connection to that host. Perhaps the host is down; try again later.

If all your web accesses result in this message, then perhaps your network connection is through a proxy server. If it is, then you must tell Stata.

Contact your system administrator. Ask for the name and port of the “http proxy server”. Say that you are told

```
http proxy server: jupiter.myuni.edu
port number: 8080
```

In Stata, type

```
. set httpproxyhost jupiter.myuni.edu
. set httpproxyport 8080
. set httpproxy on
```

Your web accesses should then work.

## 2. connection timed out r(2);

If you see

```
connection timed out
r(2);
```

then an Internet connection has timed out. This can happen when

- a. the connection between you and the host is slow, or
- b. the connection between you and the host has disappeared, and so it eventually “timed out”.

For (b), wait a while (say, 5 minutes) and try again (sometimes pieces of the Internet can break for up to a day, but that is rare). For (a), you can reset the limits for what constitutes “timed out”. There are two numbers to set.

The time to establish the initial connection is **timeout1**. By default, Stata waits 120 seconds before declaring a timeout. You can change the limit:

```
. set timeout1 #seconds
```

You might try doubling the usual limit and specify 240; *#seconds* must be between 1 and 32,000.

The time to retrieve data from an open connection is **timeout2**. By default, Stata waits 300 seconds (5 minutes) before declaring a timeout. To change the limit, type

```
. set timeout2 #seconds
```

You might try doubling the usual limit and specify 600; *#seconds* must be between 1 and 32,000.

## Also see

[R] [query](#) — Display system parameters

[P] [creturn](#) — Return c-class values

[U] [28 Using the Internet to keep up to date](#)

# Title

news — Report Stata news

## Syntax

news

## Menu

Help > News

## Description

news displays a brief listing of recent Stata news and information, which it obtains from Stata's web site. news requires that your computer be connected to the Internet.

You may also execute news by selecting **Help > News**.

## Remarks

news provides an easy way of displaying a brief list of the latest Stata news:

```
. news
```

```

 ___ ___ ___ ___ ___
 /__ / / ___/ / ___/
 ___/ / /___/ / ___/ News The latest from http://www.stata.com

```

---

### **8 October 2009. Official update available for download**

Click here (equivalent to pulling down **Help** and selecting **Official Updates**) or type update from <http://www.stata.com>.

### **27 July 2009. Stata 11 available**

Stata 11 -- multiple imputation -- competing-risks regression -- generalized method of moments -- multivariate GARCH -- state-space and dynamic-factor models -- predictive margins -- factor variables -- marginal analysis -- PDF documentation -- new Data Editor -- new Do-file Editor -- Variables Manager -- is now available. Click here or visit <http://www.stata.com/stata11/> for more information.

### **16 April 2009. NetCourse schedule updated**

See <http://www.stata.com/netcourse/> for more information.

(output omitted)

```
<end>
```

## Also see

[U] [28 Using the Internet to keep up to date](#)

## Syntax

*Interactive version*

```
nl (depvar = <sexp>) [if] [in] [weight] [, options]
```

*Programmed substitutable expression version*

```
nl sexp_prog : depvar [varlist] [if] [in] [weight] [, options]
```

*Function evaluator program version*

```
nl func_prog @ depvar [varlist] [if] [in] [weight] ,
 { parameters(namelist) | nparameters(#) } [options]
```

where

*depvar* is the dependent variable;

<*sexp*> is a substitutable expression;

*sexp\_prog* is a substitutable expression program; and

*func\_prog* is a function evaluator program.



| <i>options</i>                       | description                                                                                                                                                                                                      |
|--------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                |                                                                                                                                                                                                                  |
| <code>variables(varlist)</code>      | variables in model                                                                                                                                                                                               |
| <code>initial(initial_values)</code> | initial values for parameters                                                                                                                                                                                    |
| * <code>parameters(namelist)</code>  | parameters in model (function evaluator program version only)                                                                                                                                                    |
| * <code>nparameters(#)</code>        | number of parameters in model<br>(function evaluator program version only)                                                                                                                                       |
| <code>sexp_options</code>            | options for substitutable expression program                                                                                                                                                                     |
| <code>func_options</code>            | options for function evaluator program                                                                                                                                                                           |
| Model 2                              |                                                                                                                                                                                                                  |
| <code>lnlsq(#)</code>                | use log least-squares where $\ln(\text{depvar} - \#)$ is assumed to be normally distributed                                                                                                                      |
| <code>noconstant</code>              | the model has no constant term; seldom used                                                                                                                                                                      |
| <code>hasconstant(name)</code>       | use <i>name</i> as constant term; seldom used                                                                                                                                                                    |
| SE/Robust                            |                                                                                                                                                                                                                  |
| <code>vce(vctype)</code>             | <i>vctype</i> may be <code>gnr</code> , <code>robust</code> , <code>cluster clustvar</code> , <code>bootstrap</code> , <code>jackknife</code> , <code>hac kernel</code> , <code>hc2</code> , or <code>hc3</code> |
| Reporting                            |                                                                                                                                                                                                                  |
| <code>level(#)</code>                | set confidence level; default is <code>level(95)</code>                                                                                                                                                          |
| <code>leave</code>                   | create variables containing derivative of $E(y)$                                                                                                                                                                 |
| <code>title(string)</code>           | display <i>string</i> as title above the table of parameter estimates                                                                                                                                            |
| <code>title2(string)</code>          | display <i>string</i> as subtitle                                                                                                                                                                                |
| Optimization                         |                                                                                                                                                                                                                  |
| <code>optimization_options</code>    | control the optimization process; seldom used                                                                                                                                                                    |
| <code>eps(#)</code>                  | specify <i>#</i> for convergence criterion; default is <code>eps(1e-5)</code>                                                                                                                                    |
| <code>delta(#)</code>                | specify <i>#</i> for computing derivatives; default is <code>delta(4e-7)</code>                                                                                                                                  |

\* For function evaluator program version, you must specify `parameters(namelist)` or `nparameters(#)`, or both. `bootstrap`, `by`, `jackknife`, `rolling`, `statsby`, and `svy` are allowed; see [U] 11.1.10 Prefix commands.

Weights are not allowed with the `bootstrap` prefix.

`aweights` are not allowed with the `jackknife` prefix.

`vce()`, `leave`, and weights are not allowed with the `svy` prefix.

`aweights`, `fweights`, and `ifweights` are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

## Menu

Statistics > Linear models and related > Other > Nonlinear least squares

## Description

`nl` fits an arbitrary nonlinear regression function by least squares. With the interactive version of the command, you enter the function directly on the command line or dialog box by using a *substitutable expression*. If you have a function that you use regularly, you can write a *substitutable*

*expression program* and use the second syntax to avoid having to reenter the function every time. The function evaluator program version gives you the most flexibility in exchange for increased complexity; with this version, your program is given a vector of parameters and a variable list, and your program computes the regression function.

When you write a substitutable expression program or function evaluator program, the first two letters of the name must be `nl`. *sexp\_prog* and *func\_prog* refer to the name of the program without the first two letters. For example, if you wrote a function evaluator program named `nlregss`, you would type `nl regss @ ...` to estimate the parameters.

## Options

### Model

`variables(varlist)` specifies the variables in the model. `nl` ignores observations for which any of these variables have missing values. If you do not specify `variables()`, then `nl` issues an error message with return code 480 if the estimation sample contains any missing values.

`initial(initial_values)` specifies the initial values to begin the estimation. You can specify a  $1 \times k$  matrix, where  $k$  is the number of parameters in the model, or you can specify a parameter name, its initial value, another parameter name, its initial value, and so on. For example, to initialize `alpha` to 1.23 and `delta` to 4.57, you would type

```
nl ... , initial(alpha 1.23 delta 4.57) ...
```

Initial values declared using this option override any that are declared within substitutable expressions. If you specify a parameter that does not appear in your model, `nl` exits with error code 480. If you specify a matrix, the values must be in the same order that the parameters are declared in your model. `nl` ignores the row and column names of the matrix.

`parameters(namelist)` specifies the names of the parameters in the model. The names of the parameters must adhere to the naming conventions of Stata's variables; see [U] 11.3 **Naming conventions**. If you specify both `parameters()` and `nparameters()`, the number of names in the former must match the number specified in the latter; if not, `nl` issues an error message with return code 198.

`nparameters(#)` specifies the number of parameters in the model. If you do not specify names with the `parameters()` option, `nl` names them `b1`, `b2`, ..., `b#`. If you specify both `parameters()` and `nparameters()`, the number of names in the former must match the number specified in the latter; if not, `nl` issues an error message with return code 198.

*sexp\_options* refer to any options allowed by your *sexp\_prog*.

*func\_options* refer to any options allowed by your *func\_prog*.

### Model 2

`lnlsq(#)` fits the model by using log least-squares, which we define as least squares with shifted lognormal errors. In other words,  $\ln(\text{depvar} - \#)$  is assumed to be normally distributed. Sums of squares and deviance are adjusted to the same scale as *depvar*.

`noconstant` indicates that the function does not include a constant term. This option is generally not needed, even if there is no constant term in the model, unless the coefficient of variation (over observations) of the partial derivative of the function with respect to a parameter is less than `eps()` and that parameter is not a constant term.

`hasconstant(name)` indicates that parameter *name* be treated as the constant term in the model and that `nl` should not use its default algorithm to find a constant term. As with `noconstant`, this option is seldom used.

#### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

`vce(gnr)`, the default, uses the conventionally derived variance estimator for nonlinear models fit using Gauss–Newton regression.

`nl` also allows the following:

`vce(hac kernel [#])` specifies that a heteroskedasticity- and autocorrelation-consistent (HAC) variance estimate be used. HAC refers to the general form for combining weighted matrices to form the variance estimate. There are three kernels available for `nl`:

`nwest` | `gallant` | `anderson`

# specifies the number of lags. If # is not specified,  $N - 2$  is assumed.

`vce(hac kernel [#])` is not allowed if weights are specified.

`vce(hc2)` and `vce(hc3)` specify alternative bias corrections for the robust variance calculation. `vce(hc2)` and `vce(hc3)` may not be specified with the `svy` prefix. By default, `vce(robust)` uses  $\hat{\sigma}_j^2 = \{n/(n-k)\}u_j^2$  as an estimate of the variance of the  $j$ th observation, where  $u_j$  is the calculated residual and  $n/(n-k)$  is included to improve the overall estimate's small-sample properties.

`vce(hc2)` instead uses  $u_j^2/(1 - h_{jj})$  as the observation's variance estimate, where  $h_{jj}$  is the  $j$ th diagonal element of the hat (projection) matrix. This produces an unbiased estimate of the covariance matrix if the model is homoskedastic. `vce(hc2)` tends to produce slightly more conservative confidence intervals than `vce(robust)`.

`vce(hc3)` uses  $u_j^2/(1 - h_{jj})^2$  as suggested by Davidson and MacKinnon (1993 and 2004), who report that this often produces better results when the model is heteroskedastic. `vce(hc3)` produces confidence intervals that tend to be even more conservative.

See, in particular, Davidson and MacKinnon (2004, 239), who advocate the use of `vce(hc2)` or `vce(hc3)` instead of the plain robust estimator for nonlinear least squares.

#### Reporting

`level(#)`; see [R] [estimation options](#).

`leave` leaves behind after estimation a set of new variables with the same names as the estimated parameters containing the derivatives of  $E(y)$  with respect to the parameters. If the dataset contains an existing variable with the same name as a parameter, then using `leave` causes `nl` to issue an error message with return code 110.

`leave` may not be specified with `vce(cluster clustvar)` or the `svy` prefix.

`title(string)` specifies an optional title that will be displayed just above the table of parameter estimates.

`title2(string)` specifies an optional subtitle that will be displayed between the title specified in `title()` and the table of parameter estimates. If `title2()` is specified but `title()` is not, `title2()` has the same effect as `title()`.

*optimization\_options*: `iterate(#)`, `[no]log`, `trace`. `iterate()` specifies the maximum number of iterations, `log/no` specifies whether to show the iteration log, and `trace` specifies that the iteration log should include the current parameter vector. These options are seldom used.

`eps(#)` specifies the convergence criterion for successive parameter estimates and for the residual sum of squares. The default is `eps(1e-5)`.

`delta(#)` specifies the relative change in a parameter to be used in computing the numeric derivatives. The derivative for parameter  $\beta_i$  is computed as  $\{f(X, \beta_1, \beta_2, \dots, \beta_i + d, \beta_{i+1}, \dots) - f(X, \beta_1, \beta_2, \dots, \beta_i, \beta_{i+1}, \dots)\}/d$ , where  $d$  is  $\delta(\beta_i + \delta)$ . The default is `delta(4e-7)`.

## Remarks

Remarks are presented under the following headings:

- [Substitutable expressions](#)
- [Substitutable expression programs](#)
- [Built-in functions](#)
- [Lognormal errors](#)
- [Other uses](#)
- [Weights](#)
- [Potential errors](#)
- [General comments on fitting nonlinear models](#)
- [Function evaluator programs](#)

`nl` fits an arbitrary nonlinear function by least squares. The interactive version allows you to enter the function directly on the command line or dialog box using *substitutable expressions*. You can write a *substitutable expression program* for functions that you fit frequently to save yourself time. Finally, *function evaluator programs* give you the most flexibility in defining your nonlinear function, though they are more complicated to use.

The next section explains the substitutable expressions that are used to define the regression function, and the section thereafter explains how to write substitutable expression program files so that you do not need to type in commonly used functions over and over. Later sections highlight other features of `nl`.

The final section discusses function evaluator programs. If you find substitutable expressions adequate to define your nonlinear function, then you can skip that section entirely. Function evaluator programs are generally needed only for complicated problems, such as multistep estimators. The program receives a vector of parameters at which it is to compute the function and a variable into which the results are to be placed.

## Substitutable expressions

You define the nonlinear function to be fit by `nl` by using a substitutable expression. Substitutable expressions are just like any other mathematical expressions involving scalars and variables, such as those you would use with Stata's `generate` command, except that the parameters to be estimated are bound in braces. See [\[U\] 13.2 Operators](#) and [\[U\] 13.3 Functions](#) for more information on expressions.

For example, suppose that you wish to fit the function

$$y_i = \beta_0(1 - e^{-\beta_1 x_i}) + \epsilon_i$$

where  $\beta_0$  and  $\beta_1$  are the parameters to be estimated and  $\epsilon_i$  is an error term. You would simply type

```
. nl (y = {b0}*(1 - exp(-1*{b1}*x)))
```

You must enclose the entire equation in parentheses. Because `b0` and `b1` are enclosed in braces, `nl` knows that they are parameters in the model. `nl` will initialize `b0` and `b1` to zero by default. To request that `nl` initialize `b0` to 1 and `b1` to 0.25, you would type

```
. nl (y = {b0=1}*(1 - exp(-1*{b1=0.25}*x)))
```

That is, inside the braces denoting a parameter, you put the parameter name followed by an equal sign and the initial value. If a parameter appears in your function multiple times, you need only specify an initial value only once (or never, if you wish to set the initial value to zero). If you do specify more than one initial value for the same parameter, `nl` will use the *last* value given. Parameter names must follow the same conventions as variable names. See [U] [11.3 Naming conventions](#).

Frequently, even nonlinear functions contain linear combinations of variables. As an example, suppose that you wish to fit the function

$$y_i = \beta_0 \left\{ 1 - e^{-(\beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i})} \right\} + \epsilon_i$$

`nl` allows you to declare a linear combination of variables by using the shorthand notation

```
. nl (y = {b0=1}*(1 - exp(-1*{xb: x1 x2 x3})))
```

In the syntax `{xb: x1 x2 x3}`, you are telling `nl` that you are declaring a linear combination named `xb` that is a function of three variables, `x1`, `x2`, and `x3`. `nl` will create three parameters, named `xb_x1`, `xb_x2`, and `xb_x3`, and initialize them to zero. Instead of typing the previous command, you could have typed

```
. nl (y = {b0=1}*(1 - exp(-1*({xb_x1}*x1 + {xb_x2}*x2 + {xb_x3}*x3))))
```

and yielded the same result. You can refer to the parameters created by `nl` in the linear combination later in the function, though you must declare the linear combination first if you intend to do that. When creating linear combinations, `nl` ensures that the parameter names it chooses are unique and have not yet been used in the function.

In general, there are three rules to follow when defining substitutable expressions:

1. Parameters of the model are bound in braces: `{b0}`, `{param}`, etc.
2. Initial values for parameters are given by including an equal sign and the initial value inside the braces: `{b0=1}`, `{param=3.571}`, etc.
3. Linear combinations of variables can be included using the notation `{eqname:varlist}`, for example, `{xb: mpg price weight}`, `{score: w x z}`, etc. Parameters of linear combinations are initialized to zero.

(Continued on next page)

If you specify initial values by using the `initial()` option, they override whatever initial values are given within the substitutable expression. Substitutable expressions are so named because, once values are assigned to the parameters, the resulting expression can be handled by `generate` and `replace`.

► Example 1

We wish to fit the CES production function

$$\ln Q_i = \beta_0 - \frac{1}{\rho} \ln \{ \delta K_i^{-\rho} + (1 - \delta) L_i^{-\rho} \} + \epsilon_i \tag{1}$$

where  $\ln Q_i$  is the log of output for firm  $i$ ;  $K_i$  and  $L_i$  are firm  $i$ 's capital and labor usage, respectively; and  $\epsilon_i$  is a regression error term. Because  $\rho$  appears in the denominator of a fraction, zero is not a feasible initial value; for a CES production function,  $\rho = 1$  is a reasonable choice. Setting  $\delta = 0.5$  implies that labor and capital have equal impacts on output, which is also a reasonable choice for an initial value. We type

```
. use http://www.stata-press.com/data/r11/production
. nl (lnoutput = {b0} - 1/{rho=3}*ln({delta=0.5}*capital^(-1*{rho}) +
> (1 - {delta})*labor^(-1*{rho})))
(obs = 100)
Iteration 0: residual SS = 30.4558
Iteration 1: residual SS = 29.37008
Iteration 2: residual SS = 29.36595
Iteration 3: residual SS = 29.36581
Iteration 4: residual SS = 29.36581
Iteration 5: residual SS = 29.36581
Iteration 6: residual SS = 29.36581
Iteration 7: residual SS = 29.36581
```

| Source   | SS         | df | MS         |                 |          |  |
|----------|------------|----|------------|-----------------|----------|--|
| Model    | 91.1449924 | 2  | 45.5724962 | Number of obs = | 100      |  |
| Residual | 29.3658055 | 97 | .302740263 | R-squared =     | 0.7563   |  |
|          |            |    |            | Adj R-squared = | 0.7513   |  |
|          |            |    |            | Root MSE =      | .5502184 |  |
| Total    | 120.510798 | 99 | 1.21728079 | Res. dev. =     | 161.2538 |  |

| lnoutput | Coef.    | Std. Err. | t     | P> t  | [95% Conf. Interval] |
|----------|----------|-----------|-------|-------|----------------------|
| /b0      | 3.792157 | .0996818  | 38.04 | 0.000 | 3.594317 3.989998    |
| /rho     | 1.386991 | .4725791  | 2.93  | 0.004 | .4490524 2.32493     |
| /delta   | .4823616 | .0519789  | 9.28  | 0.000 | .3791978 .5855253    |

Parameter b0 taken as constant term in model & ANOVA table

`nl` will attempt to find a constant term in the model and, if one is found, mention it at the bottom of the output. `nl` found `b0` to be a constant because the partial derivative  $\partial \ln Q_i / \partial b_0$  has a coefficient of variation less than `eps()` in the estimation sample.

The elasticity of substitution for the CES production function is  $\sigma = 1/(1 + \rho)$ ; and, having fit the model, we can use `nlcom` to estimate it:

```
. nlcom (1/(1 + _b[/rho]))
 _nl_1: 1/(1 + _b[/rho])
```

| lnoutput | Coef.    | Std. Err. | t    | P> t  | [95% Conf. Interval] |
|----------|----------|-----------|------|-------|----------------------|
| _nl_1    | .4189374 | .0829417  | 5.05 | 0.000 | .2543211 .5835538    |

See [R] [nlcom](#) and [U] [13.5 Accessing coefficients and standard errors](#) for more information.

◀

`nl`'s output closely mimics that of `regress`; see [R] [regress](#) for more information. The  $R^2$ , sums of squares, and similar statistics are calculated in the same way that `regress` calculates them. If no “constant” term is specified, the usual caveats apply to the interpretation of the  $R^2$  statistic; see the comments and references in [Goldstein \(1992\)](#). Unlike `regress`, `nl` does not report a model  $F$  statistic, because a test of the joint significance of all the parameters except the constant term may not be relevant in a nonlinear model.

## Substitutable expression programs

If you fit the same model often or if you want to write an estimator that will operate on whatever variables you specify, then you will want to write a substitutable expression program. That program will return a macro containing a substitutable expression that `nl` can then evaluate, and it may optionally calculate initial values as well. The name of the program must begin with the letters `nl`.

To illustrate, suppose that you use the CES production function often in your work. Instead of typing in the formula each time, you can write a program like this:

```
program nlces, rclass
 version 11
 syntax varlist(min=3 max=3) if
 local logout : word 1 of `varlist'
 local capital : word 2 of `varlist'
 local labor : word 3 of `varlist'
 // Initial value for b0 given delta=0.5 and rho=1
 tempvar y
 generate double `y' = `logout' + ln(0.5*`capital'^(1-rho) + 0.5*`labor'^(1-rho))
 summarize `y' `if', meanonly
 local b0val = r(mean)
 // Terms for substitutable expression
 local capterm "{delta=0.5}*`capital'^(1-rho)"
 local labterm "(1-{delta})*`labor'^(1-rho)"
 local term2 "1/{rho=1}*ln(`capterm' + `labterm')"
 // Return substitutable expression and title
 return local eq "`logout' = {b0=`b0val'} - `term2'"
 return local title "CES ftn., ln Q=`logout', K=`capital', L=`labor'"
end
```

The program accepts three variables for log output, capital, and labor, and it accepts an `if exp` qualifier to restrict the estimation sample. All programs that you write to use with `nl` must accept an `if exp` qualifier because, when `nl` calls the program, it passes a binary variable that marks the estimation sample (the variable equals one if the observation is in the sample and zero otherwise). When calculating initial values, you will want to restrict your computations to the estimation sample, and you can do so by using `if` with any commands that accept `if exp` qualifiers. Even if your program does not calculate initial values or otherwise use the `if` qualifier, the `syntax` statement must still allow it. See [P] [syntax](#) for more information on the `syntax` command and the use of `if`.

As in the previous example, reasonable initial values for  $\delta$  and  $\rho$  are 0.5 and 1, respectively. Conditional on those values, (1) can be rewritten as

$$\beta_0 = \ln Q_i + \ln(0.5K_i^{-1} + 0.5L_i^{-1}) - \epsilon_i \quad (2)$$

so a good initial value for  $\beta_0$  is the mean of the right-hand side of (2) ignoring  $\epsilon_i$ . Lines 7–10 of the function evaluator program calculate that mean and store it in a local macro. Notice the use of `if` in the `summarize` statement so that the mean is calculated only for the estimation sample.

The final part of the program returns two macros. The macro `title` is optional and defines a short description of the model that will be displayed in the output immediately above the table of parameter estimates. The macro `eq` is required and defines the substitutable expression that `nl` will use. If the expression is short, you can define it all at once. However, because the expression used here is somewhat lengthy, defining local macros and then building up the final expression from them is easier.

To verify that there are no errors in your program, you can call it directly and then use `return list`:

```
. use http://www.stata-press.com/data/r11/production, clear
. nlces lnoutput capital labor
. return list

macros:
 r(title) : "CES ftn., ln Q=lnoutput, K=capital, L=labor"
 r(eq) : "lnoutput = {b0=3.711606264663641} - 1/{rho=1}*ln({delt
> a=0.5}*capital^(-1*{rho}) + (1-{delta})*labor^(-1*{rho}))"
```

The macro `r(eq)` contains the same substitutable expression that we specified at the command line in the preceding example, except for the initial value for `b0`. In short, an `nl` substitutable expression program should return in `r(eq)` the same substitutable expression you would type at the command line. The only difference is that when writing a substitutable expression program, you do not bind the entire expression inside parentheses.

Having written the program, you can use it by typing

```
. nl ces: lnoutput capital labor
```

(There is a space between `nl` and `ces`.) The output is identical to that shown in example 1, save for the title defined in the function evaluator program that appears immediately above the table of parameter estimates.

## □ Technical note

You will want to store `nlces` as an ado-file called `nlces.ado`. The alternative is to type the code into Stata interactively or to place the code in a do-file. While those alternatives are adequate for occasional use, if you save the program as an ado-file, you can use the function anytime you use Stata without having to redefine the program. When `nl` attempts to execute `nlces`, if the program is not in Stata's memory, Stata will search the disk(s) for an ado-file of the same name and, if found, automatically load it. All you have to do is name the file with the `.ado` suffix and then place it in a directory where Stata will find it. You should put the file in the directory Stata reserves for user-written ado-files, which, depending on your operating system, is `c:\ado\personal` (Windows), `~/ado/personal` (Unix), or `~:ado:personal` (Mac). See [\[U\] 17 Ado-files](#). □

Sometimes you may want to pass additional options to the substitutable expression program. You can modify the `syntax` statement of your program to accept whatever options you wish. Then when you call `nl` with the `syntax`

```
. nl func_prog: varlist, options
```

any *options* that are not recognized by `nl` (see the table of options at the beginning of this entry) are passed on to your function evaluator program. The only other restriction is that your program cannot accept an option named `at` because `nl` uses that option with function evaluator programs.



## Built-in functions

Some functions are used so often that `nl` has them built in so that you do not need to write them yourself. `nl` automatically chooses initial values for the parameters, though you can use the `initial(...)` option to override them.

Three alternatives are provided for exponential regression with one asymptote:

$$\begin{aligned}\text{exp3} & y_i = \beta_0 + \beta_1 \beta_2^{x_i} + \epsilon_i \\ \text{exp2} & y_i = \beta_1 \beta_2^{x_i} + \epsilon_i \\ \text{exp2a} & y_i = \beta_1 (1 - \beta_2^{x_i}) + \epsilon_i\end{aligned}$$

For instance, typing `nl exp3: ras dvl` fits the three-parameter exponential model (parameters  $\beta_0$ ,  $\beta_1$ , and  $\beta_2$ ) using  $y_i = \text{ras}$  and  $x_i = \text{dvl}$ .

Two alternatives are provided for the logistic function (symmetric sigmoid shape; not to be confused with logistic regression):

$$\begin{aligned}\text{log4} & y_i = \beta_0 + \beta_1 / \left[ 1 + \exp\{-\beta_2(x_i - \beta_3)\} \right] + \epsilon_i \\ \text{log3} & y_i = \beta_1 / \left[ 1 + \exp\{-\beta_2(x_i - \beta_3)\} \right] + \epsilon_i\end{aligned}$$

Finally, two alternatives are provided for the Gompertz function (asymmetric sigmoid shape):

$$\begin{aligned}\text{gom4} & y_i = \beta_0 + \beta_1 \exp\left[-\exp\{-\beta_2(x_i - \beta_3)\}\right] + \epsilon_i \\ \text{gom3} & y_i = \beta_1 \exp\left[-\exp\{-\beta_2(x_i - \beta_3)\}\right] + \epsilon_i\end{aligned}$$

## Lognormal errors

A nonlinear model with errors that are independent and identically distributed normal may be written

$$y_i = f(\mathbf{x}_i, \boldsymbol{\beta}) + u_i, \quad u_i \sim N(0, \sigma^2) \quad (3)$$

for  $i = 1, \dots, n$ . If the  $y_i$  are thought to have a  $k$ -shifted lognormal instead of a normal distribution—that is,  $\ln(y_i - k) \sim N(\zeta_i, \tau^2)$ , and the systematic part  $f(\mathbf{x}_i, \boldsymbol{\beta})$  of the original model is still thought appropriate for  $y_i$ —the model becomes

$$\ln(y_i - k) = \zeta_i + v_i = \ln\{f(\mathbf{x}_i, \boldsymbol{\beta}) - k\} + v_i, \quad v_i \sim N(0, \tau^2) \quad (4)$$

This model is fit if `lnlsq(k)` is specified.

If model (4) is correct, the variance of  $(y_i - k)$  is proportional to  $\{f(\mathbf{x}_i, \boldsymbol{\beta}) - k\}^2$ . Probably the most common case is  $k = 0$ , sometimes called “proportional errors” because the standard error of  $y_i$  is proportional to its expectation,  $f(\mathbf{x}_i, \boldsymbol{\beta})$ . Assuming that the value of  $k$  is known, (4) is just another nonlinear model in  $\boldsymbol{\beta}$ , and it may be fit as usual. However, we may wish to compare the fit of (3) with that of (4) using the residual sum of squares (RSS) or the deviance  $D$ ,  $D = -2 \times \log\text{-likelihood}$ , from each model. To do so, we must allow for the change in scale introduced by the log transformation.

Assuming, then, the  $y_i$  to be normally distributed, [Atkinson \(1985, 85–87, 184\)](#), by considering the Jacobian  $\prod |\partial \ln(y_i - k) / \partial y_i|$ , showed that multiplying both sides of (4) by the geometric mean of  $y_i - k$ ,  $\dot{y}$ , gives residuals on the same scale as those of  $y_i$ . The geometric mean is given by

$$\dot{y} = e^{n^{-1} \sum \ln(y_i - k)}$$

which is a constant for a given dataset. The residual deviance for (3) and for (4) may be expressed as

$$D(\hat{\boldsymbol{\beta}}) = \left\{ 1 + \ln(2\pi\hat{\sigma}^2) \right\} n \quad (5)$$

where  $\hat{\beta}$  is the maximum likelihood estimate (MLE) of  $\beta$  for each model and  $n\hat{\sigma}^2$  is the RSS from (3), or that from (4) multiplied by  $\hat{y}^2$ .

Because (3) and (4) are models with different error structures but the same functional form, the arithmetic difference in their RSS or deviances is not easily tested for statistical significance. However, if the deviance difference is large ( $>4$ , say), we would naturally prefer the model with the smaller deviance. Of course, the residuals for each model should be examined for departures from assumptions (nonconstant variance, nonnormality, serial correlations, etc.) in the usual way.

Alternatively, consider modeling

$$E(y_i) = 1/(C + Ae^{Bx_i}) \tag{6}$$

$$E(1/y_i) = E(y'_i) = C + Ae^{Bx_i} \tag{7}$$

where  $C$ ,  $A$ , and  $B$  are parameters to be estimated. Using the data  $(y, x) = (0.04, 5), (0.06, 12), (0.08, 25), (0.1, 35), (0.15, 42), (0.2, 48), (0.25, 60), (0.3, 75),$  and  $(0.5, 120)$  (Danuso 1991), fitting the models yields

| Model                           | $C$   | $A$   | $B$      | RSS       | Deviance |
|---------------------------------|-------|-------|----------|-----------|----------|
| (6)                             | 1.781 | 25.74 | −0.03926 | −0.001640 | −51.95   |
| (6) with <code>nlslsq(0)</code> | 1.799 | 25.45 | −0.04051 | −0.001431 | −53.18   |
| (7)                             | 1.781 | 25.74 | −0.03926 | 8.197     | 24.70    |
| (7) with <code>nlslsq(0)</code> | 1.799 | 27.45 | −0.04051 | 3.651     | 17.42    |

There is little to choose between the two versions of the logistic model (6), whereas for the exponential model (7), the fit using `nlslsq(0)` is much better (a deviance difference of 7.28). The reciprocal transformation has introduced heteroskedasticity into  $y'_i$ , which is countered by the proportional errors property of the lognormal distribution implicit in `nlslsq(0)`. The deviances are not comparable between the logistic and exponential models because the change of scale has not been allowed for, although in principle it could be.

Other uses

Even if you are fitting linear regression models, you may find that `nl` can save you some typing. Because you specify the parameters of your model explicitly, you can impose constraints on them directly.

► Example 2

In example 2 of [R] `cnsgreg`, we showed how to fit the model

$$\text{mpg} = \beta_0 + \beta_1\text{price} + \beta_2\text{weight} + \beta_3\text{displ} + \beta_4\text{gear\_ratio} + \beta_5\text{foreign} + \beta_6\text{length} + u$$

subject to the constraints

$$\begin{aligned} \beta_1 &= \beta_2 = \beta_3 = \beta_6 \\ \beta_4 &= -\beta_5 = \beta_0/20 \end{aligned}$$

An alternative way is to use `nl`:

```
. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)

. nl (mpg = {b0} + {b1}*price + {b1}*weight + {b1}*displ +
> {b0}/20*gear_ratio - {b0}/20*foreign + {b1}*length)
(obs = 74)
```

Iteration 0: residual SS = 1578.522

Iteration 1: residual SS = 1578.522

| Source   | SS         | df | MS         |                 |          |  |
|----------|------------|----|------------|-----------------|----------|--|
| Model    | 34429.4777 | 2  | 17214.7389 | Number of obs = | 74       |  |
| Residual | 1578.52226 | 72 | 21.9239203 | R-squared =     | 0.9562   |  |
|          |            |    |            | Adj R-squared = | 0.9549   |  |
|          |            |    |            | Root MSE =      | 4.682299 |  |
| Total    | 36008      | 74 | 486.594595 | Res. dev. =     | 436.4562 |  |

| mpg | Coef.    | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|-----|----------|-----------|-------|-------|----------------------|-----------|
| b0  | 26.52229 | 1.375178  | 19.29 | 0.000 | 23.78092             | 29.26365  |
| b1  | -.000923 | .0001534  | -6.02 | 0.000 | -.0012288            | -.0006172 |

The point estimates and standard errors for  $\beta_0$  and  $\beta_1$  are identical to those reported in [R] `cnsreg`. To get the estimate for  $\beta_4$ , we can use `nlcom`:

```
. nlcom _b[/b0]/20
 _nl_1: _b[b0]/20
```

| mpg   | Coef.    | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|-------|----------|-----------|-------|-------|----------------------|----------|
| _nl_1 | 1.326114 | .0687589  | 19.29 | 0.000 | 1.189046             | 1.463183 |

The advantage to using `nl` is that we do not need to use the `constraint` command six times.

◀

`nl` is also a useful tool when doing exploratory data analysis. For example, you may want to run a regression of  $y$  on a function of  $x$ , though you have not decided whether to use  $\sqrt{x}$  or  $\ln(x)$ . You can use `nl` to run both regressions without having first to generate two new variables:

```
. nl (y = {b0} + {b1}*ln(x))
. nl (y = {b0} + {b1}*sqrt(x))
```

Poi (2008) shows the advantages of using `nl` when marginal effects of transformed variables are desired as well.

## Weights

Weights are specified in the usual way—analytic and frequency weights as well as `iweights` are supported; see [U] 20.18 **Weighted estimation**. Use of analytic weights implies that the  $y_i$  have different variances. Therefore, model (3) may be rewritten as

$$y_i = f(\mathbf{x}_i, \boldsymbol{\beta}) + u_i, \quad u_i \sim N(0, \sigma^2/w_i) \quad (3a)$$

where  $w_i$  are (positive) weights, assumed to be known and normalized such that their sum equals the number of observations. The residual deviance for (3a) is

$$D(\hat{\beta}) = \{1 + \ln(2\pi\hat{\sigma}^2)\}n - \sum \ln(w_i) \quad (5a)$$

[compare with (5)], where

$$n\hat{\sigma}^2 = \text{RSS} = \sum w_i \{y_i - f(\mathbf{x}_i, \hat{\beta})\}^2$$

Defining and fitting a model equivalent to (4) when weights have been specified as in (3a) is not straightforward and has not been attempted. Thus deviances using and not using the `nlsql()` option may not be strictly comparable when analytic weights (other than 0 and 1) are used.

You do not need to modify your substitutable expression in any way to use weights. If, however, you write a substitutable expression program, then you should account for weights when obtaining initial values. When `nl` calls your program, it passes whatever weight expression (if any) was specified by the user. Here is an outline of a substitutable expression program that accepts weights:

```
program nl name, rclass
 version 11
 syntax varlist [aw fw iw] if
 ...
 // Obtain initial values allowing weights
 // Use the syntax ['weight','exp']. For example,
 summarize varname ['weight','exp'] 'if'
 regress depvar varlist ['weight','exp'] 'if'
 ...
 // Return substitutable expression
 return local eq "substitutable expression"
 return local title "description of estimator"
end
```

For details on how the `syntax` command processes weight expressions, see [P] [syntax](#).

## Potential errors

`nl` is reasonably robust to the inability of your nonlinear function to be evaluated at some parameter values. `nl` does assume that your function can be evaluated at the initial values of the parameters. If your function cannot be evaluated at the initial values, an error message is issued with return code 480. Recall that if you do not specify an initial value for a parameter, then `nl` initializes it to zero. Many nonlinear functions cannot be evaluated when some parameters are zero, so in those cases specifying alternative initial values is crucial.

Thereafter, as `nl` changes the parameter values, it monitors your function for unexpected missing values. If these are detected, `nl` backs up. That is, `nl` finds a point between the previous, known-to-be-good parameter vector and the new, known-to-be-bad vector at which the function can be evaluated and continues its iterations from that point.

`nl` requires that once a parameter vector is found where the predictions can be calculated, small changes to the parameter vector be made to calculate numeric derivatives. If a boundary is encountered at this point, an error message is issued with return code 481.

When specifying `lnlsq()`, an attempt to take logarithms of  $y_i - k$  when  $y_i \leq k$  results in an error message with return code 482.

If `iterate()` iterations are performed and estimates still have not converged, results are presented with a warning, and the return code is set to 430.

If you use the programmed substitutable expression version of `nl` with a function evaluator program, or vice versa, Stata issues an error message. Verify that you are using the syntax appropriate for the program you have.

## General comments on fitting nonlinear models

Achieving convergence is often problematic. For example, a unique minimum of the sum-of-squares function may not exist. Much literature exists on different algorithms that have been used, on strategies for obtaining good initial parameter values, and on tricks for parameterizing the model to make its behavior as linear-like as possible. Selected references are [Kennedy Jr. and Gentle \(1980, chap. 10\)](#) for computational matters and [Ross \(1990\)](#) and [Ratkowsky \(1983\)](#) for all three aspects. Ratkowsky's book is particularly clear and approachable, with useful discussion on the meaning and practical implications of intrinsic and parameter-effects nonlinearity. An excellent text on nonlinear estimation is [Gallant \(1987\)](#). Also see [Davidson and MacKinnon \(1993 and 2004\)](#).

To enhance the success of `nl`, pay attention to the form of the model fit, along the lines of [Ratkowsky and Ross](#). For example, [Ratkowsky \(1983, 49–59\)](#) analyzes three possible three-parameter yield-density models for plant growth:

$$E(y_i) = \begin{cases} (\alpha + \beta x_i)^{-1/\theta} \\ (\alpha + \beta x_i + \gamma x_i^2)^{-1} \\ (\alpha + \beta x_i^\phi)^{-1} \end{cases}$$

All three models give similar fits. However, he shows that the second formulation is dramatically more linear-like than the other two and therefore has better convergence properties. In addition, the parameter estimates are virtually unbiased and normally distributed, and the asymptotic approximation to the standard errors, correlations, and confidence intervals is much more accurate than for the other models. Even within a given model, the way the parameters are expressed (e.g.,  $\phi^{x_i}$  or  $e^{\theta x_i}$ ) affects the degree of linearity and convergence behavior.

## Function evaluator programs

Occasionally, a nonlinear function may be so complex that writing a substitutable expression for it is impractical. For example, there could be many parameters in the model. Alternatively, if you are implementing a two-step estimator, writing a substitutable expression may be altogether impossible. Function evaluator programs can be used in these situations.

`nl` will pass to your function evaluator program a list of variables, a weight expression, a variable marking the estimation sample, and a vector of parameters. Your program is to replace the dependent variable, which is the first variable in the variables list, with the values of the nonlinear function evaluated at those parameters. As with substitutable expression programs, the first two letters of the name must be `nl`.

To focus on the mechanics of the function evaluator program, again let's compare the CES production function to the previous examples. The function evaluator program is

```

program nlces2
 version 11
 syntax varlist(min=3 max=3) if, at(name)
 local logout : word 1 of 'varlist'
 local capital : word 2 of 'varlist'
 local labor : word 3 of 'varlist'
 // Retrieve parameters out of at matrix
 tempname b0 rho delta
 scalar 'b0' = 'at'[1, 1]
 scalar 'rho' = 'at'[1, 2]
 scalar 'delta' = 'at'[1, 3]
 tempvar kterm lterm
 generate double 'kterm' = 'delta'*'capital'^(-1*'rho') 'if'
 generate double 'lterm' = (1-'delta')*'labor'^(-1*'rho') 'if'
 // Fill in dependent variable
 replace 'logout' = 'b0' - 1/'rho'*ln('kterm' + 'lterm') 'if'
end

```

Unlike the previous `nlces` program, this one is not declared to be `r-class`. The `syntax` statement again accepts three variables: one for log output, one for capital, and one for labor. An *if exp* is again required because `nl` will pass a binary variable marking the estimation sample. All function evaluator programs must accept an option named `at()` that takes a `name` as an argument—that is how `nl` passes the parameter vector to your program.

The next part of the program retrieves the output, labor, and capital variables from the variables list. It then breaks up the temporary matrix `at` and retrieves the parameters `b0`, `rho`, and `delta`. Pay careful attention to the order in which the parameters refer to the columns of the `at` matrix because that will affect the syntax you use with `nl`. The temporary names you use inside this program are immaterial, however.

The rest of the program computes the nonlinear function, using some temporary variables to hold intermediate results. The final line of the program then replaces the dependent variable with the values of the function. Notice the use of `'if'` to restrict attention to the estimation sample. `nl` makes a copy of your dependent variable so that when the command is finished your data are left unchanged.

To use the program and fit your model, you type

```

. use http://www.stata-press.com/data/r11/production, clear
. nl ces2 @ lnoutput capital labor, parameters(b0 rho delta)
> initial(b0 0 rho 1 delta 0.5)

```

The output is again identical to that shown in [example 1](#). The order in which the parameters were specified in the `parameters()` option is the same in which they are retrieved from the `at` matrix in the program. To initialize them, you simply list the parameter name, a space, the initial value, and so on.

If you use the `nparameters()` option instead of the `parameters()` option, the parameters are named `b1`, `b2`, ..., `bk`, where  $k$  is the number of parameters. Thus you could have typed

```

. nl ces2 @ lnoutput capital labor, nparameters(3) initial(b1 0 b2 1 b3 0.5)

```

With that syntax, the parameters called `b0`, `rho`, and `delta` in the program will be labeled `b1`, `b2`, and `b3`, respectively. In programming situations or if there are many parameters, instead of listing the parameter names and initial values in the `initial()` option, you may find it more convenient to pass a column vector. In those cases, you could type

```

. matrix myvals = (0, 1, 0.5)
. nl ces2 @ lnoutput capital labor, nparameters(3) initial(myvals)

```

In summary, a function evaluator program receives a list of variables, the first of which is the dependent variable that you are to replace with the values of your nonlinear function. Additionally, it must accept an *if exp*, as well as an option named **at** that will contain the vector of parameters at which **nl** wants the function evaluated. You are then free to do whatever is necessary to evaluate your function and replace the dependent variable.

If you wish to use weights, your function evaluator program's **syntax** statement must accept them. If your program consists only of, for example, **generate** statements, you need not do anything with the weights passed to your program. However, if in calculating the nonlinear function you use commands such as **summarize** or **regress**, then you will want to use the weights with those commands.

As with substitutable expression programs, **nl** will pass to it any options specified that **nl** does not accept, providing you with a way to pass more information to your function.

## □ Technical note

Before version 9 of Stata, the **nl** command used a different syntax, which required you to write an *nlfcn* program, and it did not have a syntax for interactive use other than the seven functions that were built-in. The old syntax of **nl** still works, and you can still use those *nlfcn* programs. If **nl** does not see a colon, an at sign, or a set of parentheses surrounding the equation in your command, it assumes that the old syntax is being used.

The current version of **nl** uses scalars and matrices to store intermediate calculations instead of local and global macros as the old version did, so the current version produces more accurate results. In practice, however, any discrepancies are likely to be small.

□

(Continued on next page)

## Saved results

nl saves the following in `e()`:

### Scalars

|                            |                                                                         |
|----------------------------|-------------------------------------------------------------------------|
| <code>e(N)</code>          | number of observations                                                  |
| <code>e(k)</code>          | number of parameters                                                    |
| <code>e(k_eq_model)</code> | number of equations to use for model test; always 0                     |
| <code>e(k_aux)</code>      | number of estimated parameters                                          |
| <code>e(df_m)</code>       | model degrees of freedom                                                |
| <code>e(df_r)</code>       | residual degrees of freedom                                             |
| <code>e(df_t)</code>       | total degrees of freedom                                                |
| <code>e(mss)</code>        | model sum of squares                                                    |
| <code>e(rss)</code>        | residual sum of squares                                                 |
| <code>e(tss)</code>        | total sum of squares                                                    |
| <code>e(mms)</code>        | model mean square                                                       |
| <code>e(msr)</code>        | residual mean square                                                    |
| <code>e(ll)</code>         | log likelihood assuming i.i.d. normal errors                            |
| <code>e(r2)</code>         | <i>R</i> -squared                                                       |
| <code>e(r2_a)</code>       | adjusted <i>R</i> -squared                                              |
| <code>e(rmse)</code>       | root mean squared error                                                 |
| <code>e(dev)</code>        | residual deviance                                                       |
| <code>e(N_clust)</code>    | number of clusters                                                      |
| <code>e(lnlsq)</code>      | value of <code>lnlsq</code> if specified                                |
| <code>e(log_t)</code>      | 1 if <code>lnlsq</code> specified, 0 otherwise                          |
| <code>e(gm_2)</code>       | square of geometric mean of $(y-k)$ if <code>lnlsq</code> ; 1 otherwise |
| <code>e(cj)</code>         | position of constant in <code>e(b)</code> or 0 if no constant           |
| <code>e(delta)</code>      | relative change used to compute derivatives                             |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                                               |
| <code>e(ic)</code>         | number of iterations                                                    |
| <code>e(converge)</code>   | 1 if converged, 0 otherwise                                             |



## Macros

|                              |                                                                                                                |
|------------------------------|----------------------------------------------------------------------------------------------------------------|
| <code>e(cmd)</code>          | <code>nl</code>                                                                                                |
| <code>e(cmdline)</code>      | command as typed                                                                                               |
| <code>e(depvar)</code>       | name of dependent variable                                                                                     |
| <code>e(wtype)</code>        | weight type                                                                                                    |
| <code>e(wexp)</code>         | weight expression                                                                                              |
| <code>e(title)</code>        | title in estimation output                                                                                     |
| <code>e(title_2)</code>      | secondary title in estimation output                                                                           |
| <code>e(clustvar)</code>     | name of cluster variable                                                                                       |
| <code>e(hac_kernel)</code>   | HAC kernel                                                                                                     |
| <code>e(hac_lag)</code>      | HAC lag                                                                                                        |
| <code>e(vce)</code>          | <i>vcetype</i> specified in <code>vce()</code>                                                                 |
| <code>e(vcetype)</code>      | title used to label Std. Err.                                                                                  |
| <code>e(type)</code>         | 1 = interactively entered expression<br>2 = substitutable expression program<br>3 = function evaluator program |
| <code>e(sexp)</code>         | substitutable expression                                                                                       |
| <code>e(params)</code>       | names of parameters                                                                                            |
| <code>e(funcprog)</code>     | function evaluator program                                                                                     |
| <code>e(rhs)</code>          | contents of <code>variables()</code>                                                                           |
| <code>e(properties)</code>   | <code>b V</code>                                                                                               |
| <code>e(predict)</code>      | program used to implement <code>predict</code>                                                                 |
| <code>e(marginsnotok)</code> | predictions disallowed by <code>margins</code>                                                                 |
| <code>e(marginsprop)</code>  | signals to the <code>margins</code> command                                                                    |

## Matrices

|                      |                                              |
|----------------------|----------------------------------------------|
| <code>e(b)</code>    | coefficient vector                           |
| <code>e(init)</code> | initial values vector                        |
| <code>e(V)</code>    | variance–covariance matrix of the estimators |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`nl` is implemented as an ado-file.

The derivation here is based on [Davidson and MacKinnon \(2004, chap. 6\)](#). Let  $\beta$  denote the  $k \times 1$  vector of parameters, and write the regression function using matrix notation as  $\mathbf{y} = \mathbf{f}(\mathbf{x}, \beta) + \mathbf{u}$  so that the objective function can be written as

$$\text{SSR}(\beta) = \{\mathbf{y} - \mathbf{f}(\mathbf{x}, \beta)\}' \mathbf{D} \{\mathbf{y} - \mathbf{f}(\mathbf{x}, \beta)\}$$

The  $\mathbf{D}$  matrix contains the weights and is defined in [\[R\] regress](#); if no weights are specified, then  $\mathbf{D}$  is the  $N \times N$  identity matrix. Taking a second-order Taylor series expansion centered at  $\beta_0$  yields

$$\text{SSR}(\beta) \approx \text{SSR}(\beta_0) + \mathbf{g}'(\beta_0)(\beta - \beta_0) + \frac{1}{2}(\beta - \beta_0)' \mathbf{H}(\beta_0)(\beta - \beta_0) \quad (8)$$

where  $\mathbf{g}(\beta_0)$  denotes the  $k \times 1$  gradient of  $\text{SSR}(\beta)$  evaluated at  $\beta_0$  and  $\mathbf{H}(\beta_0)$  denotes the  $k \times k$  Hessian of  $\text{SSR}(\beta)$  evaluated at  $\beta_0$ . Letting  $\mathbf{X}$  denote the  $N \times k$  matrix of derivatives of  $\mathbf{f}(\mathbf{x}, \beta)$  with respect to  $\beta$ , the gradient  $\mathbf{g}(\beta)$  is

$$\mathbf{g}(\beta) = -2\mathbf{X}'\mathbf{D}\mathbf{u} \quad (9)$$

$\mathbf{X}$  and  $\mathbf{u}$  are obviously functions of  $\beta$ , though for notational simplicity that dependence is not shown explicitly. The  $(m, n)$  element of the Hessian can be written

$$H_{mn}(\beta) = -2 \sum_{i=1}^{i=N} d_{ii} \left[ \frac{\partial^2 f_i}{\partial \beta_m \partial \beta_n} u_i - X_{im} X_{in} \right] \quad (10)$$

where  $d_{ii}$  is the  $i$ th diagonal element of  $\mathbf{D}$ . As discussed in [Davidson and MacKinnon \(2004, chap. 6\)](#), the first term inside the brackets of (10) has expectation zero, so the Hessian can be approximated as

$$\mathbf{H}(\beta) = 2\mathbf{X}'\mathbf{D}\mathbf{X} \quad (11)$$

Differentiating the Taylor series expansion of  $\text{SSR}(\beta)$  shown in (8) yields the first-order condition for a minimum

$$\mathbf{g}(\beta_0) + \mathbf{H}(\beta_0)(\beta - \beta_0) = \mathbf{0}$$

which suggests the iterative procedure

$$\beta_{j+1} = \beta_j - \alpha \mathbf{H}^{-1}(\beta_j) \mathbf{g}(\beta_j) \quad (12)$$

where  $\alpha$  is a “step size” parameter chosen at each iteration to improve convergence. Using (9) and (11), we can write (12) as

$$\beta_{j+1} = \beta_j + \alpha (\mathbf{X}'\mathbf{D}\mathbf{X})^{-1} \mathbf{X}'\mathbf{D}\mathbf{u} \quad (13)$$

where  $\mathbf{X}$  and  $\mathbf{u}$  are evaluated at  $\beta_j$ . Apart from the scalar  $\alpha$ , the second term on the right-hand side of (13) can be computed via a (weighted) regression of the columns of  $\mathbf{X}$  on the errors. `n1` computes the derivatives numerically and then calls `regress`. At each iteration,  $\alpha$  is set to one, and a candidate value  $\beta_{j+1}^*$  is computed by (13). If  $\text{SSR}(\beta_{j+1}^*) < \text{SSR}(\beta_j)$ , then  $\beta_{j+1} = \beta_{j+1}^*$  and the iteration is complete. Otherwise,  $\alpha$  is halved, a new  $\beta_{j+1}^*$  is calculated, and the process is repeated. Convergence is declared when  $\alpha|\beta_{j+1,m}| \leq \epsilon(|\beta_{jm}| + \tau)$  for all  $m = 1, \dots, k$ . `n1` uses  $\tau = 10^{-3}$  and, by default,  $\epsilon = 10^{-5}$ , though you can specify an alternative value of  $\epsilon$  with the `eps()` option.

As derived, for example, in [Davidson and MacKinnon \(2004, chap. 6\)](#), an expedient way to obtain the covariance matrix is to compute  $\mathbf{u}$  and the columns of  $\mathbf{X}$  at the final estimate  $\hat{\beta}$  and then regress that  $\mathbf{u}$  on  $\mathbf{X}$ . The covariance matrix of the estimated parameters of that regression serves as an estimate of  $\text{Var}(\hat{\beta})$ . If that regression employs a robust covariance matrix estimator, then the covariance matrix for the parameters of the nonlinear regression will also be robust.

All other statistics are calculated analogously to those in linear regression, except that the nonlinear function  $f(\mathbf{x}_i, \beta)$  plays the role of the linear function  $\mathbf{x}_i'\beta$ . See [\[R\] regress](#).

This command supports estimation with survey data. For details on VCEs with survey data, see [\[SVY\] variance estimation](#).

## Acknowledgments

The original version of `n1` was written by Patrick Royston of the MRC Clinical Trials Unit, London, and published in [Royston \(1992\)](#). Francesco Danuso’s menu-driven nonlinear regression program (1991) provided the inspiration.

## References

- Atkinson, A. C. 1985. *Plots, Transformations, and Regression: An Introduction to Graphical Methods of Diagnostic Regression Analysis*. Oxford: Oxford University Press.
- Danuso, F. 1991. [sg1: Nonlinear regression command](#). *Stata Technical Bulletin* 1: 17–19. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 96–98. College Station, TX: Stata Press.
- Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.
- . 2004. *Econometric Theory and Methods*. New York: Oxford University Press.
- Gallant, A. R. 1987. *Nonlinear Statistical Models*. New York: Wiley.
- Goldstein, R. 1992. [srd7: Adjusted summary statistics for logarithmic regressions](#). *Stata Technical Bulletin* 5: 17–21. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 178–183. College Station, TX: Stata Press.
- Kennedy Jr., W. J., and J. E. Gentle. 1980. *Statistical Computing*. New York: Dekker.
- Poi, B. P. 2008. [Stata tip 58: nl is not just for nonlinear models](#). *Stata Journal* 8: 139–141.
- Ratkowsky, D. A. 1983. *Nonlinear Regression Modeling: A Unified Practical Approach*. New York: Dekker.
- Ross, G. J. S. 1987. *MLP User Manual, release 3.08*. Oxford: Numerical Algorithms Group.
- . 1990. *Nonlinear Estimation*. New York: Springer.
- Royston, P. 1992. [sg7: Centile estimation command](#). *Stata Technical Bulletin* 8: 12–15. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 122–125. College Station, TX: Stata Press.
- . 1993. [sg1.4: Standard nonlinear curve fits](#). *Stata Technical Bulletin* 11: 17. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, p. 121. College Station, TX: Stata Press.

## Also see

- [R] [nl postestimation](#) — Postestimation tools for nl
- [R] [gmm](#) — Generalized method of moments estimation
- [R] [ml](#) — Maximum likelihood estimation
- [R] [nlcom](#) — Nonlinear combinations of estimators
- [R] [nlsur](#) — Estimation of nonlinear systems of equations
- [R] [regress](#) — Linear regression
- [SVY] [svy estimation](#) — Estimation commands for survey data
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `nl`:

| command                           | description                                                                                         |
|-----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>                | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>          | postestimation statistics for survey data                                                           |
| <code>estimates</code>            | cataloging estimation results                                                                       |
| <code>lincom</code>               | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code> <sup>1</sup>  | likelihood-ratio test                                                                               |
| <code>margins</code> <sup>2</sup> | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>                | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>              | predictions and residuals                                                                           |
| <code>predictnl</code>            | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>                 | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>               | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

<sup>2</sup> You must specify the `variables()` option with `nl`.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic]

predict [type] { stub*|newvar1 ... newvark } [if] [in] , scores
```

where *k* is the number of parameters in the model.

| statistic               | description                                 |
|-------------------------|---------------------------------------------|
| Main                    |                                             |
| <code>_bhat</code>      | fitted values; the default                  |
| <code>_residuals</code> | residuals                                   |
| <code>pr(a,b)</code>    | $\Pr(y_j \mid a < y_j < b)$                 |
| <code>e(a,b)</code>     | $E(y_j \mid a < y_j < b)$                   |
| <code>ystar(a,b)</code> | $E(y_j^*), y_j^* = \max\{a, \min(y_j, b)\}$ |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

**yhat**, the default, calculates the fitted values.

**residuals** calculates the residuals.

**pr(*a*,*b*)** calculates  $\Pr(a < \mathbf{x}_j \mathbf{b} + u_j < b)$ , the probability that  $y_j | \mathbf{x}_j$  would be observed in the interval  $(a, b)$ , assuming that the error term in the model is independent and identically distributed.

*a* and *b* may be specified as numbers or variable names; *lb* and *ub* are variable names;

**pr(20,30)** calculates  $\Pr(20 < \mathbf{x}_j \mathbf{b} + u_j < 30)$ ;

**pr(*lb*,*ub*)** calculates  $\Pr(lb < \mathbf{x}_j \mathbf{b} + u_j < ub)$ ; and

**pr(20,*ub*)** calculates  $\Pr(20 < \mathbf{x}_j \mathbf{b} + u_j < ub)$ .

*a* missing (*a* ≥ .) means  $-\infty$ ; **pr(. ,30)** calculates  $\Pr(-\infty < \mathbf{x}_j \mathbf{b} + u_j < 30)$ ;

**pr(*lb*,30)** calculates  $\Pr(-\infty < \mathbf{x}_j \mathbf{b} + u_j < 30)$  in observations for which *lb* ≥ .

and calculates  $\Pr(lb < \mathbf{x}_j \mathbf{b} + u_j < 30)$  elsewhere.

*b* missing (*b* ≥ .) means  $+\infty$ ; **pr(20,.)** calculates  $\Pr(+\infty > \mathbf{x}_j \mathbf{b} + u_j > 20)$ ;

**pr(20,*ub*)** calculates  $\Pr(+\infty > \mathbf{x}_j \mathbf{b} + u_j > 20)$  in observations for which *ub* ≥ .

and calculates  $\Pr(20 < \mathbf{x}_j \mathbf{b} + u_j < ub)$  elsewhere.

**e(*a*,*b*)** calculates  $E(\mathbf{x}_j \mathbf{b} + u_j \mid a < \mathbf{x}_j \mathbf{b} + u_j < b)$ , the expected value of  $y_j | \mathbf{x}_j$  conditional on  $y_j | \mathbf{x}_j$  being in the interval  $(a, b)$ , meaning that  $y_j | \mathbf{x}_j$  is censored.

*a* and *b* are specified as they are for **pr()**.

**ystar(*a*,*b*)** calculates  $E(y_j^*)$ , where  $y_j^* = a$  if  $\mathbf{x}_j \mathbf{b} + u_j \leq a$ ,  $y_j^* = b$  if  $\mathbf{x}_j \mathbf{b} + u_j \geq b$ , and  $y_j^* = \mathbf{x}_j \mathbf{b} + u_j$  otherwise, meaning that  $y_j^*$  is truncated. *a* and *b* are specified as they are for **pr()**.

**scores** calculates the scores. The *j*th new variable created will contain the score for the *j*th parameter in **e(b)**.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] **nl** — Nonlinear least-squares estimation

[U] **20 Estimation and postestimation commands**

Syntax

Nonlinear combination of estimators—one expression

```
nlcom [name:]exp [, options]
```

Nonlinear combinations of estimators—more than one expression

```
nlcom ([name:]exp) ([([name:]exp) ...] [, options]
```

The second syntax means that if more than one expression is specified, each must be surrounded by parentheses. *exp* is any function of the parameter estimates that is valid syntax for `testnl`; see [R] [testnl](#). However, *exp* may not contain an equal sign or a comma. The optional *name* is any valid Stata name and labels the transformations.

| options                            | description                                             |
|------------------------------------|---------------------------------------------------------|
| <code>level(#)</code>              | set confidence level; default is <code>level(95)</code> |
| <code>iterate(#)</code>            | maximum number of iterations                            |
| <code>post</code>                  | post estimation results                                 |
| <sup>†</sup> <code>noheader</code> | suppress output header                                  |

<sup>†</sup>`noheader` does not appear in the dialog box.

Menu

Statistics > Postestimation > Nonlinear combinations of estimates

Description

`nlcom` computes point estimates, standard errors, test statistics, significance levels, and confidence intervals for (possibly) nonlinear combinations of parameter estimates after any Stata estimation command. Results are displayed in the usual table format used for displaying estimation results. Calculations are based on the “delta method”, an approximation appropriate in large samples.

`nlcom` can be used with `svy` estimation results; see [SVY] [svy postestimation](#).

Options

- `level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see [U] [20.7 Specifying the width of confidence intervals](#).
- `iterate(#)` specifies the maximum number of iterations used to find the optimal step size in calculating numerical derivatives of the transformation(s) with respect to the original parameters. By default, the maximum number of iterations is 100, but convergence is usually achieved after only a few iterations. You should rarely have to use this option.

`post` causes `nlcom` to behave like a Stata estimation (`eclass`) command. When `post` is specified, `nlcom` will post the vector of transformed estimators and its estimated variance–covariance matrix to `e()`. This option, in essence, makes the transformation permanent. Thus you could, after posting, treat the transformed estimation results in the same way as you would treat results from other Stata estimation commands. For example, after posting, you could redisplay the results by typing `nlcom` without any arguments, or use `test` to perform simultaneous tests of hypotheses on linear combinations of the transformed estimators.

Specifying `post` clears out the previous estimation results, which can be recovered only by refitting the original model or by storing the estimation results before running `nlcom` and then restoring them; see [R] [estimates store](#).

The following option is available with `nlcom` but is not shown in the dialog box:

`noheader` suppresses the output header.

## Remarks

Remarks are presented under the following headings:

[Introduction](#)

[Basics](#)

[Using the post option](#)

[Reparameterizing ML estimators for univariate data](#)

[nlcom versus eform](#)

## Introduction

`nlcom` and `predictnl` are Stata's delta method commands—they take nonlinear transformations of the estimated parameter vector from some fitted model and apply the delta method to calculate the variance, standard error, Wald test statistic, etc., of the transformations. `nlcom` is designed for functions of the parameters, and `predictnl` is designed for functions of the parameters and of the data, that is, for predictions.

`nlcom` generalizes `lincom` (see [R] [lincom](#)) in two ways. First, `nlcom` allows the transformations to be nonlinear. Second, `nlcom` can be used to simultaneously estimate many transformations (whether linear or nonlinear) and to obtain the estimated variance–covariance matrix of these transformations.

## Basics

In [R] [lincom](#), the following regression was performed:

(Continued on next page)

```
. use http://www.stata-press.com/data/r11/regress
. regress y x1 x2 x3
```

| Source   | SS         | df  | MS         |                 |        |  |
|----------|------------|-----|------------|-----------------|--------|--|
| Model    | 3259.3561  | 3   | 1086.45203 | Number of obs = | 148    |  |
| Residual | 1627.56282 | 144 | 11.3025196 | F( 3, 144) =    | 96.12  |  |
|          |            |     |            | Prob > F =      | 0.0000 |  |
|          |            |     |            | R-squared =     | 0.6670 |  |
|          |            |     |            | Adj R-squared = | 0.6600 |  |
|          |            |     |            | Root MSE =      | 3.3619 |  |
| Total    | 4886.91892 | 147 | 33.2443464 |                 |        |  |

| y     | Coef.    | Std. Err. | t      | P> t  | [95% Conf. Interval] |           |
|-------|----------|-----------|--------|-------|----------------------|-----------|
| x1    | 1.457113 | 1.07461   | 1.36   | 0.177 | -.666934             | 3.581161  |
| x2    | 2.221682 | .8610358  | 2.58   | 0.011 | .5197797             | 3.923583  |
| x3    | -.006139 | .0005543  | -11.08 | 0.000 | -.0072345            | -.0050435 |
| _cons | 36.10135 | 4.382693  | 8.24   | 0.000 | 27.43863             | 44.76407  |

Then `lincom` was used to estimate the difference between the coefficients of `x1` and `x2`:

```
. lincom _b[x2] - _b[x1]
(1) - x1 + x2 = 0
```

| y   | Coef.    | Std. Err. | t    | P> t  | [95% Conf. Interval] |          |
|-----|----------|-----------|------|-------|----------------------|----------|
| (1) | .7645682 | .9950282  | 0.77 | 0.444 | -1.20218             | 2.731316 |

It was noted, however, that nonlinear expressions are not allowed with `lincom`:

```
. lincom _b[x2]/_b[x1]
not possible with test
r(131);
```

Nonlinear transformations are instead estimated using `nlcom`:

```
. nlcom _b[x2]/_b[x1]
 _nl_1: _b[x2]/_b[x1]
```

| y     | Coef.    | Std. Err. | t    | P> t  | [95% Conf. Interval] |          |
|-------|----------|-----------|------|-------|----------------------|----------|
| _nl_1 | 1.524714 | .9812848  | 1.55 | 0.122 | -.4148688            | 3.464297 |

❏ Technical note

The notation `_b[name]` is the standard way in Stata to refer to regression coefficients; see [\[U\] 13.5 Accessing coefficients and standard errors](#). Some commands, such as `lincom` and `test`, allow you to drop the `_b[` and just refer to the coefficients by *name*. `nlcom`, however, requires the full specification `_b[name]`.



Returning to our linear regression example, `nlcom` also allows simultaneous estimation of more than one combination:



```
. nlcom (_b[x2]/_b[x1]) (_b[x3]/_b[x1]) (_b[x3]/_b[x2])
 _nl_1: _b[x2]/_b[x1]
 _nl_2: _b[x3]/_b[x1]
 _nl_3: _b[x3]/_b[x2]
```

| y     | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|-------|-----------|-----------|-------|-------|----------------------|-----------|
| _nl_1 | 1.524714  | .9812848  | 1.55  | 0.122 | -.4148688            | 3.464297  |
| _nl_2 | -.0042131 | .0033483  | -1.26 | 0.210 | -.0108313            | .002405   |
| _nl_3 | -.0027632 | .0010695  | -2.58 | 0.011 | -.0048772            | -.0006493 |

We can also label the transformations to produce more informative names in the estimation table:

```
. nlcom (ratio21:_b[x2]/_b[x1]) (ratio31:_b[x3]/_b[x1]) (ratio32:_b[x3]/_b[x2])
 ratio21: _b[x2]/_b[x1]
 ratio31: _b[x3]/_b[x1]
 ratio32: _b[x3]/_b[x2]
```

| y       | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|---------|-----------|-----------|-------|-------|----------------------|-----------|
| ratio21 | 1.524714  | .9812848  | 1.55  | 0.122 | -.4148688            | 3.464297  |
| ratio31 | -.0042131 | .0033483  | -1.26 | 0.210 | -.0108313            | .002405   |
| ratio32 | -.0027632 | .0010695  | -2.58 | 0.011 | -.0048772            | -.0006493 |

nlcom saves the vector of estimated combinations and its estimated variance–covariance matrix in `r()`.

```
. matrix list r(b)
r(b)[1,3]
 ratio21 ratio31 ratio32
c1 1.5247143 -.00421315 -.00276324
. matrix list r(V)
symmetric r(V)[3,3]
 ratio21 ratio31 ratio32
ratio21 .96291982
ratio31 -.00287781 .00001121
ratio32 -.00014234 2.137e-06 1.144e-06
```

## Using the post option

When used with the `post` option, nlcom saves the estimation vector and variance–covariance matrix in `e()`, making the transformation permanent:

```
. quietly nlcom (ratio21:_b[x2]/_b[x1]) (ratio31:_b[x3]/_b[x1])
> (ratio32:_b[x3]/_b[x2]), post
. matrix list e(b)
e(b)[1,3]
 ratio21 ratio31 ratio32
y1 1.5247143 -.00421315 -.00276324
. matrix list e(V)
symmetric e(V)[3,3]
 ratio21 ratio31 ratio32
ratio21 .96291982
ratio31 -.00287781 .00001121
ratio32 -.00014234 2.137e-06 1.144e-06
```

After posting, we can proceed as if we had just run a Stata estimation (`eclass`) command. For instance, we can replay the results,

```
. nlcom
```

| y       | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|---------|-----------|-----------|-------|-------|----------------------|-----------|
| ratio21 | 1.524714  | .9812848  | 1.55  | 0.122 | -.4148688            | 3.464297  |
| ratio31 | -.0042131 | .0033483  | -1.26 | 0.210 | -.0108313            | .002405   |
| ratio32 | -.0027632 | .0010695  | -2.58 | 0.011 | -.0048772            | -.0006493 |

or perform other postestimation tasks in the transformed metric, this time making reference to the new “coefficients”:

```
. display _b[ratio31]
-.00421315
. estat vce, correlation
Correlation matrix of coefficients of nlcom model
```

| e(V)    | ratio21 | ratio31 | ratio32 |
|---------|---------|---------|---------|
| ratio21 | 1.0000  |         |         |
| ratio31 | -0.8759 | 1.0000  |         |
| ratio32 | -0.1356 | 0.5969  | 1.0000  |

```
. test _b[ratio21] = 1
(1) ratio21 = 1
 F(1, 144) = 0.29
 Prob > F = 0.5937
```

We see that testing `_b[ratio21]=1` in the transformed metric is equivalent to testing using `testnl _b[x2]/_b[x1]=1` in the original metric:

```
. quietly reg y x1 x2 x3
. testnl _b[x2]/_b[x1] = 1
(1) _b[x2]/_b[x1] = 1
 F(1, 144) = 0.29
 Prob > F = 0.5937
```

We needed to refit the regression model to recover the original parameter estimates.

❑ Technical note

In a previous technical note, we mentioned that commands such as `lincom` and `test` permit reference to *name* instead of `_b[name]`. This is not the case when `lincom` and `test` are used after `nlcom`, `post`. In the above, we used

```
. test _b[ratio21] = 1
```

rather than

```
. test ratio21 = 1
```

which would have returned an error. Consider this a limitation of Stata. For the shorthand notation to work, you need a variable named *name* in the data. In `nlcom`, however, *name* is just a coefficient label that does not necessarily correspond to any variable in the data.



## Reparameterizing ML estimators for univariate data

When run using only a response and no covariates, Stata's maximum likelihood (ML) estimation commands will produce ML estimates of the parameters of some assumed univariate distribution for the response. The parameterization, however, is usually not one we are used to dealing with in a nonregression setting. In such cases, `nlcom` can be used to transform the estimation results from a regression model to those from a maximum likelihood estimation of the parameters of a univariate probability distribution in a more familiar metric.

### ► Example 1

Consider the following univariate data on  $Y = \#$  of traffic accidents at a certain intersection in a given year:

```
. use http://www.stata-press.com/data/r11/trafint
. summarize accidents
```

| Variable  | Obs | Mean     | Std. Dev. | Min | Max |
|-----------|-----|----------|-----------|-----|-----|
| accidents | 12  | 13.83333 | 14.47778  | 0   | 41  |

A quick glance of the output from `summarize` leads us to quickly reject the assumption that  $Y$  is distributed as Poisson because the estimated variance of  $Y$  is much greater than the estimated mean of  $Y$ .

Instead, we choose to model the data as univariate negative binomial, of which a common parameterization is

$$\Pr(Y = y) = \frac{\Gamma(r + y)}{\Gamma(r)\Gamma(y + 1)} p^r (1 - p)^y \quad 0 \leq p \leq 1, \quad r > 0, \quad y = 0, 1, \dots$$

with

$$E(Y) = \frac{r(1 - p)}{p} \quad \text{Var}(Y) = \frac{r(1 - p)}{p^2}$$

There exist no closed-form solutions for the maximum likelihood estimates of  $p$  and  $r$ , yet they may be estimated by the iterative method of Newton–Raphson. One way to get these estimates would be to write our own Newton–Raphson program for the negative binomial. Another way would be to write our own ML evaluator; see [R] [ml](#).

The easiest solution, however, would be to use Stata's existing negative binomial ML regression command, `nbreg`. The only problem with this solution is that `nbreg` estimates a different parameterization of the negative binomial, but we can worry about that later.

```
. nbreg accidents
Fitting Poisson model:
Iteration 0: log likelihood = -105.05361
Iteration 1: log likelihood = -105.05361
Fitting constant-only model:
Iteration 0: log likelihood = -43.948619
Iteration 1: log likelihood = -43.891483
Iteration 2: log likelihood = -43.89144
Iteration 3: log likelihood = -43.89144
Fitting full model:
Iteration 0: log likelihood = -43.89144
Iteration 1: log likelihood = -43.89144
```

Negative binomial regression

Dispersion = mean

Log likelihood = -43.89144

Number of obs = 12

LR chi2(0) = 0.00

Prob > chi2 = .

Pseudo R2 = 0.0000

| accidents | Coef.    | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|-----------|----------|-----------|------|-------|----------------------|----------|
| _cons     | 2.627081 | .3192233  | 8.23 | 0.000 | 2.001415             | 3.252747 |
| /lnalpha  | .1402425 | .4187147  |      |       | -.6804233            | .9609083 |
| alpha     | 1.150553 | .4817534  |      |       | .5064026             | 2.61407  |

Likelihood-ratio test of alpha=0:   chibar2(01) = 122.32 Prob>=chibar2 = 0.000

. nbreg, coeflegend

Negative binomial regression

Dispersion = mean

Log likelihood = -43.89144

Number of obs = 12

LR chi2(0) = 0.00

Prob > chi2 = .

Pseudo R2 = 0.0000

| accidents | Coef.    | Legend              |
|-----------|----------|---------------------|
| _cons     | 2.627081 | _b[accidents:_cons] |
| /lnalpha  | .1402425 | _b[lnalpha:_cons]   |
| alpha     | 1.150553 |                     |

Likelihood-ratio test of alpha=0:   chibar2(01) = 122.32 Prob>=chibar2 = 0.000

From this output, we see that, when used with univariate data, `nbreg` estimates a regression intercept,  $\beta_0$ , and the logarithm of some parameter  $\alpha$ . This parameterization is useful in regression models:  $\beta_0$  is the intercept meant to be augmented with other terms of the linear predictor, and  $\alpha$  is an overdispersion parameter used for comparison with the Poisson regression model.

However, we need to transform  $(\beta_0, \ln\alpha)$  to  $(p, r)$ . Examining [Methods and formulas](#) of [R] `nbreg` reveals the transformation as

$$p = \{1 + \alpha \exp(\beta_0)\}^{-1} \qquad r = \alpha^{-1}$$

which we apply using `nlcom`:

```
. nlcom (p:1/(1 + exp([lnalpha]_b[_cons] + _b[_cons])))
> (r:exp(-[lnalpha]_b[_cons]))

 p: 1/(1 + exp([lnalpha]_b[_cons] + _b[_cons]))
 r: exp(-[lnalpha]_b[_cons])
```

| accidents | Coef.    | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|-----------|----------|-----------|------|-------|----------------------|----------|
| p         | .0591157 | .0292857  | 2.02 | 0.044 | .0017168             | .1165146 |
| r         | .8691474 | .3639248  | 2.39 | 0.017 | .1558679             | 1.582427 |

Given the invariance of maximum likelihood estimators and the properties of the delta method, the above parameter estimates, standard errors, etc., are precisely those we would have obtained had we instead performed the Newton–Raphson optimization in the  $(p, r)$  metric.

## □ Technical note

Note how we referred to the estimate of  $\ln\alpha$  above as `[lnalpha]_b[_cons]`. This is not entirely evident from the output of `nbreg`, which is why we redisplayed the results using the `coeflegend` option so that we would know how to refer to the coefficients; [U] 13.5 Accessing coefficients and standard errors.

□

## nlcom versus eform

Many Stata estimation commands allow you to display exponentiated regression coefficients, some by default, some optionally. Known as “eform” in Stata terminology, this reparameterization serves many uses: it gives odds ratios for logistic models, hazard ratios in survival models, incidence-rate ratios in Poisson models, and relative-risk ratios in multinomial logit models, to name a few.

For example, consider the following estimation taken directly from the technical note in [R] poisson:

```
. use http://www.stata-press.com/data/r11/airline
. gen lnN = ln(n)
. poisson injuries XYZowned lnN
Iteration 0: log likelihood = -22.333875
Iteration 1: log likelihood = -22.332276
Iteration 2: log likelihood = -22.332276
Poisson regression
```

|               |   |        |
|---------------|---|--------|
| Number of obs | = | 9      |
| LR chi2(2)    | = | 19.15  |
| Prob > chi2   | = | 0.0001 |
| Pseudo R2     | = | 0.3001 |

Log likelihood = -22.332276

| injuries | Coef.    | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|----------|----------|-----------|------|-------|----------------------|----------|
| XYZowned | .6840667 | .3895877  | 1.76 | 0.079 | -.0795111            | 1.447645 |
| lnN      | 1.424169 | .3725155  | 3.82 | 0.000 | .6940517             | 2.154285 |
| _cons    | 4.863891 | .7090501  | 6.86 | 0.000 | 3.474178             | 6.253603 |

When we replay results and specify the `irr` (incidence-rate ratios) option,

```
. poisson, irr
Poisson regression
```

|               |   |        |
|---------------|---|--------|
| Number of obs | = | 9      |
| LR chi2(2)    | = | 19.15  |
| Prob > chi2   | = | 0.0001 |
| Pseudo R2     | = | 0.3001 |

Log likelihood = -22.332276

| injuries | IRR      | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|----------|----------|-----------|------|-------|----------------------|----------|
| XYZowned | 1.981921 | .7721322  | 1.76 | 0.079 | .9235678             | 4.253085 |
| lnN      | 4.154402 | 1.547579  | 3.82 | 0.000 | 2.00181              | 8.621728 |

we obtain the exponentiated regression coefficients and their estimated standard errors.

Contrast this with what we obtain if we exponentiate the coefficients manually by using `nlcom`:

```
. nlcom (E_XYZowned:exp(_b[XYZowned])) (E_lnN:exp(_b[lnN]))
 E_XYZowned: exp(_b[XYZowned])
 E_lnN: exp(_b[lnN])
```

| injuries   | Coef.    | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|------------|----------|-----------|------|-------|----------------------|----------|
| E_XYZowned | 1.981921 | .7721322  | 2.57 | 0.010 | .4685701             | 3.495273 |
| E_lnN      | 4.154402 | 1.547579  | 2.68 | 0.007 | 1.121203             | 7.187602 |

There are three things to note when comparing `poisson`, `irr` (and `eform` in general) with `nlcom`:

1. The exponentiated coefficients and standard errors are identical. This is certainly good news.
2. The Wald test statistic ( $z$ ) and level of significance are different. When using `poisson`, `irr` and other related `eform` options, the Wald test does not change from what you would have obtained without the `eform` option, and you can see this by comparing both versions of the `poisson` output given previously.

When you use `eform`, Stata knows that what is usually desired is a test of

$$H_0: \exp(\beta) = 1$$

and not the uninformative-by-comparison

$$H_0: \exp(\beta) = 0$$

The test of  $H_0: \exp(\beta) = 1$  is asymptotically equivalent to a test of  $H_0: \beta = 0$ , the Wald test in the original metric, but the latter has better small-sample properties. Thus if you specify `eform`, you get a test of  $H_0: \beta = 0$ .

`nlcom`, however, is general. It does not attempt to infer the test of greatest interest for a given transformation, and so a test of

$$H_0: \text{transformed coefficient} = 0$$

is always given, regardless of the transformation.

3. You may be surprised to see that, even though the coefficients and standard errors are identical, the confidence intervals (both 95%) are different.

`eform` confidence intervals are standard confidence intervals with the endpoints transformed. For example, the confidence interval for the coefficient on `lnN` is  $[0.694, 2.154]$ , whereas the confidence interval for the incidence-rate ratio due to `lnN` is  $[\exp(0.694), \exp(2.154)] = [2.002, 8.619]$ , which, except for some roundoff error, is what we see from the output of `poisson`, `irr`. For exponentiated coefficients, confidence intervals based on transform-the-endpoints methodology generally have better small-sample properties than their asymptotically equivalent counterparts.

The transform-the-endpoints method, however, gives valid coverage only when the transformation is monotonic. `nlcom` uses a more general and asymptotically equivalent method for calculating confidence intervals, as described in [Methods and formulas](#).

## Saved results

`nlcom` saves the following in `r()`:

### Scalars

`r(N)`            number of observations  
`r(df_r)`        residual degrees of freedom

### Matrices

`r(b)`            vector of transformed coefficients  
`r(V)`            estimated variance–covariance matrix of the transformed coefficients

If `post` is specified, `nlcom` also saves the following in `e()`:

### Scalars

`e(N)`            number of observations  
`e(df_r)`        residual degrees of freedom  
`e(N_strata)`    number of strata  $L$ , if used after `svy`  
`e(N_psu)`       number of sampled PSUs  $n$ , if used after `svy`  
`e(rank)`        rank of `e(V)`

### Macros

`e(cmd)`            `nlcom`  
`e(predict)`       program used to implement `predict`  
`e(properties)`    `b V`

### Matrices

`e(b)`            vector of transformed coefficients  
`e(V)`            estimated variance–covariance matrix of the transformed coefficients  
`e(V_srs)`       simple-random-sampling-without-replacement (co)variance  $\widehat{V}_{\text{srswor}}$ , if `svy`  
`e(V_srswr)`     simple-random-sampling-with-replacement (co)variance  $\widehat{V}_{\text{srswr}}$ , if `svy` and `fpc()`  
`e(V_msp)`       misspecification (co)variance  $\widehat{V}_{\text{msp}}$ , if `svy` and available

### Functions

`e(sample)`       marks estimation sample

## Methods and formulas

`nlcom` is implemented as an ado-file.

Given a  $1 \times k$  vector of parameter estimates,  $\widehat{\theta} = (\widehat{\theta}_1, \dots, \widehat{\theta}_k)$ , consider the estimated  $p$ -dimensional transformation

$$g(\widehat{\theta}) = [g_1(\widehat{\theta}), g_2(\widehat{\theta}), \dots, g_p(\widehat{\theta})]$$

The estimated variance–covariance of  $g(\widehat{\theta})$  is given by

$$\widehat{\text{Var}} \{g(\widehat{\theta})\} = \mathbf{G} \mathbf{V} \mathbf{G}'$$

where  $\mathbf{G}$  is the  $p \times k$  matrix of derivatives for which

$$\mathbf{G}_{ij} = \left. \frac{\partial g_i(\boldsymbol{\theta})}{\partial \theta_j} \right|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}} \quad i = 1, \dots, p \quad j = 1, \dots, k$$

and  $\mathbf{V}$  is the estimated variance–covariance matrix of  $\hat{\boldsymbol{\theta}}$ . Standard errors are obtained as the square roots of the variances.

The Wald test statistic for testing

$$H_0: g_i(\boldsymbol{\theta}) = 0$$

versus the two-sided alternative is given by

$$Z_i = \frac{g_i(\hat{\boldsymbol{\theta}})}{\left[ \widehat{\text{Var}}_{ii} \left\{ g(\hat{\boldsymbol{\theta}}) \right\} \right]^{1/2}}$$

When the variance–covariance matrix of  $\hat{\boldsymbol{\theta}}$  is an asymptotic covariance matrix,  $Z_i$  is approximately distributed as Gaussian. For linear regression,  $Z_i$  is taken to be approximately distributed as  $t_{1,r}$  where  $r$  is the residual degrees of freedom from the original fitted model.

A  $(1 - \alpha) \times 100\%$  confidence interval for  $g_i(\boldsymbol{\theta})$  is given by

$$g_i(\hat{\boldsymbol{\theta}}) \pm z_{\alpha/2} \left[ \widehat{\text{Var}}_{ii} \left\{ g(\hat{\boldsymbol{\theta}}) \right\} \right]^{1/2}$$

for those cases where  $Z_i$  is Gaussian and

$$g_i(\hat{\boldsymbol{\theta}}) \pm t_{\alpha/2,r} \left[ \widehat{\text{Var}}_{ii} \left\{ g(\hat{\boldsymbol{\theta}}) \right\} \right]^{1/2}$$

for those cases where  $Z_i$  is  $t$  distributed.  $z_p$  is the  $1 - p$  quantile of the standard normal distribution, and  $t_{p,r}$  is the  $1 - p$  quantile of the  $t$  distribution with  $r$  degrees of freedom.

## References

- Feiveson, A. H. 1999. FAQ: What is the delta method and how is it used to estimate the standard error of a transformed parameter? <http://www.stata.com/support/faqs/stat/deltam.html>.
- Gould, W. W. 1996. [crc43: Wald test of nonlinear hypotheses after model estimation](#). *Stata Technical Bulletin* 29: 2–4. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 15–18. College Station, TX: Stata Press.
- Oehlert, G. W. 1992. A note on the delta method. *American Statistician* 46: 27–29.
- Phillips, P. C. B., and J. Y. Park. 1988. On the formulation of Wald tests of nonlinear restrictions. *Econometrica* 56: 1065–1083.

## Also see

- [R] [lincom](#) — Linear combinations of estimators
- [R] [predictnl](#) — Obtain nonlinear predictions, standard errors, etc., after estimation
- [R] [test](#) — Test linear hypotheses after estimation
- [R] [testnl](#) — Test nonlinear hypotheses after estimation
- [U] [20 Estimation and postestimation commands](#)



## Syntax

*Nested logit regression*

```
nlogit depvar [indepvars] [if] [in] [weight] [| | lev1_equation
 [| | lev2_equation ...] | | altvar: [byaltvarlist] , case(varname) [options]
```

where the syntax of *lev#\_equation* is

```
altvar: [byaltvarlist] [, base(#|lbl) estconst]
```

*Create variable based on specification of branches*

```
nlogitgen newaltvar = altvar (branchlist) [, nolog]
```

where *branchlist* is

```
branch, branch [, branch ...]
```

and *branch* is

```
[label:] alternative [| alternative [| alternative ...]]
```

*Display tree structure*

```
nlogittree altvarlist [if] [in] [weight] [, choice(depvar) nolabel nobranches]
```

| options                                      | description                                                                                                                                              |
|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                        |                                                                                                                                                          |
| * <code>case(<i>varname</i>)</code>          | use <i>varname</i> to identify cases                                                                                                                     |
| <code>base(<i>#</i>   <i>lbl</i>)</code>     | use the specified level or label of <i>altvar</i> as the base alternative for the bottom level                                                           |
| <code>noconstant</code>                      | suppress the constant terms for the bottom-level alternatives                                                                                            |
| <code>nonnormalized</code>                   | use the nonnormalized parameterization                                                                                                                   |
| <code>altwise</code>                         | use alternatively deletion instead of casewise deletion                                                                                                  |
| <code>constraints(<i>constraints</i>)</code> | apply specified linear constraints                                                                                                                       |
| <code>collinear</code>                       | keep collinear variables                                                                                                                                 |
| SE/Robust                                    |                                                                                                                                                          |
| <code>vce(<i>vcetype</i>)</code>             | <i>vcetype</i> may be <code>oim</code> , <code>robust</code> , <code>cluster <i>clustvar</i></code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                                    |                                                                                                                                                          |
| <code>level(<i>#</i>)</code>                 | set confidence level; default is <code>level(95)</code>                                                                                                  |
| <code>notree</code>                          | suppress display of tree-structure output; see also <code>nlogittree</code> options <code>no label</code> and <code>no branches</code>                   |
| <code>nocnsreport</code>                     | do not display constraints                                                                                                                               |
| Maximization                                 |                                                                                                                                                          |
| <code>maximize_options</code>                | control the maximization process; seldom used                                                                                                            |

\* `case(varname)` is required.

`bootstrap`, `by`, `jackknife`, `statsby`, and `xi` are allowed; see [U] 11.1.10 Prefix commands.

Weights are not allowed with the `bootstrap` prefix.

`fweights`, `iweights`, and `pweights` are allowed with `nlogit`, and `fweights` are allowed with `nlogittree`; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

## Menu

### nlogit

Statistics > Categorical outcomes > Nested logit regression

### nlogitgen

Statistics > Categorical outcomes > Setup for nested logit regression

### nlogittree

Statistics > Categorical outcomes > Display nested logit tree structure

## Description

`nlogit` performs full information maximum-likelihood estimation for nested logit models. These models relax the assumption of independently distributed errors and the independence of irrelevant alternatives inherent in conditional and multinomial logit models by clustering similar alternatives into nests.

By default, `nlogit` uses a parameterization that is consistent with random utility maximization (RUM). Before version 10 of Stata, a nonnormalized version of the nested logit model was fit, which you can request by specifying the `nonnormalized` option.

You must use `nlogitgen` to generate a new categorical variable to specify the branches of the decision tree before calling `nlogit`.

## Options

### Specification and options for `lev#_equation`

*altvar* is a variable identifying alternatives at this level of the hierarchy.

*byaltvarlist* specifies the variables to be used to compute the by-alternative regression coefficients for that level. For each variable specified in the variable list, there will be one regression coefficient for each alternative of that level of the hierarchy. If the variable is constant across each alternative (a case-specific variable), the regression coefficient associated with the base alternative is not identifiable. These regression coefficients are labeled as (base) in the regression table. If the variable varies among the alternatives, a regression coefficient is estimated for each alternative.

`base(#|lbl)` can be specified in each level equation where it identifies the base alternative to be used at that level. The default is the alternative that has the highest frequency.

If `vce(bootstrap)` or `vce(jackknife)` is specified, you must specify the base alternative for each level that has a *byaltvarlist* or if the constants will be estimated. Doing so ensures that the same model is fit with each call to `nlogit`.

`estconst` applies to all the level equations except the bottom-level equation. Specifying `estconst` requests that constants for each alternative (except the base alternative) be estimated. By default, no constant is estimated at these levels. Constants can be estimated in only one level of the tree hierarchy. If you specify `estconst` for one of the level equations, you must specify `noconstant` for the bottom-level equation.

### Options for `nlogit`

#### Model

`case(varname)` specifies the variable that identifies each case. `case()` is required.

`base(#|lbl)` can be specified in each level equation where it identifies the base alternative to be used at that level. The default is the alternative that has the highest frequency.

If `vce(bootstrap)` or `vce(jackknife)` is specified, you must specify the base alternative for each level that has a *byaltvarlist* or if the constants will be estimated. Doing so ensures that the same model is fit with each call to `nlogit`.

`noconstant` applies only to the equation defining the bottom level of the hierarchy. By default, constants are estimated for each alternative of *altvar*, less the base alternative. To suppress the constant terms for this level, specify `noconstant`. If you do not specify `noconstant`, you cannot specify `estconst` for the higher-level equations.

`nonnormalized` requests a nonnormalized parameterization of the model that does not scale the inclusive values by the degree of dissimilarity of the alternatives within each nest. Use this option to replicate results from older versions of Stata. The default is to use the RUM-consistent parameterization.

**altwise** specifies that alternatively deletion be used when marking out observations because of missing values in your variables. The default is to use casewise deletion. This option does not apply to observations that are marked out by the **if** or **in** qualifier or the **by** prefix.

**constraints**(*constraints*); see [\[R\] estimation options](#).

The inclusive-valued/dissimilarity parameters are parameterized as **m1** ancillary parameters. They are labeled as `[alternative_tau]const`, where *alternative* is one of the alternatives defining a branch in the tree. To constrain the inclusive-valued/dissimilarity parameter for alternative **a1** to be, say, equal to alternative **a2**, you would use the following syntax:

```
. constraint 1 [a1_tau]cons = [a2_tau]cons
. nlogit ..., constraints(1)
```

**collinear** prevents collinear variables from being dropped. Use this option when you know that you have collinear variables and you are applying **constraints()** to handle the rank reduction. See [\[R\] estimation options](#) for details on using **collinear** with **constraints()**.

---

#### SE/Robust

**vce**(*vcetype*) specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [\[R\] vce\\_option](#). If **vce**(**robust**) or **vce**(**cluster** *clustvar*) is specified, the likelihood-ratio test for the independence of irrelevant alternatives (IIA) is not computed.

---

#### Reporting

**level**(#); see [\[R\] estimation options](#).

**notree** specifies that the tree structure of the nested logit model not be displayed. See also **no label** and **nobranches** below for when **notree** is not specified.

**nocnsreport**; see [\[R\] estimation options](#).

---

#### Maximization

**maximize\_options**: **difficult**, **technique**(*algorithm\_spec*), **iterate**(#), **[no]log**, **trace**, **gradient**, **showstep**, **hessian**, **showtolerance**, **tolerance**(#), **ltolerance**(#), **nrtolerance**(#), **nonrtolerance**, **from**(*init\_specs*); see [\[R\] maximize](#). These options are seldom used.

The **technique**(**bhhh**) option is not allowed.

## Specification and options for **nlogitgen**

**newaltvar** and **altvar** are variables identifying alternatives at each level of the hierarchy.

**label** defines a label to associate with the branch. If no label is given, a numeric value is used.

**alternative** specifies an alternative, of **altvar** specified in the syntax, to be included in the branch. It is either a numeric value or the label associated with that value. An example of **nlogitgen** is

```
. nlogitgen type = restaurant(fast: 1 | 2,
> family: CafeEccell | LosNortenos | WingsNmore, fancy: 6 | 7)
```

**nolog** suppresses the display of the iteration log.

## Specification and options for nlogittree

Main

*altvarlist* is a list of alternative variables that define the tree hierarchy. The first variable must define bottom-level alternatives, and the order continues to the variable defining the top-level alternatives.

*choice(depvar)* defines the choice indicator variable and forces `nlogittree` to compute and display choice frequencies for each bottom-level alternative.

*no label* forces `nlogittree` to suppress value labels in tree-structure output.

*no branches* forces `nlogittree` to suppress drawing branches in the tree-structure output.

## Remarks

Remarks are presented under the following headings:

*Introduction*

*Data setup and the tree structure*

*Estimation*

*Testing for the IIA*

*Nonnormalized model*

## Introduction

`nlogit` performs full information maximum-likelihood estimation for nested logit models. These models relax the assumption of independently distributed errors and the IIA inherent in conditional and multinomial logit models by clustering similar alternatives into nests. Because the nested logit model is a direct generalization of the alternative-specific conditional logit model (also known as McFadden's choice model), you may want to read [R] [asclogit](#) before continuing.

By default, `nlogit` uses a parameterization that is consistent with RUM. Before version 10 of Stata, a nonnormalized version of the nested logit model was fit, which you can request by specifying the `nonnormalized` option. We recommend using the RUM-consistent version of the model for new projects because it is based on a sound model of consumer behavior.

McFadden (1977, 1981) showed how this model can be derived from a rational choice framework. Amemiya (1985, chap. 9) contains a nice discussion of how this model can be derived under the assumption of utility maximization. Hensher, Rose, and Greene (2005) provide a lucid introduction to choice models including nested logit.

Throughout this entry, we consider a model of restaurant choice. We begin by introducing the data.

### ► Example 1

We have fictional data on 300 families and their choice of seven local restaurants. Freebirds and Mama's Pizza are fast food restaurants; Café Eccell, Los Norteños, and Wings 'N More are family restaurants; and Christopher's and Mad Cows are fancy restaurants. We want to model the decision of where to eat as a function of household income (`income`, in thousands of dollars), the number of children in the household (`kids`), the rating of the restaurant according to a local restaurant guide (`rating`, coded 0–5), the average meal cost per person (`cost`), and the distance between the household and the restaurant (`distance`, in miles). `income` and `kids` are attributes of the family, `rating` is an attribute of the alternative (the restaurant), and `cost` and `distance` are attributes of the alternative as perceived by the families—that is, each family has its own cost and distance for each restaurant.

We begin by loading the data and listing some of the variables for the first three families:

```
. use http://www.stata-press.com/data/r11/restaurant
. describe
Contains data from http://www.stata-press.com/data/r11/restaurant.dta
 obs: 2,100
 vars: 8 10 Mar 2009 01:17
 size: 75,600 (99.9% of memory free)
```

| variable name | storage<br>type | display<br>format | value<br>label | variable label                          |
|---------------|-----------------|-------------------|----------------|-----------------------------------------|
| family_id     | float           | %9.0g             |                | family ID                               |
| restaurant    | float           | %12.0g            | names          | choices of restaurants                  |
| income        | float           | %9.0g             |                | household income                        |
| cost          | float           | %9.0g             |                | average meal cost per person            |
| kids          | float           | %9.0g             |                | number of kids in the household         |
| rating        | float           | %9.0g             |                | ratings in local restaurant<br>guide    |
| distance      | float           | %9.0g             |                | distance between home and<br>restaurant |
| chosen        | float           | %9.0g             |                | 0 no 1 yes                              |

Sorted by: family\_id

```
. list family_id restaurant chosen kids rating distance in 1/21, sepby(fam)
> abbrev(10)
```

|     | family_id | restaurant   | chosen | kids | rating | distance |
|-----|-----------|--------------|--------|------|--------|----------|
| 1.  | 1         | Freebirds    | 1      | 1    | 0      | 1.245553 |
| 2.  | 1         | MamasPizza   | 0      | 1    | 1      | 2.82493  |
| 3.  | 1         | CafeEccell   | 0      | 1    | 2      | 4.21293  |
| 4.  | 1         | LosNortenos  | 0      | 1    | 3      | 4.167634 |
| 5.  | 1         | WingsNmore   | 0      | 1    | 2      | 6.330531 |
| 6.  | 1         | Christophers | 0      | 1    | 4      | 10.19829 |
| 7.  | 1         | MadCows      | 0      | 1    | 5      | 5.601388 |
| 8.  | 2         | Freebirds    | 0      | 3    | 0      | 4.162657 |
| 9.  | 2         | MamasPizza   | 0      | 3    | 1      | 2.865081 |
| 10. | 2         | CafeEccell   | 0      | 3    | 2      | 5.337799 |
| 11. | 2         | LosNortenos  | 1      | 3    | 3      | 4.282864 |
| 12. | 2         | WingsNmore   | 0      | 3    | 2      | 8.133914 |
| 13. | 2         | Christophers | 0      | 3    | 4      | 8.664631 |
| 14. | 2         | MadCows      | 0      | 3    | 5      | 9.119597 |
| 15. | 3         | Freebirds    | 1      | 3    | 0      | 2.112586 |
| 16. | 3         | MamasPizza   | 0      | 3    | 1      | 2.215329 |
| 17. | 3         | CafeEccell   | 0      | 3    | 2      | 6.978715 |
| 18. | 3         | LosNortenos  | 0      | 3    | 3      | 5.117877 |
| 19. | 3         | WingsNmore   | 0      | 3    | 2      | 5.312941 |
| 20. | 3         | Christophers | 0      | 3    | 4      | 9.551273 |
| 21. | 3         | MadCows      | 0      | 3    | 5      | 5.539806 |

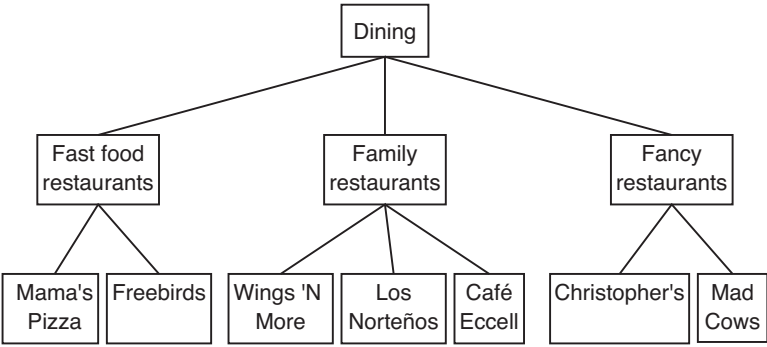
Because each family chose among seven restaurants, there are 7 observations in the dataset for each family. The variable `chosen` is coded 0/1, with 1 indicating the chosen restaurant and 0 otherwise.

◀

We could fit a conditional logit model to our data. Because `income` and `kids` are constant within each family, we would use the `asclogit` command instead of `clogit`. However, the conditional logit may be inappropriate. That model assumes that the random errors are independent, and as a result it forces the odds ratio of any two alternatives to be independent of the other alternatives, a property known as the IIA. We will discuss the IIA assumption in more detail later.

Assuming that unobserved shocks influencing a decision maker's attitude toward one alternative have no effect on his attitudes toward the other alternatives may seem innocuous, but often this assumption is too restrictive. Suppose that when a family was deciding which restaurant to visit, they were pressed for time because of plans to attend a movie later. The unobserved shock (being in a hurry) would raise the likelihood that the family goes to either fast food restaurant (Freebirds or Mama's Pizza). Similarly, another family might be choosing a restaurant to celebrate a birthday and therefore be inclined to attend a fancy restaurant (Christopher's or Mad Cows).

Nested logit models relax the independence assumption and allow us to group alternatives for which unobserved shocks may have concomitant effects. Here we suspect that restaurants should be grouped by type (fast, family, or fancy). The tree structure of a family's decision about where to eat might look like this:



At the bottom of the tree are the individual restaurants, indicating that there are some random shocks that affect a family’s decision to eat at each restaurant independently. Above the restaurants are the three types of restaurants, indicating that other random shocks affect the type of restaurant chosen. As is customary when drawing decision trees, at the top level is one box, representing the family making the decision.

- We use the following terms to describe nested logit models.
- level*, or decision level, is the level or stage at which a decision is made. The example above has only two levels. In the first level, a type of restaurant is chosen—fast food, family, or fancy—and in the second level, a specific restaurant is chosen.
  - bottom level* is the level where the final decision is made. In our example, this is when we choose a specific restaurant.
  - alternative set* is the set of all possible alternatives at any given decision level.
  - bottom alternative set* is the set of all possible alternatives at the bottom level. This concept is often referred to as the choice set in the economics-choice literature. In our example, the bottom alternative set is all seven of the specific restaurants.
  - alternative* is a specific alternative within an alternative set. In the first level of our example, “fast food” is an alternative. In the second or bottom level, “Mad Cows” is an alternative. Not all alternatives within an alternative set are available to someone making a choice at a specific stage, only those that are nested within all higher-level decisions.
  - chosen alternative* is the alternative from an alternative set that we observe someone having chosen.

□ Technical note

Although decision trees in nested logit analysis are often interpreted as implying that the highest-level decisions are made first, followed by decisions at lower levels, and finally the decision among alternatives at the bottom level, no such temporal ordering is implied. See [Hensher, Rose, and Greene \(2005, chap. 13\)](#). In our example, we are not assuming that families first choose whether to attend a fast, family, or fancy restaurant and then choose the particular restaurant; we assume merely that they choose one of the seven restaurants.

□



## Data setup and the tree structure

To fit a nested logit model, you must first create a variable that defines the structure of your decision tree.

### ► Example 2

To run `nlogit`, we need to generate a categorical variable that identifies the first-level set of alternatives: fast food, family restaurants, or fancy restaurants. We can do so easily by using `nlogitgen`.

```
. nlogitgen type = restaurant(fast: Freebirds | MamasPizza,
> family: CafeEccell | LosNortenos| WingsNmore, fancy: Christophers | MadCows)
new variable type is generated with 3 groups
label list lb_type
lb_type:
 1 fast
 2 family
 3 fancy
```

```
. nlogittree restaurant type, choice(chosen)
tree structure specified for the nested logit model
```

| type   | N   |    | restaurant   | N    | k   |
|--------|-----|----|--------------|------|-----|
| fast   | 600 | └─ | Freebirds    | 300  | 12  |
|        |     |    | MamasPizza   | 300  | 15  |
| family | 900 | └─ | CafeEccell   | 300  | 78  |
|        |     |    | LosNortenos  | 300  | 75  |
|        |     |    | WingsNmore   | 300  | 69  |
| fancy  | 600 | └─ | Christophers | 300  | 27  |
|        |     |    | MadCows      | 300  | 24  |
| total  |     |    |              | 2100 | 300 |

k = number of times alternative is chosen

N = number of observations at each level

The new categorical variable is `type`, which takes on value 1 (fast) if `restaurant` is Freebirds or Mama's Pizza; value 2 (family) if `restaurant` is Café Eccell, Los Norteños, or Wings 'N More; and value 3 (fancy) otherwise. `nlogittree` displays the tree structure.



### □ Technical note

We could also use values instead of value labels of `restaurant` in `nlogitgen`. Value labels are optional, and the default value labels for `type` are `type1`, `type2`, and `type3`. The vertical bar is also optional.

```
. use http://www.stata-press.com/data/r11/restaurant, clear
. nlogitgen type = restaurant(1 2, 3 4 5, 6 7)
new variable type is generated with 3 groups
label list lb_type
lb_type:
 1 type1
 2 type2
 3 type3
```

```
. nlogittree restaurant type
tree structure specified for the nested logit model
type N restaurant N

type1 600 └─ Freebirds 300
 └─ MamasPizza 300
type2 900 └─ CafeEccell 300
 └─ LosNortenos 300
 └─ WingsNmore 300
type3 600 └─ Christophers 300
 └─ MadCows 300
total 2100
N = number of observations at each level
```



In our dataset, every family was able to choose among all seven restaurants. However, in other applications some decision makers may not have been able to choose among all possible alternatives. For example, two cases may have choice hierarchies of

| case 1 |                 | case 2 |                |
|--------|-----------------|--------|----------------|
| type   | restaurant      | type   | restaurant     |
| fast   | └─ Freebirds    | fast   | └─ Freebirds   |
|        | └─ MamasPizza   |        | └─ MamasPizza  |
| family | └─ CafeEccell   | family | └─ LosNortenos |
|        | └─ LosNortenos  |        | └─ WingsNmore  |
|        | └─ WingsNmore   |        |                |
| fancy  | └─ Christophers | fancy  | — Christophers |
|        | └─ MadCows      |        |                |

where the second case does not have the restaurant alternatives Café Eccell or Mad Cows available to them. The only restriction is that the relationships between higher- and lower-level alternative sets be the same for all decision makers. In this two-level example, Freebirds and Mama’s Pizza are classified as fast food restaurants for both cases; Café Eccell, Los Norteños, and Wings ’N More are family restaurants; and Christopher’s and Mad Cows are fancy restaurants. `nlogit` requires only that hierarchy be maintained for all cases.

Estimation

➤ Example 3

With our `type` variable created that defines the three types of restaurants, we can now examine how the alternative-specific attributes (`cost`, `rating`, and `distance`) apply to the bottom alternative set (the seven restaurants) and how family-specific attributes (`income` and `kid`) apply to the alternative set at the first decision level (the three types of restaurants).

```

. nlogit chosen cost rating distance || type: income kids, base(family) ||
> restaurant:, noconstant case(family_id)
tree structure specified for the nested logit model

```

| type   | N   | restaurant   | N    | k   |
|--------|-----|--------------|------|-----|
| fast   | 600 | Freebirds    | 300  | 12  |
|        |     | MamasPizza   | 300  | 15  |
| family | 900 | CafeEccell   | 300  | 78  |
|        |     | LosNortenos  | 300  | 75  |
|        |     | WingsNmore   | 300  | 69  |
| fancy  | 600 | Christophers | 300  | 27  |
|        |     | MadCows      | 300  | 24  |
| total  |     |              | 2100 | 300 |

k = number of times alternative is chosen  
 N = number of observations at each level  
 Iteration 0: log likelihood = -541.93581  
 (output omitted)  
 Iteration 17: log likelihood = -485.47331  
 RUM-consistent nested logit regression  
 Case variable: family\_id  
 Alternative variable: restaurant

|                    |   |        |
|--------------------|---|--------|
| Number of obs      | = | 2100   |
| Number of cases    | = | 300    |
| Alts per case: min | = | 7      |
| avg                | = | 7.0    |
| max                | = | 7      |
| Wald chi2(7)       | = | 46.71  |
| Prob > chi2        | = | 0.0000 |

Log likelihood = -485.47331

| chosen     | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|------------|-----------|-----------|-------|-------|----------------------|-----------|
| restaurant |           |           |       |       |                      |           |
| cost       | -.1843847 | .0933975  | -1.97 | 0.048 | -.3674404            | -.0013289 |
| rating     | .463694   | .3264935  | 1.42  | 0.156 | -.1762215            | 1.10361   |
| distance   | -.3797474 | .1003828  | -3.78 | 0.000 | -.5764941            | -.1830007 |

type equations

|        |           |          |       |       |           |           |
|--------|-----------|----------|-------|-------|-----------|-----------|
| fast   |           |          |       |       |           |           |
| income | -.0266038 | .0117306 | -2.27 | 0.023 | -.0495952 | -.0036123 |
| kids   | -.0872584 | .1385026 | -0.63 | 0.529 | -.3587184 | .1842016  |
| family |           |          |       |       |           |           |
| income | (base)    |          |       |       |           |           |
| kids   | (base)    |          |       |       |           |           |
| fancy  |           |          |       |       |           |           |
| income | .0461827  | .0090936 | 5.08  | 0.000 | .0283595  | .0640059  |
| kids   | -.3959413 | .1220356 | -3.24 | 0.001 | -.6351267 | -.1567559 |

dissimilarity parameters

| type        |          |          |  |           |          |
|-------------|----------|----------|--|-----------|----------|
| /fast_tau   | 1.712878 | 1.48685  |  | -1.201295 | 4.627051 |
| /family_tau | 2.505113 | .9646351 |  | .614463   | 4.395763 |
| /fancy_tau  | 4.099844 | 2.810123 |  | -1.407896 | 9.607583 |

LR test for IIA (tau = 1): chi2(3) = 6.87 Prob > chi2 = 0.0762

First, let's examine how we called `nlogit`. The delimiters (`||`) separate equations. The first equation specifies the dependent variable, `chosen`, and three alternative-specific variables, `cost`,

rating, and distance. We refer to these variables as alternative-specific because they vary among the bottom-level alternatives, the restaurants. We obtain one parameter estimate for each variable. These estimates are listed in the equation subtable labeled `restaurant`.

For the second equation, we specify the `type` variable. It identifies the first-level alternatives, the restaurant types. Following the colon after `type`, we specify two case-specific variables, `income` and `kids`. Here we obtain a parameter estimate for each variable for each alternative at this level. That is why we call these variable lists *by-alternative* variables. Because `income` and `kids` do not vary within each case, to identify the model one alternative's set of parameters must be set to zero. We specified the `base(family)` option with this equation to restrict the parameters for the `family` alternative.

The variable identifying the bottom-level alternatives, `restaurant`, is specified after the second equation delimiter. We do not specify any variables after the colon delimiter at this level. Had we specified variables here, we would have obtained an estimate for each variable in each equation. As we will see below, these variables parameterize the constant term in the utility equation for each bottom-level alternative. The `noconstant` option suppresses bottom-level alternative-specific constant terms.

Near the bottom of the output are the dissimilarity parameters, which measure the degree of correlation of random shocks within each of the three types of restaurants. Dissimilarity parameters greater than one imply that the model is inconsistent with RUM; [Hensher, Rose, and Greene \(2005, sec. 13.6\)](#) discuss this in detail. We will ignore the fact that all our dissimilarity parameters exceed one.

The conditional logit model is a special case of nested logit in which all the dissimilarity parameters are equal to one. At the bottom of the output, we find a likelihood-ratio test of this hypothesis. Here we have mixed evidence of the null hypothesis that all the parameters are one. Equivalently, the property known as the IIA imposed by the conditional logit model holds if and only if all dissimilarity parameters are equal to one. We discuss the IIA in more detail now.

◀

## Testing for the IIA

The IIA is a property of the multinomial and conditional logit models that forces the odds of choosing one alternative over another to be independent of the other alternatives. For simplicity, suppose that a family was choosing only between Freebirds and Mama's Pizza, and the family was equally likely to choose either of the restaurants. The probability of going to each restaurant is 50%. Now suppose that Bill's Burritos opens up next door to Freebirds, which is also a burrito restaurant. If the IIA holds, then the probability of going to each restaurant must now be 33.33% so that the family remains equally likely to go to Mama's Pizza or Freebirds.

The IIA may sometimes be a plausible assumption. However, a more likely scenario would be for the probability of going to Mama's Pizza to remain at 50% and the probabilities of going to Freebirds and Bill's Burritos to be 25% each, because the two restaurants are next door to each other and serve the same food. Nested logit analysis would allow us to relax the IIA assumption of conditional logit. We could group Bill's Burritos and Freebirds into one nest that encompasses all burrito restaurants and create a second nest for pizzerias.

The IIA is a consequence of assuming that the errors are independent and identically distributed (i.i.d.). Because the errors are i.i.d., they cannot contain any alternative-specific unobserved information, and therefore adding a new alternative cannot affect the relationship between a pair of existing alternatives.

In the previous example, we saw that a joint test that the dissimilarity parameters were equal to one is one way to test for IIA. However, that test required us to specify a decision tree for the

nested logit model, and different specifications could lead to conflicting results of the test. [Hausman and McFadden \(1984\)](#) suggest that if part of the choice set truly is irrelevant with respect to the other alternatives, omitting that subset from the conditional logit model will not lead to inconsistent estimates. Therefore, Hausman's (1978) specification test can be used to test for IIA, and this test will not be sensitive to the tree structure we specify for a nested logit model.

## ► Example 4

We want to test the IIA for the subset of family restaurants against the alternatives of fast food and fancy restaurants. To do so, we need to use Stata's `hausman` command; see [\[R\] hausman](#).

We first run the estimation on the full bottom alternative set, save the results by using `estimates store`, and then run the estimation on the bottom alternative set, excluding the alternatives of family restaurants. We then run the `hausman` test.

```
. gen incFast = (type == 1) * income
. gen incFancy = (type == 3) * income
. gen kidFast = (type == 1) * kids
. gen kidFancy = (type == 3) * kids
. clogit chosen cost rating dist incFast incFancy kidFast kidFancy,
> group(family_id) nolog
```

|                                                 |               |   |        |
|-------------------------------------------------|---------------|---|--------|
| Conditional (fixed-effects) logistic regression | Number of obs | = | 2100   |
|                                                 | LR chi2(7)    | = | 189.73 |
|                                                 | Prob > chi2   | = | 0.0000 |
|                                                 | Pseudo R2     | = | 0.1625 |

Log likelihood = -488.90834

| chosen   | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| cost     | -.1367799 | .0358479  | -3.82 | 0.000 | -.2070404            | -.0665193 |
| rating   | .3066622  | .1418291  | 2.16  | 0.031 | .0286823             | .584642   |
| distance | -.1977505 | .0471653  | -4.19 | 0.000 | -.2901927            | -.1053082 |
| incFast  | -.0390183 | .0094018  | -4.15 | 0.000 | -.0574455            | -.0205911 |
| incFancy | .0407053  | .0080405  | 5.06  | 0.000 | .0249462             | .0564644  |
| kidFast  | -.2398757 | .1063674  | -2.26 | 0.024 | -.448352             | -.0313994 |
| kidFancy | -.3893862 | .1143797  | -3.40 | 0.001 | -.6135662            | -.1652061 |

```
. estimates store fullset
. clogit chosen cost rating dist incFast kidFast if type !=2, group(family_id)
> nolog
```

note: 222 groups (888 obs) dropped because of all positive or all negative outcomes.

|                                                 |               |   |        |
|-------------------------------------------------|---------------|---|--------|
| Conditional (fixed-effects) logistic regression | Number of obs | = | 312    |
|                                                 | LR chi2(5)    | = | 44.35  |
|                                                 | Prob > chi2   | = | 0.0000 |
|                                                 | Pseudo R2     | = | 0.2051 |

Log likelihood = -85.955324

| chosen   | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| cost     | -.0616621 | .067852   | -0.91 | 0.363 | -.1946496            | .0713254  |
| rating   | .1659001  | .2832041  | 0.59  | 0.558 | -.3891698            | .72097    |
| distance | -.244396  | .0995056  | -2.46 | 0.014 | -.4394234            | -.0493687 |
| incFast  | -.0737506 | .0177444  | -4.16 | 0.000 | -.108529             | -.0389721 |
| kidFast  | .4105386  | .2137051  | 1.92  | 0.055 | -.0083157            | .8293928  |

```
. hausman . fullset
```

|          | Coefficients |           |            |                     |
|----------|--------------|-----------|------------|---------------------|
|          | (b)          | (B)       | (b-B)      | sqrt(diag(V_b-V_B)) |
|          | .            | fullset   | Difference | S.E.                |
| cost     | -.0616621    | -.1367799 | .0751178   | .0576092            |
| rating   | .1659001     | .3066622  | -.1407621  | .2451308            |
| distance | -.244396     | -.1977505 | -.0466456  | .0876173            |
| incFast  | -.0737506    | -.0390183 | -.0347323  | .015049             |
| kidFast  | .4105386     | -.2398757 | .6504143   | .1853533            |

b = consistent under Ho and Ha; obtained from clogit  
B = inconsistent under Ha, efficient under Ho; obtained from clogit

Test: Ho: difference in coefficients not systematic

chi2(5) = (b-B)'[(V\_b-V\_B)^(-1)](b-B)  
= 10.70  
Prob>chi2 = 0.0577  
(V\_b-V\_B is not positive definite)

Similar to our findings in example 3, the results of the test of the IIA are mixed. We cannot reject the IIA at the commonly used 5% significance level, but we could at the 10% level. Substantively, a significant test result suggests that the odds of going to one of the fancy restaurants versus going to one of the fast food restaurants changes if we include the family restaurants in the alternative set and that a nested logit specification may be warranted.



Nonnormalized model

Previous versions of Stata fit a nonnormalized nested logit model that is available via the `nonnormalized` option. The nonnormalized version is presented in, for example, [Greene \(2008, 847–850\)](#). Here we outline the differences between the RUM-consistent and nonnormalized models. Our discussion follows [Heiss \(2002\)](#) and assumes the decision tree has two levels, with  $M$  alternatives at the upper level and a total of  $J$  alternatives at the bottom level.

In a RUM framework, by consuming alternative  $j$ , decision maker  $i$  obtains utility

$$U_{ij} = V_{ij} + \epsilon_{ij} = \alpha_j + \mathbf{x}_{ij}\boldsymbol{\beta}_j + \mathbf{z}_i\boldsymbol{\gamma}_j + \epsilon_{ij}$$

where  $V_{ij}$  is the deterministic part of utility and  $\epsilon_{ij}$  is the random part.  $\mathbf{x}_{ij}$  are alternative-specific variables and  $\mathbf{z}_i$  are case-specific variables. The set of errors  $\epsilon_{i1}, \dots, \epsilon_{iJ}$  are assumed to follow the generalized extreme-value (GEV) distribution, which is a generalization of the type 1 extreme-value distribution that allows for alternatives within nests of the tree structure to be correlated. Let  $\rho_m$  denote the correlation in nest  $m$ , and define the dissimilarity parameter  $\tau_m = \sqrt{1 - \rho_m}$ .  $\tau_m = 0$  implies that the alternatives in nest  $m$  are perfectly correlated, whereas  $\tau_m = 1$  implies independence.

The *inclusive value* for the  $m$ th nest corresponds to the expected value of the utility that decision maker  $i$  obtains by consuming an alternative in nest  $m$ . Denote this value by  $\text{IV}_m$ :

$$\text{IV}_m = \ln \sum_{j \in B_m} \exp(V_k/\tau_m) \tag{1}$$

where  $B_m$  denotes the set of alternatives in nest  $m$ . Given the inclusive values, we can show that the probability that random-utility-maximizing decision maker  $i$  chooses alternative  $j$  is

$$\text{Pr}_j = \frac{\exp\{V_j/\tau(j)\}}{\exp\{\text{IV}(j)\}} \frac{\exp\{\tau(j)\text{IV}(j)\}}{\sum_m \exp(\tau_m \text{IV}_m)}$$

where  $\tau(j)$  and  $\text{IV}(j)$  are the dissimilarity parameter and inclusive value for the nest in which alternative  $j$  lies.

In contrast, for the nonnormalized model, we have a latent variable

$$\tilde{V}_{i,j} = \tilde{\alpha}_j + \mathbf{x}_{i,j}\tilde{\beta}_j + \mathbf{z}_i\tilde{\gamma}_j$$

and corresponding inclusive values

$$\widetilde{\text{IV}}_m = \ln \sum_{j \in B_m} \exp(\tilde{V}_k) \quad (2)$$

The probability of choosing alternative  $j$  is

$$\text{Pr}_j = \frac{\exp(\tilde{V}_j) \exp\{\tau(j)\widetilde{\text{IV}}(j)\}}{\exp\{\widetilde{\text{IV}}(j)\} \sum_m \exp(\tau_m \widetilde{\text{IV}}_m)}$$

Equations (1) and (2) represent the key difference between the RUM-consistent and nonnormalized models. By scaling the  $V_{ij}$  within each nest, the RUM-consistent model allows utilities to be compared across nests. Without the rescaling, utilities can be compared only for goods within the same nest. Moreover, adding a constant to each  $V_{ij}$  for consumer  $i$  will not affect the probabilities of the RUM-consistent model, but adding a constant to each  $\tilde{V}_{ij}$  will affect the probabilities from the nonnormalized model. Decisions based on utility maximization can depend only on utility differences and not the scale or zero point of the utility function because utility is an ordinal concept, so the nonnormalized model cannot be consistent with utility maximization.

Heiss (2002) showed that the nonnormalized model can be RUM consistent in the special case where all the variables are specified in the bottom-level equation. Then multiplying the nonnormalized coefficients by the respective dissimilarity parameters results in the RUM-consistent coefficients.

## □ Technical note

Degenerate nests occur when there is only one alternative in a branch of the tree hierarchy. The associated dissimilarity parameter of the RUM model is not defined. The inclusive-valued parameter of the nonnormalized model will be identifiable if there are alternative-specific variables specified in (1) of the model specification (the *indepvars* in the model syntax). Numerically, you can skirt the issue of nonidentifiable/undefined parameters by setting constraints on them. For the RUM model constraint, set the dissimilarity parameter to 1. See the description of `constraints()` in *Options* for details on setting constraints on the dissimilarity parameters.

□

## Saved results

nlogit saves the following in `e()`:

### Scalars

|                             |                                                     |
|-----------------------------|-----------------------------------------------------|
| <code>e(N)</code>           | number of observations                              |
| <code>e(N_case)</code>      | number of cases                                     |
| <code>e(k_eq)</code>        | number of equations in <code>e(b)</code>            |
| <code>e(k_eq_model)</code>  | number of equations in model Wald test              |
| <code>e(k_alt)</code>       | number of alternatives for bottom level             |
| <code>e(k_altj)</code>      | number of alternatives for $j$ th level             |
| <code>e(k_indvars)</code>   | number of independent variables                     |
| <code>e(k_ind2vars)</code>  | number of by-alternative variables for bottom level |
| <code>e(k_ind2varsj)</code> | number of by-alternative variables for $j$ th level |
| <code>e(k_autoCns)</code>   | number of base, empty, and omitted constraints      |
| <code>e(df_m)</code>        | model degrees of freedom                            |
| <code>e(df_c)</code>        | clogit model degrees of freedom                     |
| <code>e(ll)</code>          | log likelihood                                      |
| <code>e(ll_c)</code>        | clogit model log likelihood                         |
| <code>e(N_clust)</code>     | number of clusters                                  |
| <code>e(chi2)</code>        | $\chi^2$                                            |
| <code>e(chi2_c)</code>      | likelihood-ratio test for IIA                       |
| <code>e(p)</code>           | $p$ -value for model Wald test                      |
| <code>e(p_c)</code>         | $p$ -value for IIA test                             |
| <code>e(i_base)</code>      | base index for bottom level                         |
| <code>e(i_basej)</code>     | base index for $j$ th level                         |
| <code>e(levels)</code>      | number of levels                                    |
| <code>e(alt_min)</code>     | minimum number of alternatives                      |
| <code>e(alt_avg)</code>     | average number of alternatives                      |
| <code>e(alt_max)</code>     | maximum number of alternatives                      |
| <code>e(const)</code>       | constant indicator for bottom level                 |
| <code>e(constj)</code>      | constant indicator for $j$ th level                 |
| <code>e(rum)</code>         | 1 if RUM model, 0 otherwise                         |
| <code>e(rank)</code>        | rank of <code>e(V)</code>                           |
| <code>e(ic)</code>          | number of iterations                                |
| <code>e(rc)</code>          | return code                                         |
| <code>e(converged)</code>   | 1 if converged, 0 otherwise                         |



## Macros

|                                   |                                                                          |
|-----------------------------------|--------------------------------------------------------------------------|
| <code>e(cmd)</code>               | <code>nlogit</code>                                                      |
| <code>e(cmdline)</code>           | command as typed                                                         |
| <code>e(depvar)</code>            | name of dependent variable                                               |
| <code>e(indvars)</code>           | name of independent variables                                            |
| <code>e(ind2vars)</code>          | by-alternative variables for bottom level                                |
| <code>e(ind2varsj)</code>         | by-alternative variables for <i>j</i> th level                           |
| <code>e(case)</code>              | variable defining cases                                                  |
| <code>e(altvar)</code>            | alternative variable for bottom level                                    |
| <code>e(altvarj)</code>           | alternative variable for <i>j</i> th level                               |
| <code>e(alteqs)</code>            | equation names for bottom level                                          |
| <code>e(alteqsj)</code>           | equation names for <i>j</i> th level                                     |
| <code>e(alti)</code>              | <i>i</i> th alternative for bottom level                                 |
| <code>e(altj_i)</code>            | <i>i</i> th alternative for <i>j</i> th level                            |
| <code>e(wtype)</code>             | weight type                                                              |
| <code>e(wexp)</code>              | weight expression                                                        |
| <code>e(title)</code>             | title in estimation output                                               |
| <code>e(clustvar)</code>          | name of cluster variable                                                 |
| <code>e(chi2type)</code>          | Wald, type of model $\chi^2$ test                                        |
| <code>e(vce)</code>               | <i>vcetype</i> specified in <code>vce()</code>                           |
| <code>e(vcetype)</code>           | title used to label Std. Err.                                            |
| <code>e(opt)</code>               | type of optimization                                                     |
| <code>e(which)</code>             | max or min; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>         | type of ml method                                                        |
| <code>e(user)</code>              | name of likelihood-evaluator program                                     |
| <code>e(technique)</code>         | maximization technique                                                   |
| <code>e(singularHmethod)</code>   | m-marquardt or hybrid; method used when Hessian is singular              |
| <code>e(crittype)</code>          | optimization criterion                                                   |
| <code>e(datasignature)</code>     | the checksum                                                             |
| <code>e(datasignaturevars)</code> | variables used in calculation of checksum                                |
| <code>e(properties)</code>        | b V                                                                      |
| <code>e(estat_cmd)</code>         | program used to implement <code>estat</code>                             |
| <code>e(predict)</code>           | program used to implement <code>predict</code>                           |
| <code>e(marginsnotok)</code>      | predictions disallowed by <code>margins</code>                           |

(Continued on next page)

**Matrices**

|                               |                                                                                                     |
|-------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>e(b)</code>             | coefficient vector                                                                                  |
| <code>e(Cns)</code>           | constraints matrix                                                                                  |
| <code>e(k_altern)</code>      | number of alternatives at each level                                                                |
| <code>e(k_branchj)</code>     | number of branches at each alternative of $j$ th level                                              |
| <code>e(stats)</code>         | alternative statistics for bottom level                                                             |
| <code>e(statsj)</code>        | alternative statistics for $j$ th level                                                             |
| <code>e(altidxj)</code>       | alternative indices for $j$ th level                                                                |
| <code>e(alt_ind2vars)</code>  | indicators for bottom level estimated by-alternative<br>variable— $e(k\_alt) \times e(k\_ind2vars)$ |
| <code>e(alt_ind2varsj)</code> | indicators for $j$ th level estimated by-alternative variable— $e(k\_altj) \times e(k\_ind2varsj)$  |
| <code>e(ilog)</code>          | iteration log (up to 20 iterations)                                                                 |
| <code>e(gradient)</code>      | gradient vector                                                                                     |
| <code>e(V)</code>             | variance–covariance matrix of the estimators                                                        |
| <code>e(V_modelbased)</code>  | model-based variance                                                                                |

**Functions**

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`nlogit`, `nlogitgen`, and `nlogittree` are implemented as ado-files.

Methods and formulas are presented under the following headings:

*Two-level nested logit model*  
*Three-level nested logit model*

### Two-level nested logit model

Consider our two-level nested logit model for restaurant choice. We define  $T = \{1, 2, 3\}$  to be the set of indices denoting the three restaurant types and  $R_1 = \{1, 2\}$ ,  $R_2 = \{3, 4, 5\}$ , and  $R_3 = \{6, 7\}$  to be the set of indices representing each restaurant within type  $t \in T$ . Let  $C_1$  and  $C_2$  be the random variables that represent the choices made for the first level, restaurant type, and second level, restaurant, of the hierarchy, where we observe the choices  $C_1 = t, t \in T$ , and  $C_2 = j, j \in R_t$ . Let  $\mathbf{z}_t$  and  $\mathbf{x}_{tj}$ , for  $t \in T$  and  $j \in R_t$ , refer to the row vectors of explanatory variables for the first-level alternatives and bottom-level alternatives for one case, respectively. We write the utilities (latent variables) as  $U_{tj} = \mathbf{z}_t \alpha_t + \mathbf{x}_{tj} \beta_j + \epsilon_{tj} = \eta_{tj} + \epsilon_{tj}$ , where  $\alpha_t$  and  $\beta_j$  are column vectors and the  $\epsilon_{tj}$  are random disturbances. When the  $\mathbf{x}_{tj}$  are alternative specific, we can drop the indices from  $\beta$ , where we estimate one coefficient for each alternative in  $R_t$ ,  $t \in T$ . These variables are specified in the first equation of the `nlogit` syntax (see [example 3](#)).

When the random-utility framework is used to describe the choice behavior, the alternative that is chosen is the alternative that has the highest utility. Assume for our restaurant example that we choose restaurant type  $t \in T$ . For the RUM parameterization of `nlogit`, the conditional distribution of  $\epsilon_{tj}$  given choice of restaurant type  $t$  is a multivariate version of Gumbel's extreme-value distribution,

$$F_{R|T}(\epsilon | t) = \exp \left[ - \left\{ \sum_{m \in R_t} \exp(\epsilon_{tm} / \tau_t) \right\}^{\tau_t} \right] \quad (3)$$

where it has been shown that the  $\epsilon_{tj}$ ,  $j \in R_t$ , are exchangeable with correlation  $1 - \tau_t^2$ , for  $\tau_t \in (0, 1]$  (Kotz and Nadarajah 2000). For example, the probability of choosing Christopher's,  $j = 6$  given type  $t = 3$ , is

$$\begin{aligned} \Pr(C_2 = 6 \mid C_1 = 3) &= \Pr(U_{36} - U_{37} > 0) \\ &= \Pr(\epsilon_{37} \leq \epsilon_{36} + \eta_{36} - \eta_{37}) \\ &= \int_{-\infty}^{\infty} \left\{ \int_{-\infty}^{\epsilon_{36} + \eta_{36} - \eta_{37}} f_{R|T}(\epsilon_{36}, \epsilon_{37}) d\epsilon_{37} \right\} d\epsilon_{36} \end{aligned}$$

where  $f = \frac{\partial F}{\partial \epsilon_{36} \partial \epsilon_{37}}$  is the joint density function of  $\epsilon$  given  $t$ .  $U_{37}$  is the utility of eating at Mad Cows, the other fancy ( $t = 3$ ) restaurant. Amemiya (1985) demonstrates that this integral evaluates to the logistic function

$$\begin{aligned} \Pr(C_2 = 6 \mid C_1 = 3) &= \frac{\exp(\eta_{36}/\tau_3)}{\exp(\eta_{36}/\tau_3) + \exp(\eta_{37}/\tau_3)} \\ &= \frac{\exp(\mathbf{x}_{36}\beta_6/\tau_3)}{\exp(\mathbf{x}_{36}\beta_6/\tau_3) + \exp(\mathbf{x}_{37}\beta_7/\tau_3)} \end{aligned}$$

and in general

$$\Pr(C_2 = j \mid C_1 = t) = \frac{\exp(\mathbf{x}_{tj}\beta_j/\tau_t)}{\sum_{m \in R_t} \exp(\mathbf{x}_{tm}\beta_m/\tau_t)} \quad (4)$$

Letting  $\tau_t = 1$  in (3) reduces to the product of independent extreme-value distributions, and (4) reduces to the multinomial logistic function.

For the logistic function in (4), we scale the linear predictors by the dissimilarity parameters. Another formulation of the conditional probability of choosing alternative  $j \in R_t$  given choice  $t \in T$  is the logistic function without this normalization:

$$\Pr(C_2 = j \mid C_1 = t) = \frac{\exp(\mathbf{x}_{tj}\beta_j)}{\sum_{m \in R_t} \exp(\mathbf{x}_{tm}\beta_m)}$$

and this is what is used in nlogit's nonnormalized parameterization.

Amemiya (1985) defines the general form for the joint distribution of the  $\epsilon$ 's as

$$F_{T,R}(\epsilon) = \exp \left\{ - \sum_{k \in T} \theta_k \left( \sum_{m \in R_k} \exp(-\epsilon_{km}/\tau_k) \right)^{\tau_k} \right\}$$

from which the probability of choice  $t$ ,  $t \in T$  can be derived as

$$\Pr(C_1 = t) = \frac{\theta_t \left\{ \sum_{m \in R_t} \exp(\eta_{tm}/\tau_t) \right\}^{\tau_t}}{\sum_{k \in T} \theta_k \left\{ \sum_{m \in R_k} \exp(\eta_{km}/\tau_k) \right\}^{\tau_k}} \quad (5)$$

nlogit sets  $\theta_t = 1$ . Noting that

$$\begin{aligned} \left\{ \sum_{m \in R_t} \exp(\eta_{tm}/\tau_t) \right\}^{\tau_t} &= \left\{ \sum_{m \in R_t} \exp \left( \frac{\mathbf{z}_t \alpha_t + \mathbf{x}_{tm} \beta_m}{\tau_t} \right) \right\}^{\tau_t} \\ &= \exp(\mathbf{z}_t \alpha_t) \left\{ \sum_{m \in R_t} \exp(\mathbf{x}_{tm} \beta_m / \tau_t) \right\}^{\tau_t} \\ &= \exp(\mathbf{z}_t \alpha_t + \tau_t I_t) \end{aligned}$$

we define the inclusive values  $I_t$  as

$$I_t = \ln \left\{ \sum_{m \in R_t} \exp(\mathbf{x}_{tm} \boldsymbol{\beta}_m / \tau_t) \right\}$$

and we can view

$$\exp(\tau_t I_t) = \left\{ \sum_{m \in R_t} \exp(x_{tm} \boldsymbol{\beta}_m)^{1/\tau_t} \right\}^{\tau_t}$$

as a weighted average of the  $\exp(x_{tm} \boldsymbol{\beta}_m)$ , for  $m \in R_t$ . For the **nlogit** RUM parameterization, we can express (5) as

$$\Pr(C_1 = t) = \frac{\exp(\mathbf{z}_t \boldsymbol{\alpha}_t + \tau_t I_t)}{\sum_{k \in T} \exp(\mathbf{z}_k \boldsymbol{\alpha}_k + \tau_k I_k)}$$

Next we define inclusive values for the nonnormalized model to be

$$\tilde{I}_t = \ln \left\{ \sum_{m \in R_t} \exp(\mathbf{x}_{tm} \boldsymbol{\beta}_m) \right\}$$

and we express  $\Pr(C_1 = t)$  as

$$\Pr(C_1 = t) = \frac{\exp(\mathbf{z}_t \boldsymbol{\alpha}_t + \tau_t \tilde{I}_t)}{\sum_{k \in T} \exp(\mathbf{z}_k \boldsymbol{\alpha}_k + \tau_k \tilde{I}_k)} \quad (6)$$

Equation (5) is consistent with (6) only when  $\boldsymbol{\eta}_{ij} = \mathbf{x}_{ij} \boldsymbol{\beta}_j$ , so in general the **nlogit** nonnormalized model is not consistent with the RUM model.

Now assume that we have  $N$  cases where we add a third subscript,  $i$ , to denote case  $i$ ,  $i = 1, \dots, N$ . Denote  $y_{itj}$  to be a binary variable indicating the choice made by case  $i$  so that for each  $i$  only one  $y_{itj}$  is 1 and the rest are 0 for all  $t \in T$  and  $j \in R_t$ . The log likelihood for the two-level RUM-consistent model is

$$\begin{aligned} \log \ell &= \sum_{i=1}^N \sum_{k \in T} \sum_{m \in R_k} y_{ikm} \log \{ \Pr(C_{i1} = i) \Pr(C_{i2} = m | C_{i1} = i) \} \\ &= \sum_{i=1}^N \sum_{k \in T} \sum_{m \in R_k} y_{ikm} \left[ \mathbf{z}_{ik} \boldsymbol{\alpha}_k + \tau_k I_{ik} - \log \left\{ \sum_{l \in T} \exp(\mathbf{z}_{il} \boldsymbol{\alpha}_l + \tau_l I_{il}) \right\} + \right. \\ &\quad \left. \mathbf{x}_{ikm} \boldsymbol{\beta}_m / \tau_k - \log \left\{ \sum_{l \in R_k} \exp(\mathbf{x}_{ikl} \boldsymbol{\beta}_l / \tau_k) \right\} \right] \end{aligned}$$

The likelihood for the nonnormalized model has a similar form, replacing  $I$  with  $\tilde{I}$  and by not scaling  $\mathbf{x}_{ikj} \boldsymbol{\beta}_j$  by  $\tau_k$ .

### Three-level nested logit model

Here we define a three-level nested logit model that can be generalized to the four-level and higher models. As before, let the integer set  $T$  be the indices for the first level of choices. Let sets  $S_t$ ,  $t \in T$ , be mutually exclusive sets of integers representing the choices of the second level of the hierarchy. Finally, let  $R_j$ ,  $j \in S_t$ , be the bottom-level choices. Let  $U_{tjk} = \eta_{tjk} + \epsilon_{tjk}$ ,  $k \in R_j$ , and the distribution of  $\epsilon_{tjk}$  be Gumbel's multivariate extreme value of the form

$$F(\epsilon) = \exp \left( - \sum_{t \in T} \left[ \sum_{j \in S_t} \left\{ \sum_{k \in R_j} \exp(-\eta_{tjk}/\tau_j) \right\}^{\tau_j/v_t} \right]^{v_j} \right)$$

Let  $C_1$ ,  $C_2$ , and  $C_3$  represent the choice random variables for levels 1, 2, and the bottom, respectively. Then the set of conditional probabilities is

$$\begin{aligned} \Pr(C_3 = k \mid C_1 = t, C_2 = j) &= \frac{\exp(\eta_{tjk}/\tau_j)}{\sum_{l \in R_j} \exp(\eta_{tjl}/\tau_j)} \\ \Pr(C_2 = j \mid C_1 = t) &= \frac{\left\{ \sum_{k \in R_j} \exp(\eta_{tjk}/\tau_j) \right\}^{\tau_j/v_t}}{\sum_{l \in S_t} \left\{ \sum_{k \in R_l} \exp(\eta_{tlk}/\tau_l) \right\}^{\tau_l/v_t}} \\ \Pr(C_1 = t) &= \frac{\left[ \sum_{j \in S_t} \left\{ \sum_{k \in R_j} \exp(\eta_{tjk}/\tau_j) \right\}^{\tau_j/v_t} \right]^{v_t}}{\sum_{l \in T} \left[ \sum_{j \in S_l} \left\{ \sum_{k \in R_j} \exp(\eta_{ljk}/\tau_j) \right\}^{\tau_j/v_l} \right]^{v_l}} \end{aligned}$$

Assume that we can decompose the linear predictor as  $\eta_{tjk} = \mathbf{z}_t \alpha_t + \mathbf{u}_{tj} \gamma_j + \mathbf{x}_{tjk} \beta_k$ . Here  $\mathbf{z}_t$ ,  $\mathbf{u}_{tj}$ , and  $\mathbf{x}_{tjk}$  are the row vectors of explanatory variables for the first, second, and bottom levels of the hierarchy, respectively, and  $\alpha_t$ ,  $\gamma_j$ , and  $\beta_k$  are the corresponding column vectors of regression coefficients for  $t \in T$ ,  $j \in S_t$ , and  $k \in R_j$ . We then can define the inclusive values for the first and second levels as

$$\begin{aligned} I_{tj} &= \log \sum_{k \in R_j} \exp(\mathbf{x}_{tjk} \beta_k / \tau_j) \\ J_t &= \log \sum_{j \in S_t} \exp(\mathbf{u}_{tj} \gamma_j / v_t + \frac{\tau_j}{v_t} I_{tj}) \end{aligned}$$

and rewrite the probabilities

$$\begin{aligned} \Pr(C_3 = k \mid C_1 = t, C_2 = j) &= \frac{\exp(\mathbf{x}_{tjk} \beta_k / \tau_j)}{\sum_{l \in R_j} \exp(\mathbf{x}_{tjl} \beta_l / \tau_j)} \\ \Pr(C_2 = j \mid C_1 = t) &= \frac{\exp(\mathbf{u}_{tj} \gamma_j / v_t + \frac{\tau_j}{v_t} I_{tj})}{\sum_{l \in S_t} \exp(\mathbf{u}_{tl} \gamma_l / v_t + \frac{\tau_l}{v_t} I_{tl})} \\ \Pr(C_1 = t) &= \frac{\exp(\mathbf{z}_t \alpha_t + v_t J_t)}{\sum_{l \in T} \exp(\mathbf{z}_l \alpha_l + v_l J_l)} \end{aligned}$$

We add a fourth index,  $i$ , for case and define the indicator variable  $y_{litjk}$ ,  $l = 1, \dots, N$ , to indicate the choice made by case  $i$ ,  $t \in T$ ,  $j \in S_t$ , and  $k \in R_j$ . The log likelihood for the `nlogit` RUM-consistent model is

$$\begin{aligned} \ell = \sum_{i=1}^N \sum_{t \in T} \sum_{j \in S_t} \sum_{k \in R_j} y_{litjk} & \left\{ \mathbf{z}_{it} \boldsymbol{\alpha}_t + v_t J_{it} - \log \left( \sum_{m \in T} \mathbf{z}_{im} \boldsymbol{\alpha}_m + v_m J_{im} \right) + \right. \\ & \mathbf{u}_{itj} \gamma_j / v_t + \frac{\tau_j}{v_t} I_{itj} - \log \left( \sum_{m \in S_t} \mathbf{u}_{itm} \gamma_m / v_t + \frac{\tau_m}{v_t} I_{itm} \right) + \\ & \left. \mathbf{x}_{itjk} \beta_k / \tau_k - \sum_{m \in R_t} \exp(\mathbf{x}_{itjm} \beta_m / \tau_k) \right\} \end{aligned}$$

and for the nonnormalized `nlogit` model the log likelihood is

$$\begin{aligned} \ell = \sum_{i=1}^N \sum_{t \in T} \sum_{j \in S_t} \sum_{k \in R_j} y_{litjk} & \left\{ \mathbf{z}_{it} \boldsymbol{\alpha}_t + v_t J_{it} - \log \left( \sum_{m \in T} \mathbf{z}_{im} \boldsymbol{\alpha}_m + v_m J_{im} \right) + \right. \\ & \mathbf{u}_{itj} \gamma_j + \tau_j I_{itj} - \log \left( \sum_{m \in S_t} \mathbf{u}_{itm} \gamma_m + \tau_m I_{itm} \right) + \\ & \left. \mathbf{x}_{itjk} \beta_k - \sum_{m \in R_t} \exp(\mathbf{x}_{itjm} \beta_m) \right\} \end{aligned}$$

Extending the model to more than three levels is straightforward, albeit notationally cumbersome.

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

## References

- Amemiya, T. 1985. *Advanced Econometrics*. Cambridge, MA: Harvard University Press.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice-Hall.
- Hausman, J. A. 1978. Specification tests in econometrics. *Econometrica* 46: 1251–1271.
- Hausman, J. A., and D. L. McFadden. 1984. Specification tests for the multinomial logit model. *Econometrica* 52: 1219–1240.
- Heiss, F. 2002. [Structural choice analysis with nested logit models](#). *Stata Journal* 2: 227–252.
- Hensher, D. A., J. M. Rose, and W. H. Greene. 2005. *Applied Choice Analysis: A Primer*. New York: Cambridge University Press.
- Kotz, S., and S. Nadarajah. 2000. *Extreme Value Distributions: Theory and Applications*. London: Imperial College Press.
- Maddala, G. S. 1983. *Limited-Dependent and Qualitative Variables in Econometrics*. Cambridge: Cambridge University Press.
- McFadden, D. L. 1977. Quantitative methods for analyzing travel behaviour of individuals: Some recent developments. Working paper 474, Cowles Foundation. <http://cowles.econ.yale.edu/Pd/cd/d04b/d0474.pdf>.
- . 1981. Econometric models of probabilistic choice. In *Structural Analysis of Discrete Data with Econometric Applications*, ed. C. F. Manski and D. McFadden, 198–272. Cambridge, MA: MIT Press.

## Also see

- [R] **nlogit postestimation** — Postestimation tools for nlogit
- [R] **asclogit** — Alternative-specific conditional logit (McFadden’s choice) model
- [R] **mlogit** — Multinomial (polytomous) logistic regression
- [R] **ologit** — Ordered logistic regression
- [R] **rologit** — Rank-ordered logistic regression
- [R] **slogit** — Stereotype logistic regression
- [U] **20 Estimation and postestimation commands**

Description

The following postestimation command is of special interest after `nlogit`:

| command                         | description                    |
|---------------------------------|--------------------------------|
| <code>estat alternatives</code> | alternative summary statistics |

For information about this command, see [\[R\] `asmprobit postestimation`](#).

The following standard postestimation commands are also available:

| command                | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>     | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estimates</code> | cataloging estimation results                                                                       |
| <code>hausman</code>   | Hausman’s specification test                                                                        |
| <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code>    | likelihood-ratio test                                                                               |
| <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>   | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code> | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details.



## Syntax for predict

```
predict [type] newvar [if] [in] [, statistic hlevel(#) altwise]
```

```
predict [type] { stub* | newvarlist } [if] [in], scores
```

| statistic | description |
|-----------|-------------|
|-----------|-------------|

Main

|              |                                                                                                                                                                 |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>pr</b>    | predicted probabilities of choosing the alternatives at all levels of the hierarchy or at level #, where # is specified by <code>hlevel(#)</code> ; the default |
| <b>xb</b>    | linear predictors for all levels of the hierarchy or at level #, where # is specified by <code>hlevel(#)</code>                                                 |
| <b>condp</b> | predicted conditional probabilities at all levels of the hierarchy or at level #, where # is specified by <code>hlevel(#)</code>                                |
| <b>iv</b>    | inclusive values for levels 2, ..., <code>e(levels)</code> or for <code>hlevel(#)</code>                                                                        |

The inclusive value for the first-level alternatives is not used in estimation; therefore, it is not calculated.

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

**pr** calculates the probability of choosing each alternative at each level of the hierarchy. Use the `hlevel(#)` option to compute the alternative probabilities at level #. When `hlevel(#)` is not specified,  $j$  new variables must be given, where  $j$  is the number of levels, or use the `stub*` option to have `predict` generate  $j$  variables with the prefix `stub` and numbered from 1 to  $j$ . The `pr` option is the default and if one new variable is given, the probability of the bottom-level alternatives are computed. Otherwise, probabilities for all levels are computed and the `stub*` option is still valid.

**xb** calculates the linear prediction for each alternative at each level. Use the `hlevel(#)` option to compute the linear predictor at level #. When `hlevel(#)` is not specified,  $j$  new variables must be given, where  $j$  is the number of levels, or use the `stub*` option to have `predict` generate  $j$  variables with the prefix `stub` and numbered from 1 to  $j$ .

**condp** calculates the conditional probabilities for each alternative at each level. Use the `hlevel(#)` option to compute the conditional probabilities of the alternatives at level #. When `hlevel(#)` is not specified,  $j$  new variables must be given, where  $j$  is the number of levels, or use the `stub*` option to have `predict` generate  $j$  variables with the prefix `stub` and numbered from 1 to  $j$ .

**iv** calculates the inclusive value for each alternative at each level. Use the `hlevel(#)` option to compute the inclusive value at level #. There is no inclusive value at level 1. If `hlevel(#)` is not used,  $j - 1$  new variables are required, where  $j$  is the number of levels, or use `stub*` to have `predict` generate  $j - 1$  variables with the prefix `stub` and numbered from 2 to  $j$ . See [Methods and formulas](#) in [R] **nlogit** for a definition of the inclusive values.

**hlevel(#)** calculates the prediction only for hierarchy level #.

`altwise` specifies that alternativewise deletion be used when marking out observations due to missing values in your variables. The default is to use casewise deletion. The `xb` option always uses alternativewise deletion.

`scores` calculates the scores for each coefficient in `e(b)`. This option requires a new-variable list of length equal to the number of columns in `e(b)`. Otherwise, use the `stub*` option to have `predict` generate enumerated variables with prefix `stub`.

Remarks

`predict` may be used after `nlogit` to obtain the predicted values of the probabilities, the conditional probabilities, the linear predictions, and the inclusive values for each level of the nested logit model. Predicted probabilities for `nlogit` must be interpreted carefully. Probabilities are estimated for each case as a whole and not for individual observations.

➤ Example 1

Continuing with our model in [example 3](#) of [\[R\] nlogit](#), we refit the model and then examine a summary of the alternatives and their frequencies in the estimation sample.

```
. use http://www.stata-press.com/data/r11/restaurant
. nlogitgen type = restaurant(fast: Freebirds | MamasPizza,
> family: CafeEccell | LosNortenos | WingsNmore, fancy: Christophers | MadCows)
(output omitted)
. nlogit chosen cost rating distance || type: income kids, base(family) ||
> restaurant:, noconst case(family_id)
(output omitted)
. estat alternatives
Alternatives summary for type
```

| index | Alternative value | label  | Cases present | Frequency selected | Percent selected |
|-------|-------------------|--------|---------------|--------------------|------------------|
| 1     | 1                 | fast   | 600           | 27                 | 9.00             |
| 2     | 2                 | family | 900           | 222                | 74.00            |
| 3     | 3                 | fancy  | 600           | 51                 | 17.00            |

Alternatives summary for restaurant

| index | Alternative value | label        | Cases present | Frequency selected | Percent selected |
|-------|-------------------|--------------|---------------|--------------------|------------------|
| 1     | 1                 | Freebirds    | 300           | 12                 | 4.00             |
| 2     | 2                 | MamasPizza   | 300           | 15                 | 5.00             |
| 3     | 3                 | CafeEccell   | 300           | 78                 | 26.00            |
| 4     | 4                 | LosNortenos  | 300           | 75                 | 25.00            |
| 5     | 5                 | WingsNmore   | 300           | 69                 | 23.00            |
| 6     | 6                 | Christophers | 300           | 27                 | 9.00             |
| 7     | 7                 | MadCows      | 300           | 24                 | 8.00             |

Next we predict  $p2 = \Pr(\text{restaurant})$ ;  $p1 = \Pr(\text{type})$ ;  $\text{condp} = \Pr(\text{restaurant} \mid \text{type})$ ;  $\text{xb2}$ , the linear prediction for the bottom-level alternatives;  $\text{xb1}$ , the linear prediction for the first-level alternatives; and  $\text{iv}$ , the inclusive values for the bottom-level alternatives.

```
. predict p*
(option pr assumed)
. predict condp, condp hlevel(2)
. sort family_id type restaurant
. list restaurant type chosen p2 p1 condp in 1/14, sepby(family_id) divider
```

|     | restaurant   | type   | chosen | p2       | p1       | condp    |
|-----|--------------|--------|--------|----------|----------|----------|
| 1.  | Freebirds    | fast   | 1      | .0642332 | .1189609 | .5399519 |
| 2.  | MamasPizza   | fast   | 0      | .0547278 | .1189609 | .4600481 |
| 3.  | CafeEccell   | family | 0      | .284409  | .7738761 | .3675124 |
| 4.  | LosNortenos  | family | 0      | .3045242 | .7738761 | .3935051 |
| 5.  | WingsNmore   | family | 0      | .1849429 | .7738761 | .2389825 |
| 6.  | Christophers | fancy  | 0      | .0429508 | .107163  | .4007991 |
| 7.  | MadCows      | fancy  | 0      | .0642122 | .107163  | .5992009 |
|     |              |        |        |          |          |          |
| 8.  | Freebirds    | fast   | 0      | .0183578 | .0488948 | .3754559 |
| 9.  | MamasPizza   | fast   | 0      | .030537  | .0488948 | .6245441 |
| 10. | CafeEccell   | family | 0      | .2832149 | .756065  | .3745907 |
| 11. | LosNortenos  | family | 1      | .3038883 | .756065  | .4019341 |
| 12. | WingsNmore   | family | 0      | .1689618 | .756065  | .2234752 |
| 13. | Christophers | fancy  | 0      | .1041277 | .1950402 | .533878  |
| 14. | MadCows      | fancy  | 0      | .0909125 | .1950402 | .466122  |

```
. predict xb*, xb
. predict iv, iv
. list restaurant type chosen xb* iv in 1/14, sepby(family_id) divider
```

|     | restaurant   | type   | chosen | xb1       | xb2       | iv        |
|-----|--------------|--------|--------|-----------|-----------|-----------|
| 1.  | Freebirds    | fast   | 1      | -1.124805 | -1.476914 | -.9026417 |
| 2.  | MamasPizza   | fast   | 0      | -1.124805 | -1.751229 | -.9026417 |
| 3.  | CafeEccell   | family | 0      | 0         | -2.181112 | .1303341  |
| 4.  | LosNortenos  | family | 0      | 0         | -2.00992  | .1303341  |
| 5.  | WingsNmore   | family | 0      | 0         | -3.259229 | .1303341  |
| 6.  | Christophers | fancy  | 0      | 1.405185  | -6.804211 | -.4025908 |
| 7.  | MadCows      | fancy  | 0      | 1.405185  | -5.155514 | -.4025908 |
|     |              |        |        |           |           |           |
| 8.  | Freebirds    | fast   | 0      | -1.804794 | -2.552233 | -1.564074 |
| 9.  | MamasPizza   | fast   | 0      | -1.804794 | -1.680583 | -1.564074 |
| 10. | CafeEccell   | family | 0      | 0         | -2.400434 | .0237072  |
| 11. | LosNortenos  | family | 1      | 0         | -2.223939 | .0237072  |
| 12. | WingsNmore   | family | 0      | 0         | -3.694409 | .0237072  |
| 13. | Christophers | fancy  | 0      | 1.490775  | -5.35932  | -.3159956 |
| 14. | MadCows      | fancy  | 0      | 1.490775  | -5.915751 | -.3159956 |

◀

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [nlogit](#) — Nested logit regression

[U] [20 Estimation and postestimation commands](#)

## Syntax

*Interactive version*

```
nlsur (depvar_1 = <sexp_1>) (depvar_2 = <sexp_2>) ... [if] [in] [weight] [, options]
```

*Programmed substitutable expression version*

```
nlsur sexp_prog : depvar_1 depvar_2 ... [varlist] [if] [in] [weight] [, options]
```

*Function evaluator program version*

```
nlsur func_prog @ depvar_1 depvar_2 ... [varlist] [if] [in] [weight] ,
 nequations(#) { parameters(namelist) | nparameters(#) } [options]
```

where

*depvar*\_*j* is the dependent variable for equation *j*;

<*sexp*>\_*j* is the substitutable expression for equation *j*;

*sexp\_prog* is a substitutable expression program; and

*func\_prog* is a function evaluator program.

| <i>options</i>                              | description                                                                                                                                              |
|---------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                       |                                                                                                                                                          |
| <code>fgnls</code>                          | use two-step FGNLS estimator; the default                                                                                                                |
| <code>ifgnls</code>                         | use iterative FGNLS estimator                                                                                                                            |
| <code>nls</code>                            | use NLS estimator                                                                                                                                        |
| <code>variables(<i>varlist</i>)</code>      | variables in model                                                                                                                                       |
| <code>initial(<i>initial_values</i>)</code> | initial values for parameters                                                                                                                            |
| <code>nequations(#)</code>                  | number of equations in model (function evaluator program version only)                                                                                   |
| <code>*parameters(<i>namelist</i>)</code>   | parameters in model (function evaluator program version only)                                                                                            |
| <code>*nparameters(#)</code>                | number of parameters in model                                                                                                                            |
|                                             | (function evaluator program version only)                                                                                                                |
| <code>sexp_options</code>                   | options for substitutable expression program                                                                                                             |
| <code>func_options</code>                   | options for function evaluator program                                                                                                                   |
| SE/Robust                                   |                                                                                                                                                          |
| <code>vce(<i>vcetype</i>)</code>            | <i>vcetype</i> may be <code>gnr</code> , <code>robust</code> , <code>cluster <i>clustvar</i></code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                                   |                                                                                                                                                          |
| <code>level(#)</code>                       | set confidence level; default is <code>level(95)</code>                                                                                                  |
| <code>title(<i>string</i>)</code>           | display <i>string</i> as title above the table of parameter estimates                                                                                    |
| <code>title2(<i>string</i>)</code>          | display <i>string</i> as subtitle                                                                                                                        |
| Optimization                                |                                                                                                                                                          |
| <code>optimization_options</code>           | control the optimization process; seldom used                                                                                                            |
| <code>eps(#)</code>                         | specify # for convergence criteria; default is <code>eps(1e-5)</code>                                                                                    |
| <code>ifgnlsiterate(#)</code>               | set maximum number of FGNLS iterations                                                                                                                   |
| <code>ifgnlseps(#)</code>                   | specify # for FGNLS convergence criterion; default is <code>ifgnlseps(1e-10)</code>                                                                      |
| <code>delta(#)</code>                       | specify stepsize # for computing derivatives; default is <code>delta(4e-7)</code>                                                                        |
| <code>noconstants</code>                    | no equations have constant terms                                                                                                                         |
| <code>hasconstants(<i>namelist</i>)</code>  | use <i>namelist</i> as constant terms                                                                                                                    |

\* You must specify `parameters(namelist)`, `nparameters(#)`, or both.

`bootstrap`, `by`, `jackknife`, `rolling`, and `statsby` are allowed; see [U] 11.1.10 Prefix commands.

Weights are not allowed with the `bootstrap` prefix.

`aweight`s are not allowed with the `jackknife` prefix.

`fweight`s, `aweight`s, `pweight`s, and `iweight`s are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Linear models and related > Multiple-equation models > Nonlinear seemingly unrelated regression

Description

`nlsur` fits a system of nonlinear equations by feasible generalized nonlinear least squares (FGNLS). With the interactive version of the command, you enter the system of equations on the command line or in the dialog box, using substitutable expressions. If you have a system that you fit regularly, you can write a substitutable expression program and use the second syntax to avoid having to reenter the

system every time. The function evaluator program version gives you the most flexibility in exchange for increased complexity; with this version, your program is given a vector of parameters and a variable list, and your program computes the system of equations.

When you write a substitutable expression program or a function evaluator program, the first five letters of the name must be `nlsur`. *sexp\_prog* and *func\_prog* refer to the name of the program without the first five letters. For example, if you wrote a function evaluator program named `nlsurregss`, you would type `nlsur regss @ ...` to estimate the parameters.

## Options

### Model

`fgnls` requests the two-step FGNLS estimator. This is the default.

`ifgnls` requests the iterative FGNLS estimator. For the nonlinear systems estimator, this is equivalent to maximum likelihood estimation.

`nls` requests the nonlinear least-squares (NLS) estimator.

`variables(varlist)` specifies the variables in the system. `nlsur` ignores observations for which any of these variables has missing values. If you do not specify `variables()`, `nlsur` issues an error if the estimation sample contains any missing values.

`initial(initial_values)` specifies the initial values to begin the estimation. You can specify a  $1 \times k$  matrix, where  $k$  is the total number of parameters in the system, or you can specify a parameter name, its initial value, another parameter name, its initial value, and so on. For example, to initialize `alpha` to 1.23 and `delta` to 4.57, you would type

```
. nlsur ..., initial(alpha 1.23 delta 4.57) ...
```

Initial values declared using this option override any that are declared within substitutable expressions. If you specify a matrix, the values must be in the same order in which the parameters are declared in your model. `nlsur` ignores the row and column names of the matrix.

`nequations(#)` specifies the number of equations in the system.

`parameters(namelist)` specifies the names of the parameters in the system. The names of the parameters must adhere to the naming conventions of Stata's variables; see [U] [11.3 Naming conventions](#). If you specify both `parameters()` and `nparameters()`, the number of names in the former must match the number specified in the latter.

`nparameters(#)` specifies the number of parameters in the system. If you do not specify names with the `parameters()` options, `nlsur` names them `b1`, `b2`, ..., `b#`. If you specify both `parameters()` and `nparameters()`, the number of names in the former must match the number specified in the latter.

*sexp\_options* refer to any options allowed by your *sexp\_prog*.

*func\_options* refer to any options allowed by your *func\_prog*.

## SE/Robust

`vce(vctype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

`vce(gnr)`, the default, uses the conventionally derived variance estimator for nonlinear models fit using Gauss–Newton regression.

## Reporting

`level(#)`; see [R] [estimation options](#).

`title(string)` specifies an optional title that will be displayed just above the table of parameter estimates.

`title2(string)` specifies an optional subtitle that will be displayed between the title specified in `title()` and the table of parameter estimates. If `title2()` is specified but `title()` is not, `title2()` has the same effect as `title()`.

## Optimization

`optimization_options: iterate(#)`, `[no]log`, `trace`. `iterate()` specifies the maximum number of iterations to use for NLS at each round of FGNLS estimation. This option is different from `ifgnlsiterate()`, which controls the maximum rounds of FGNLS estimation to use when the `ifgnls` option is specified. `log/nolog` specifies whether to show the iteration log, and `trace` specifies that the iteration log should include the current parameter vector.

`eps(#)` specifies the convergence criterion for successive parameter estimates and for the residual sum of squares (RSS). The default is `eps(1e-5)` (0.00001). `eps()` also specifies the convergence criterion for successive parameter estimates between rounds of iterative FGNLS estimation when `ifgnls` is specified.

`ifgnlsiterate(#)` specifies the maximum number of FGNLS iterations to perform. The default is the number set using `set maxiter`, which is 16,000 by default. To use this option, you must also specify the `ifgnls` option.

`ifgnlseps(#)` specifies the convergence criterion for successive estimates of the error covariance matrix during iterative FGNLS estimation. The default is `ifgnlseps(1e-10)`. To use this option, you must also specify the `ifgnls` option.

`delta(#)` specifies the relative change in a parameter,  $\delta$ , to be used in computing the numeric derivatives. The derivative for parameter  $\beta_i$  is computed as

$$\{f_i(\mathbf{x}_i, \beta_1, \beta_2, \dots, \beta_i + d, \beta_{i+1}, \dots) - f_i(\mathbf{x}_i, \beta_1, \beta_2, \dots, \beta_i, \beta_{i+1}, \dots)\} / d$$

where  $d = \delta(|\beta_i| + \delta)$ . The default is `delta(4e-7)`.

`noconstants` indicates that none of the equations in the system includes constant terms. This option is generally not needed, even if there are no constant terms in the system; though in rare cases without this option, `nlstur` may claim that there is one or more constant terms even if there are none.

`hasconstants(namelist)` indicates the parameters that are to be treated as constant terms in the system of equations. The number of elements of `namelist` must equal the number of equations in the system. The  $i$ th entry of `namelist` specifies the constant term in the  $i$ th equation. If an equation does not include a constant term, specify a period (.) instead of a parameter name. This option is seldom needed with the interactive and programmed substitutable expression versions, because in those cases `nlstur` can almost always find the constant terms automatically.



## Remarks

Remarks are presented under the following headings:

*Introduction*

*Substitutable expression programs*

*Function evaluator programs*

## Introduction

`nlstur` fits a system of nonlinear equations by FGNLS. It can be viewed as a nonlinear variant of Zellner's seemingly unrelated regression model (Zellner 1962; Zellner and Huang 1962; Zellner 1963) and is therefore commonly called nonlinear SUR or nonlinear SURE. The model is also discussed in textbooks such as Davidson and MacKinnon (1993, 2004) and Greene (2008, 300–302). Formally, the model fit by `nlstur` is

$$\begin{aligned} y_{i1} &= f_1(\mathbf{x}_i, \beta) + u_{i1} \\ y_{i2} &= f_2(\mathbf{x}_i, \beta) + u_{i2} \\ &\vdots \\ y_{iM} &= f_M(\mathbf{x}_i, \beta) + u_{iM} \end{aligned}$$

for  $i = 1, \dots, N$  observations and  $m = 1, \dots, M$  equations. The errors for the  $i$ th observation,  $u_{i1}, u_{i2}, \dots, u_{iM}$ , may be correlated, so fitting the  $m$  equations jointly may lead to more efficient estimates. Moreover, fitting the equations jointly allows us to impose cross-equation restrictions on the parameters. Not all elements of the parameter vector  $\beta$  and data vector  $\mathbf{x}_i$  must appear in all the equations, though each element of  $\beta$  must appear in at least one equation for  $\beta$  to be identified. For this model, iterative FGNLS estimation is equivalent to maximum likelihood estimation with multivariate normal disturbances.

The syntax you use with `nlstur` closely mirrors that used with `nl`. In particular, you use substitutable expressions with the interactive and programmed substitutable expression versions to define the functions in your system. See [R] [nl](#) for more information on substitutable expressions. Here we reiterate the three rules that you must follow:

1. Parameters of the model are bound in braces: `{b0}`, `{param}`, etc.
2. Initial values for parameters are given by including an equal sign and the initial value inside the braces: `{b0=1}`, `{param=3.571}`, etc. If you do not specify an initial value, that parameter is initialized to zero. The `initial()` option overrides initial values in substitutable expressions.
3. Linear combinations of variables can be included using the notation `{eqname:varlist}`, for example, `{xb: mpg price weight}`, `{score: w x z}`, etc. Parameters of linear combinations are initialized to zero.

### ► Example 1: Interactive version using two-step FGNLS estimator

We have data from an experiment in which two closely-related types of bacteria was placed in a Petri dish, and the number of each type of bacteria were recorded every hour. We suspect a two-parameter exponential growth model can be used to model each type of bacteria, but because they shared the same dish, we want to allow for correlation in the error terms. We want to fit the system of equations

$$p_1 = \beta_1 \beta_2^t + u_1$$
$$p_2 = \gamma_1 \gamma_2^t + u_2$$

where  $p_1$  and  $p_2$  are the two populations and  $t$  is time, and we want to allow for nonzero correlation between  $u_1$  and  $u_2$ . We type

```
. use http://www.stata-press.com/data/r11/petridish
. nlsur (p1 = {b1}*{b2}^t) (p2 = {g1}*{g2}^t)
(obs = 25)
```

```
Calculating NLS estimates...
Iteration 0: Residual SS = 335.5286
Iteration 1: Residual SS = 333.8583
Iteration 2: Residual SS = 219.9233
Iteration 3: Residual SS = 127.9355
Iteration 4: Residual SS = 14.86765
Iteration 5: Residual SS = 8.628459
Iteration 6: Residual SS = 8.281268
Iteration 7: Residual SS = 8.28098
Iteration 8: Residual SS = 8.280979
Iteration 9: Residual SS = 8.280979
Calculating FGNLS estimates...
Iteration 0: Scaled RSS = 49.99892
Iteration 1: Scaled RSS = 49.99892
Iteration 2: Scaled RSS = 49.99892
```

FGNLS regression

| Equation |    | Obs | Parms | RMSE     | R-sq    | Constant |
|----------|----|-----|-------|----------|---------|----------|
| 1        | p1 | 25  | 2     | .4337019 | 0.9734* | (none)   |
| 2        | p2 | 25  | 2     | .3783479 | 0.9776* | (none)   |

\* Uncentered R-sq

|     | Coef.    | Std. Err. | z      | P> z  | [95% Conf. Interval] |          |
|-----|----------|-----------|--------|-------|----------------------|----------|
| /b1 | .3926631 | .064203   | 6.12   | 0.000 | .2668275             | .5184987 |
| /b2 | 1.119593 | .0088999  | 125.80 | 0.000 | 1.102149             | 1.137036 |
| /g1 | .5090441 | .0669495  | 7.60   | 0.000 | .3778256             | .6402626 |
| /g2 | 1.102315 | .0072183  | 152.71 | 0.000 | 1.088167             | 1.116463 |

The header of the output contains a summary of each equation, including the number of observations and parameters and the root mean squared error of the residuals. `nlsur` checks to see whether each equation contains a constant term, and if an equation does contain a constant term, an  $R^2$  statistic is presented. If an equation does not have a constant term, an uncentered  $R^2$  is instead reported. The  $R^2$  statistic for each equation measures the percentage of variance explained by the nonlinear function and may be useful for descriptive purposes, though it does not have the same formal interpretation in the context of FGNLS as it does with NLS estimation. As we would expect,  $\beta_2$  and  $\gamma_2$  are both greater than one, indicating the two bacterial populations increased in size over time.



The model we fit in the next three examples is in fact linear in the parameters, so it could be fit using the `sureg` command. However, we will fit the model using `nlsur` so that we can focus on the mechanics of using the command. Moreover, using `nlsur` will obviate the need to generate several variables as well as the need to use the `constraint` command to impose parameter restrictions.

► Example 2: Interactive version using iterative FGNLS estimator—the translog production function

Greene (1997, sec. 15.6) discusses the transcendental logarithmic (translog) cost function and provides cost and input price data for capital, labor, energy, and materials for the U.S. economy. One way to fit the translog production function to these data is to fit the system of three equations

$$\begin{aligned}s_k &= \beta_k + \delta_{kk} \ln \left( \frac{p_k}{p_m} \right) + \delta_{kl} \ln \left( \frac{p_l}{p_m} \right) + \delta_{ke} \ln \left( \frac{p_e}{p_m} \right) + u_1 \\s_l &= \beta_l + \delta_{kl} \ln \left( \frac{p_k}{p_m} \right) + \delta_{ll} \ln \left( \frac{p_l}{p_m} \right) + \delta_{le} \ln \left( \frac{p_e}{p_m} \right) + u_2 \\s_e &= \beta_e + \delta_{ke} \ln \left( \frac{p_k}{p_m} \right) + \delta_{le} \ln \left( \frac{p_l}{p_m} \right) + \delta_{ee} \ln \left( \frac{p_e}{p_m} \right) + u_3\end{aligned}$$

where  $s_k$  is capital's cost share,  $s_l$  is labor's cost share, and  $s_e$  is energy's cost share;  $p_k$ ,  $p_l$ ,  $p_e$ , and  $p_m$  are the prices of capital, labor, energy, and materials, respectively; the  $u$ 's are regression error terms; and the  $\beta$ s and  $\delta$ s are parameters to be estimated. There are three cross-equation restrictions on the parameters:  $\delta_{kl}$ ,  $\delta_{ke}$ , and  $\delta_{le}$  each appear in two equations. To fit this model by using the iterative FGNLS estimator, we type

(Continued on next page)

```
. use http://www.stata-press.com/data/r11/mfgcost
. nlsur (s_k = {bk} + {dkk}*ln(pk/pm) + {dkl}*ln(pl/pm) + {dke}*ln(pe/pm))
> (s_l = {bl} + {dkl}*ln(pk/pm) + {dll}*ln(pl/pm) + {dle}*ln(pe/pm))
> (s_e = {be} + {dke}*ln(pk/pm) + {dle}*ln(pl/pm) + {dee}*ln(pe/pm)),
> ifgnls
(obs = 25)
Calculating NLS estimates...
Iteration 0: Residual SS = .0009989
Iteration 1: Residual SS = .0009989
Calculating FGNLS estimates...
Iteration 0: Scaled RSS = 65.45197
Iteration 1: Scaled RSS = 65.45197
(output omitted)
FGNLS iteration 10...
Iteration 0: Scaled RSS = 75
Iteration 1: Scaled RSS = 75
Iteration 2: Scaled RSS = 75
Parameter change = 4.076e-06
Covariance matrix change = 6.264e-10
FGNLS regression
```

| Equation |     | Obs | Parms | RMSE     | R-sq   | Constant |
|----------|-----|-----|-------|----------|--------|----------|
| 1        | s_k | 25  | 4     | .0031722 | 0.4776 | bk       |
| 2        | s_l | 25  | 4     | .0053963 | 0.8171 | bl       |
| 3        | s_e | 25  | 4     | .00177   | 0.6615 | be       |

|      | Coef.     | Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|------|-----------|-----------|--------|-------|----------------------|-----------|
| /bk  | .0568925  | .0013454  | 42.29  | 0.000 | .0542556             | .0595294  |
| /dkk | .0294833  | .0057956  | 5.09   | 0.000 | .0181241             | .0408425  |
| /dkl | -.0000471 | .0038478  | -0.01  | 0.990 | -.0075887            | .0074945  |
| /dke | -.0106749 | .0033882  | -3.15  | 0.002 | -.0173157            | -.0040341 |
| /bl  | .253438   | .0020945  | 121.00 | 0.000 | .2493329             | .2575432  |
| /dll | .0754327  | .0067572  | 11.16  | 0.000 | .0621889             | .0886766  |
| /dle | -.004756  | .002344   | -2.03  | 0.042 | -.0093502            | -.0001619 |
| /be  | .0444099  | .0008533  | 52.04  | 0.000 | .0427374             | .0460823  |
| /dee | .0183415  | .0049858  | 3.68   | 0.000 | .0085694             | .0281135  |

We draw your attention to the iteration log at the top of the output. When iterative FGNLS estimation is used, the final scaled RSS will equal the product of the number of observations in the estimation sample and the number of equations; see [Methods and formulas](#) for details. Because the RSS is scaled by the error covariance matrix during each round of FGNLS estimation, the scaled RSS is not comparable from one FGNLS iteration to the next.

❏ Technical note

You may have noticed that we mentioned having data for four factors of production, yet we fit only three share equations. Because the four shares sum to one, we must drop one of the equations to avoid having a singular error covariance matrix. The iterative FGNLS estimator is equivalent to maximum likelihood estimation, and thus it is invariant to which one of the four equations we choose to drop. The (linearly restricted) parameters of the fourth equation can be obtained using the `lincom` command. Nonlinear functions of the parameters, such as the elasticities of substitution, can be computed using `nlcom`.



## Substitutable expression programs

If you fit the same model repeatedly or you want to share code with colleagues, you can write a *substitutable expression program* to define your system of equations and avoid having to retype the system every time. The first five letters of the program's name must be `nlsur`, and the program must set the `r-class` macro `r(n_eq)` to the number of equations in your system. The first equation's substitutable expression must be returned in `r(eq_1)`, the second equation's in `r(eq_2)`, and so on. You may optionally set `r(title)` to label your output; that has the same effect as specifying the `title()` option.

### ► Example 3: Programmed substitutable expression version

We return to our translog cost function, for which a substitutable expression program is

```
program nlsurtranslog, rclass
 version 11
 syntax varlist(min=7 max=7) [if]
 tokenize 'varlist'
 args sk sl se pk pl pe pm
 local pkpm ln('pk'/'pm')
 local plpm ln('pl'/'pm')
 local pepm ln('pe'/'pm')
 return scalar n_eq = 3
 return local eq_1 "'sk' = {bk} + {dkk}*'pkpm' + {dkl}*'plpm' + {dke}*'pepm'"
 return local eq_2 "'sl' = {bl} + {dkl}*'pkpm' + {dll}*'plpm' + {dle}*'pepm'"
 return local eq_3 "'se' = {be} + {dke}*'pkpm' + {dle}*'plpm' + {dee}*'pepm'"
 return local title "4-factor translog cost function"
end
```

We made our program accept seven variables, for the three dependent variables  $s_k$ ,  $s_l$ , and  $s_e$ , and the four factor prices  $p_k$ ,  $p_l$ ,  $p_m$ , and  $p_e$ . The `tokenize` command assigns to macros '1', '2', ..., '7' the seven variables stored in 'varlist', and the `args` command transfers those numbered macros to macros 'sk', 'sl', ..., 'pm'. Because we knew our substitutable expressions were going to be somewhat long, we created local macros to hold the log price ratios. These are simply macros that hold strings such as `ln(pk/pm)`, not variables, and they will save us some repetitious typing when we define our substitutable expressions. Our program returns the number of equations in `r(n_eq)`, and we defined our substitutable expressions in `eq_1`, `eq_2`, and `eq_3`. We do not bind the expressions in parentheses as we do with the interactive version of `nlsur`. Finally, we put a title in `r(title)` to label our output.

Our `syntax` command also accepts an `if` clause, and that is how `nlsur` indicates the estimation sample to our program. In this application, we can safely ignore it, because our program does not compute initial values. However, had we used commands such as `summarize` or `regress` to obtain initial values, then we would need to restrict those commands to analyze only the estimation sample. In those cases, typically, you simply need to include 'if' with the commands you are using. For example, instead of the command

```
summarize 'depvar', meanonly
```

you would use

```
summarize 'depvar' 'if', meanonly
```

We can check our program by typing

```
. nlsurtranslog s_k s_l s_e pk pl pe pm
. return list
scalars:
 r(n_eq) = 3
macros:
 r(title) : "4-factor translog cost function"
 r(eq_3) : "s_e= {be} + {dke}*ln(pk/pm) + {dle}*ln(pl/pm) + {dee}."
 r(eq_2) : "s_l= {bl} + {dkl}*ln(pk/pm) + {dll}*ln(pl/pm) + {dle}."
 r(eq_1) : "s_k= {bk} + {dkk}*ln(pk/pm) + {dkl}*ln(pl/pm) + {dke}."
```

Now that we know that our program works, we fit our model by typing

```
. nlsur translog: s_k s_l s_e pk pl pe pm, ifgnls
(obs = 25)
```

```
Calculating NLS estimates...
Iteration 0: Residual SS = .0009989
Iteration 1: Residual SS = .0009989
Calculating FGNLS estimates...
Iteration 0: Scaled RSS = 65.45197
Iteration 1: Scaled RSS = 65.45197
FGNLS iteration 2...
Iteration 0: Scaled RSS = 73.28311
Iteration 1: Scaled RSS = 73.28311
Parameter change = 6.537e-03
Covariance matrix change = 1.002e-06
(output omitted)
FGNLS iteration 10...
Iteration 0: Scaled RSS = 75
Iteration 1: Scaled RSS = 75
Iteration 2: Scaled RSS = 75
Parameter change = 4.076e-06
Covariance matrix change = 6.264e-10
FGNLS regression
```

| Equation |     | Obs | Parms | RMSE     | R-sq   | Constant |
|----------|-----|-----|-------|----------|--------|----------|
| 1        | s_k | 25  | 4     | .0031722 | 0.4776 | bk       |
| 2        | s_l | 25  | 4     | .0053963 | 0.8171 | bl       |
| 3        | s_e | 25  | 4     | .00177   | 0.6615 | be       |

4-factor translog cost function

|      | Coef.     | Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|------|-----------|-----------|--------|-------|----------------------|-----------|
| /bk  | .0568925  | .0013454  | 42.29  | 0.000 | .0542556             | .0595294  |
| /dkk | .0294833  | .0057956  | 5.09   | 0.000 | .0181241             | .0408425  |
| /dkl | -.0000471 | .0038478  | -0.01  | 0.990 | -.0075887            | .0074945  |
| /dke | -.0106749 | .0033882  | -3.15  | 0.002 | -.0173157            | -.0040341 |
| /bl  | .253438   | .0020945  | 121.00 | 0.000 | .2493329             | .2575432  |
| /dll | .0754327  | .0067572  | 11.16  | 0.000 | .0621889             | .0886766  |
| /dle | -.004756  | .002344   | -2.03  | 0.042 | -.0093502            | -.0001619 |
| /be  | .0444099  | .0008533  | 52.04  | 0.000 | .0427374             | .0460823  |
| /dee | .0183415  | .0049858  | 3.68   | 0.000 | .0085694             | .0281135  |

Because we set `r(title)` in our substitutable expression program, the coefficient table has a title attached to it. The estimates are identical to those we obtained in example 2.

## □ Technical note

`nlsur` accepts frequency and analytic weights as well as `pweights` (sampling weights) and `iweights` (importance weights). You do not need to modify your substitutable expressions in any way to perform weighted estimation, though you must make two changes to your substitutable expression program. The general outline of a *sexp\_prog* program is

```
program nlsur name, rclass
 version 11
 syntax varlist [fw aw pw iw] [if]
 // Obtain initial values incorporating weights. For example,
 summarize varname ['weight' 'exp'] 'if'
 ...
 // Return n_eqn and substitutable expressions
 return scalar n_eq = #
 return local eq_1 = ...
 ...
end
```

First, we wrote the `syntax` statement to accept a weight expression. Here we allow all four types of weights, but if you know that your estimator is valid, say, for only frequency weights, then you should modify the `syntax` line to accept only `fweights`. Second, if your program computes starting values, then any commands you use must incorporate the weights passed to the program; you do that by including `['weight' 'exp']` when calling those commands.

□

## Function evaluator programs

Although substitutable expressions are extremely flexible, there are some problems for which the nonlinear system cannot be defined using them. You can use the function evaluator program version of `nlsur` in these cases. We present two examples, a simple one to illustrate the mechanics of function evaluator programs and a more complicated one to illustrate the power of `nlsur`.

(Continued on next page)

### ► Example 4: Function evaluator program version

Here we write a function evaluator program to fit the translog cost function used in examples 2 and 3. The function evaluator program is

```
program nlsurtranslog2
 version 11
 syntax varlist(min=7 max=7) [if], at(name)
 tokenize 'varlist'
 args sk sl se pk pl pe pm
 tempname bk dkk dkl dke bl dll dle be dee
 scalar 'bk' = 'at'[1,1]
 scalar 'dkk' = 'at'[1,2]
 scalar 'dkl' = 'at'[1,3]
 scalar 'dke' = 'at'[1,4]
 scalar 'bl' = 'at'[1,5]
 scalar 'dll' = 'at'[1,6]
 scalar 'dle' = 'at'[1,7]
 scalar 'be' = 'at'[1,8]
 scalar 'dee' = 'at'[1,9]
 local pkpm ln('pk'/'pm')
 local plpm ln('pl'/'pm')
 local pepm ln('pe'/'pm')
 quietly {
 replace 'sk' = 'bk' + 'dkk'*'pkpm' + 'dkl'*'plpm' + ///
 'dke'*'pepm' 'if'
 replace 'sl' = 'bl' + 'dkl'*'pkpm' + 'dll'*'plpm' + ///
 'dle'*'pepm' 'if'
 replace 'se' = 'be' + 'dke'*'pkpm' + 'dle'*'plpm' + ///
 'dee'*'pepm' 'if'
 }
end
```

Unlike the substitutable expression program we wrote in [example 3](#), `nlsurtranslog2` is not declared as `r-class` because we will not be returning any saved results. We are again expecting seven variables: three shares and four factor prices, and `nlсур` will again mark the estimation sample with an `if` expression.

Our function evaluator program also accepts an option named `at()`, which will receive a parameter vector at which we are to evaluate the system of equations. All function evaluator programs must accept this option. Our model has nine parameters to estimate, and we created nine temporary scalars to hold the elements of the `'at'` matrix.

Because our model has three equations, the first three variables passed to our program are the dependent variables that we are to fill in with the function values. We replaced only the observations in our estimation sample by including the `'if'` qualifier in the `replace` statements. Here we could have ignored the `'if'` qualifier because `nlсур` will skip over observations not in the estimation sample and we did not perform any computations requiring knowledge of the estimation sample. However, including the `'if'` is good practice and may result in a slight speed improvement if the functions of your model are complicated and the estimation sample is much smaller than the dataset in memory.

We could have avoided creating temporary scalars to hold our individual parameters by writing the `replace` statements as, for example,

```
replace 'sk' = 'at'[1,1] + 'at'[1,2]*'pkpm' + 'at'[1,3]*'plpm' + 'at'[1,4]*'pepm' 'if'
```

You can use whichever method you find more appealing, though giving the parameters descriptive names reduces the chance for mistakes and makes debugging easier.



To fit our model by using the function evaluator program version of nlsur, we type

```
. nlsur translog2 @ s_k s_l s_e pk pl pe pm, ifgnls nequations(3)
> parameters(bk dkk dkl dke bl dll dle be dee)
> hasconstants(bk bl be)
(obs = 25)

Calculating NLS estimates...
Iteration 0: Residual SS = .0009989
Iteration 1: Residual SS = .0009989
Calculating FGNLS estimates...
Iteration 0: Scaled RSS = 65.45197
Iteration 1: Scaled RSS = 65.45197
FGNLS iteration 2...
Iteration 0: Scaled RSS = 73.28311
Iteration 1: Scaled RSS = 73.28311
Parameter change = 6.537e-03
Covariance matrix change = 1.002e-06
FGNLS iteration 3...
Iteration 0: Scaled RSS = 74.7113
Iteration 1: Scaled RSS = 74.7113
Parameter change = 2.577e-03
Covariance matrix change = 3.956e-07
FGNLS iteration 4...
Iteration 0: Scaled RSS = 74.95356
Iteration 1: Scaled RSS = 74.95356
Parameter change = 1.023e-03
Covariance matrix change = 1.571e-07
FGNLS iteration 5...
Iteration 0: Scaled RSS = 74.99261
Iteration 1: Scaled RSS = 74.99261
Iteration 2: Scaled RSS = 74.99261
Iteration 3: Scaled RSS = 74.99261
Parameter change = 4.067e-04
Covariance matrix change = 6.250e-08
FGNLS iteration 6...
Iteration 0: Scaled RSS = 74.99883
Iteration 1: Scaled RSS = 74.99883
Parameter change = 1.619e-04
Covariance matrix change = 2.489e-08
FGNLS iteration 7...
Iteration 0: Scaled RSS = 74.99981
Iteration 1: Scaled RSS = 74.99981
Parameter change = 6.449e-05
Covariance matrix change = 9.912e-09
FGNLS iteration 8...
Iteration 0: Scaled RSS = 74.99997
Iteration 1: Scaled RSS = 74.99997
Iteration 2: Scaled RSS = 74.99997
Parameter change = 2.569e-05
Covariance matrix change = 3.948e-09
FGNLS iteration 9...
Iteration 0: Scaled RSS = 75
Iteration 1: Scaled RSS = 75
Iteration 2: Scaled RSS = 75
Parameter change = 1.023e-05
Covariance matrix change = 1.573e-09
FGNLS iteration 10...
Iteration 0: Scaled RSS = 75
Iteration 1: Scaled RSS = 75
Iteration 2: Scaled RSS = 75
Parameter change = 4.076e-06
Covariance matrix change = 6.264e-10
```

FGNLS regression

| Equation |     | Obs | Parms | RMSE     | R-sq   | Constant |
|----------|-----|-----|-------|----------|--------|----------|
| 1        | s_k | 25  | .     | .0031722 | 0.4776 | bk       |
| 2        | s_l | 25  | .     | .0053963 | 0.8171 | bl       |
| 3        | s_e | 25  | .     | .00177   | 0.6615 | be       |

|      | Coef.     | Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|------|-----------|-----------|--------|-------|----------------------|-----------|
| /bk  | .0568925  | .0013454  | 42.29  | 0.000 | .0542556             | .0595294  |
| /dkk | .0294833  | .0057956  | 5.09   | 0.000 | .0181241             | .0408425  |
| /dkl | -.0000471 | .0038478  | -0.01  | 0.990 | -.0075887            | .0074945  |
| /dke | -.0106749 | .0033882  | -3.15  | 0.002 | -.0173157            | -.0040341 |
| /bl  | .253438   | .0020945  | 121.00 | 0.000 | .2493329             | .2575432  |
| /dll | .0754327  | .0067572  | 11.16  | 0.000 | .0621889             | .0886766  |
| /dle | -.004756  | .002344   | -2.03  | 0.042 | -.0093502            | -.0001619 |
| /be  | .0444099  | .0008533  | 52.04  | 0.000 | .0427374             | .0460823  |
| /dee | .0183415  | .0049858  | 3.68   | 0.000 | .0085694             | .0281135  |

When we use the function evaluator program version, `nlsur` requires us to specify the number of equations in `nequations()`, and it requires us to either specify names for each of our parameters or the number of parameters in the model. Here we used the `parameters()` option to name our parameters; the order in which we specified them in this option is the same as the order in which we extracted them from the ‘`at`’ matrix in our program. Had we instead specified `nparameters(9)`, our parameters would have been labeled `/b1`, `/b2`, ..., `/b9` in the output.

`nlsur` has no way of telling how many parameters appear in each equation, so the `Parms` column in the header contains missing values. Moreover, the function evaluator program version of `nlsur` does not attempt to identify constant terms, so we used the `hasconstant` option to tell `nlsur` which parameter in each equation is a constant term.

The estimates are identical to those we obtained in examples 2 and 3.

□ Technical note

As with substitutable expression programs, if you intend to do weighted estimation with a function evaluator program, you must modify your *func\_prog* program’s `syntax` statement to accept weights. Moreover, if you use any statistical commands when computing your nonlinear functions, then you must include the weight expression with those commands.

➤ Example 5: Fitting the basic AIDS model using `nlsur`

Poi (2002) showed how to fit a quadratic almost ideal demand system (AIDS) by using the `m1` command. Here we show how to fit the basic AIDS model by using `nlsur`. Poi (2008) shows how to fit the quadratic AIDS model using `nlsur`. The dataset `food.dta` contains household expenditures, expenditure shares, and log prices for four broad food groups. For a four-good demand system, we need to fit the following system of three equations:

$$w_1 = \alpha_1 + \gamma_{11} \ln p_1 + \gamma_{12} \ln p_2 + \gamma_{13} \ln p_3 + \beta_1 \ln \left\{ \frac{m}{P(\mathbf{p})} \right\} + u_1$$

$$w_2 = \alpha_2 + \gamma_{12} \ln p_1 + \gamma_{22} \ln p_2 + \gamma_{23} \ln p_3 + \beta_2 \ln \left\{ \frac{m}{P(\mathbf{p})} \right\} + u_2$$

$$w_3 = \alpha_3 + \gamma_{13} \ln p_1 + \gamma_{23} \ln p_2 + \gamma_{33} \ln p_3 + \beta_3 \ln \left\{ \frac{m}{P(\mathbf{p})} \right\} + u_3$$

where  $w_k$  denotes a household's fraction of expenditures on good  $k$ ,  $\ln p_k$  denotes the logarithm of the price paid for good  $k$ ,  $m$  denotes a household's total expenditure on all four goods, the  $u$ 's are regression error terms, and

$$\ln P(\mathbf{p}) = \alpha_0 + \sum_{i=1}^4 \alpha_i \ln p_i + \frac{1}{2} \sum_{i=1}^4 \sum_{j=1}^4 \gamma_{ij} \ln p_i \ln p_j$$

The parameters for the fourth good's share equation can be recovered from the following constraints that are imposed by economic theory:

$$\sum_{i=1}^4 \alpha_i = 1 \quad \sum_{i=1}^4 \beta_i = 0 \quad \gamma_{ij} = \gamma_{ji} \quad \text{and} \quad \sum_{i=1}^4 \gamma_{ij} = 0 \quad \text{for all } j$$

Our model has a total of 12 unrestricted parameters. We will not estimate  $\alpha_0$  directly. Instead, we will set it equal to 5 as was done in [Poi \(2002\)](#); see [Deaton and Muellbauer \(1980\)](#) for a discussion of why treating  $\alpha_0$  as fixed is acceptable.

*(Continued on next page)*

Our function evaluator program is

```

program nlsuraids
 version 11
 syntax varlist(min=8 max=8) if, at(name)
 tokenize 'varlist'
 args w1 w2 w3 lnp1 lnp2 lnp3 lnp4 lnm
 tempname a1 a2 a3 a4
 scalar 'a1' = 'at'[1,1]
 scalar 'a2' = 'at'[1,2]
 scalar 'a3' = 'at'[1,3]
 scalar 'a4' = 1 - 'a1' - 'a2' - 'a3'
 tempname b1 b2 b3
 scalar 'b1' = 'at'[1,4]
 scalar 'b2' = 'at'[1,5]
 scalar 'b3' = 'at'[1,6]
 tempname g11 g12 g13 g14
 tempname g21 g22 g23 g24
 tempname g31 g32 g33 g34
 tempname g41 g42 g43 g44
 scalar 'g11' = 'at'[1,7]
 scalar 'g12' = 'at'[1,8]
 scalar 'g13' = 'at'[1,9]
 scalar 'g14' = -'g11'-'g12'-'g13'
 scalar 'g21' = 'g12'
 scalar 'g22' = 'at'[1,10]
 scalar 'g23' = 'at'[1,11]
 scalar 'g24' = -'g21'-'g22'-'g23'
 scalar 'g31' = 'g13'
 scalar 'g32' = 'g23'
 scalar 'g33' = 'at'[1,12]
 scalar 'g34' = -'g31'-'g32'-'g33'
 scalar 'g41' = 'g14'
 scalar 'g42' = 'g24'
 scalar 'g43' = 'g34'
 scalar 'g44' = -'g41'-'g42'-'g43'
 quietly {
 tempvar lnpindex
 gen double 'lnpindex' = 5 + 'a1'*'lnp1' + 'a2'*'lnp2' + ///
 'a3'*'lnp3' + 'a4'*'lnp4'
 forvalues i = 1/4 {
 forvalues j = 1/4 {
 replace 'lnpindex' = 'lnpindex' + ///
 0.5*'g'i','j'*'lnp'i'*'lnp'j',
 }
 }
 replace 'w1' = 'a1' + 'g11'*'lnp1' + 'g12'*'lnp2' + ///
 'g13'*'lnp3' + 'g14'*'lnp4' + ///
 'b1'*('lnm' - 'lnpindex')
 replace 'w2' = 'a2' + 'g21'*'lnp1' + 'g22'*'lnp2' + ///
 'g23'*'lnp3' + 'g24'*'lnp4' + ///
 'b2'*('lnm' - 'lnpindex')
 replace 'w3' = 'a3' + 'g31'*'lnp1' + 'g32'*'lnp2' + ///
 'g33'*'lnp3' + 'g34'*'lnp4' + ///
 'b3'*('lnm' - 'lnpindex')
 }
end

```

The `syntax` statement accepts eight variables: three expenditure share variables, all four log-price variables, and a variable for log expenditures (`lnm`). Most of the code simply extracts the parameters

from the ‘at’ matrix. Although we are estimating only 12 parameters, to calculate the price index term and the expenditure share equations, we need the restricted parameters as well. Notice how we impose the constraints on the parameters. We then created a temporary variable to hold  $\ln P(\mathbf{p})$ , and we filled the three dependent variables with the predicted expenditure shares.

To fit our model, we type

```
. use http://www.stata-press.com/data/r11/food, clear
. nlsur aids @ w1 w2 w3 lnp1 lnp2 lnp3 lnp4 lnxp,
> parameters(a1 a2 a3 b1 b2 b3
> g11 g12 g13 g22 g32 g33)
> neq(3) ifgnls
(obs = 4048)
```

```
Calculating NLS estimates...
Iteration 0: Residual SS = 126.9713
Iteration 1: Residual SS = 125.669
Iteration 2: Residual SS = 125.669
Iteration 3: Residual SS = 125.669
Iteration 4: Residual SS = 125.669
Calculating FGNLS estimates...
Iteration 0: Scaled RSS = 12080.14
Iteration 1: Scaled RSS = 12080.14
Iteration 2: Scaled RSS = 12080.14
Iteration 3: Scaled RSS = 12080.14
FGNLS iteration 2...
Iteration 0: Scaled RSS = 12143.99
Iteration 1: Scaled RSS = 12143.99
Iteration 2: Scaled RSS = 12143.99
Parameter change = 1.972e-04
Covariance matrix change = 2.936e-06
FGNLS iteration 3...
Iteration 0: Scaled RSS = 12144
Iteration 1: Scaled RSS = 12144
Parameter change = 2.178e-06
Covariance matrix change = 3.467e-08
```

FGNLS regression

| Equation |    | Obs  | Parms | RMSE     | R-sq    | Constant |
|----------|----|------|-------|----------|---------|----------|
| 1        | w1 | 4048 | .     | .1333175 | 0.9017* | (none)   |
| 2        | w2 | 4048 | .     | .1024166 | 0.8480* | (none)   |
| 3        | w3 | 4048 | .     | .053777  | 0.7906* | (none)   |

\* Uncentered R-sq

|      | Coef.     | Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|------|-----------|-----------|--------|-------|----------------------|-----------|
| /a1  | .3163958  | .0073871  | 42.83  | 0.000 | .3019175             | .3308742  |
| /a2  | .2712501  | .0056938  | 47.64  | 0.000 | .2600904             | .2824097  |
| /a3  | .1039898  | .0029004  | 35.85  | 0.000 | .0983051             | .1096746  |
| /b1  | .0161044  | .0034153  | 4.72   | 0.000 | .0094105             | .0227983  |
| /b2  | -.0260771 | .002623   | -9.94  | 0.000 | -.0312181            | -.0209361 |
| /b3  | .0014538  | .0013776  | 1.06   | 0.291 | -.0012463            | .004154   |
| /g11 | .1215838  | .0057186  | 21.26  | 0.000 | .1103756             | .1327921  |
| /g12 | -.0522943 | .0039305  | -13.30 | 0.000 | -.0599979            | -.0445908 |
| /g13 | -.0351292 | .0021788  | -16.12 | 0.000 | -.0393996            | -.0308588 |
| /g22 | .0644298  | .0044587  | 14.45  | 0.000 | .0556909             | .0731687  |
| /g32 | -.0011786 | .0019767  | -0.60  | 0.551 | -.0050528            | .0026957  |
| /g33 | .0424381  | .0017589  | 24.13  | 0.000 | .0389909             | .0458854  |

To get the restricted parameters for the fourth share equation, we can use `lincom`. For example, to obtain  $\alpha_4$ , we type

```
. lincom 1 - [a1]_cons - [a2]_cons - [a3]_cons
(1) - [a1]_cons - [a2]_cons - [a3]_cons = -1
```

|     | Coef.    | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-----|----------|-----------|-------|-------|----------------------|----------|
| (1) | .3083643 | .0052611  | 58.61 | 0.000 | .2980528             | .3186758 |

For more information on `lincom`, see [\[R\] lincom](#).



## Saved results

`nlsur` saves the following in `e()`:

|                            |                                                             |
|----------------------------|-------------------------------------------------------------|
| Scalars                    |                                                             |
| <code>e(N)</code>          | number of observations                                      |
| <code>e(k)</code>          | number of parameters in system                              |
| <code>e(k_#)</code>        | number of parameters for equation #                         |
| <code>e(k_aux)</code>      | number of estimates parameters                              |
| <code>e(k_eq)</code>       | number of equation names in <code>e(b)</code>               |
| <code>e(k_eq_model)</code> | number of equations in model Wald test                      |
| <code>e(n_eq)</code>       | number of equations                                         |
| <code>e(mss_#)</code>      | model sum of squares for equation #                         |
| <code>e(rss_#)</code>      | RSS for equation #                                          |
| <code>e(rmse_#)</code>     | root mean squared error for equation #                      |
| <code>e(r2_#)</code>       | $R^2$ for equation #                                        |
| <code>e(ll)</code>         | Gaussian log likelihood ( <code>iflags</code> version only) |
| <code>e(N_clust)</code>    | number of clusters                                          |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                                   |
| <code>e(converge)</code>   | 1 if converged, 0 otherwise                                 |

(Continued on next page)

# Macros

|                             |                                                                                                                |
|-----------------------------|----------------------------------------------------------------------------------------------------------------|
| <code>e(cmd)</code>         | <code>nlsur</code>                                                                                             |
| <code>e(cmdline)</code>     | command as typed                                                                                               |
| <code>e(method)</code>      | <code>fgnls</code> , <code>ifgnls</code> , or <code>nls</code>                                                 |
| <code>e(depvar)</code>      | names of dependent variables                                                                                   |
| <code>e(depvar_#)</code>    | dependent variable for equation #                                                                              |
| <code>e(wtype)</code>       | weight type                                                                                                    |
| <code>e(wexp)</code>        | weight expression                                                                                              |
| <code>e(title)</code>       | title in estimation output                                                                                     |
| <code>e(title_2)</code>     | secondary title in estimation output                                                                           |
| <code>e(clustvar)</code>    | name of cluster variable                                                                                       |
| <code>e(vce)</code>         | <i>vcetype</i> specified in <code>vce()</code>                                                                 |
| <code>e(vcetype)</code>     | title used in label Std. Err.                                                                                  |
| <code>e(type)</code>        | 1 = interactively entered expression<br>2 = substitutable expression program<br>3 = function evaluator program |
| <code>e(sexpprog)</code>    | substitutable expression program                                                                               |
| <code>e(sexp_#)</code>      | substitutable expression for equation #                                                                        |
| <code>e(params)</code>      | names of all parameters                                                                                        |
| <code>e(params_#)</code>    | parameters in equation #                                                                                       |
| <code>e(funcprog)</code>    | function evaluator program                                                                                     |
| <code>e(rhs)</code>         | contents of <code>variables()</code>                                                                           |
| <code>e(constants)</code>   | identifies constant terms                                                                                      |
| <code>e(properties)</code>  | <code>b V</code>                                                                                               |
| <code>e(predict)</code>     | program used to implement <code>predict</code>                                                                 |
| <code>e(marginsprop)</code> | signals to the <code>margins</code> command                                                                    |

# Matrices

|                       |                                                |
|-----------------------|------------------------------------------------|
| <code>e(b)</code>     | coefficient vector                             |
| <code>e(init)</code>  | initial values vector                          |
| <code>e(Sigma)</code> | error covariance matrix ( $\widehat{\Sigma}$ ) |
| <code>e(V)</code>     | variance–covariance matrix of the estimators   |

# Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`nlsur` is implemented as an ado-file.

Write the system of equations for the  $i$ th observation as

$$\mathbf{y}_i = \mathbf{f}(\mathbf{x}_i, \beta) + \mathbf{u}_i \quad (1)$$

where  $\mathbf{y}_i$  and  $\mathbf{u}_i$  are  $1 \times M$  vectors, for  $i = 1, \dots, N$ ;  $\mathbf{f}$  is a function that returns a  $1 \times M$  vector;  $\mathbf{x}_i$  represents all the exogenous variables in the system; and  $\beta$  is a  $1 \times k$  vector of parameters. The generalized nonlinear least-squares system estimator is defined as

$$\widehat{\beta} \equiv \operatorname{argmin}_{\beta} \sum_{i=1}^N \{\mathbf{y}_i - \mathbf{f}(\mathbf{x}_i, \beta)\} \Sigma^{-1} \{\mathbf{y}_i - \mathbf{f}(\mathbf{x}_i, \beta)\}'$$

where  $\Sigma = E(\mathbf{u}_i' \mathbf{u}_i)$  is an  $M \times M$  positive-definite weight matrix. Let  $\mathbf{T}$  be the Cholesky decomposition of  $\Sigma^{-1}$ ; that is,  $\mathbf{T}'\mathbf{T}' = \Sigma^{-1}$ . Postmultiply (1) by  $\mathbf{T}$ :

$$\mathbf{y}_i \mathbf{T} = \mathbf{f}(\mathbf{x}_i, \beta) \mathbf{T} + \mathbf{u}_i \mathbf{T} \quad (2)$$

Because  $E(\mathbf{T}' \mathbf{u}_i' \mathbf{u}_i \mathbf{T}) = \mathbf{I}$ , we can “stack” the columns of (2) and write

$$\begin{aligned} \mathbf{y}_1 \mathbf{T}_1 &= \mathbf{f}(\mathbf{x}_1, \beta) \mathbf{T}_1 + \tilde{u}_{11} \\ \mathbf{y}_1 \mathbf{T}_2 &= \mathbf{f}(\mathbf{x}_1, \beta) \mathbf{T}_2 + \tilde{u}_{12} \\ &\vdots = \vdots \\ \mathbf{y}_1 \mathbf{T}_M &= \mathbf{f}(\mathbf{x}_1, \beta) \mathbf{T}_M + \tilde{u}_{1M} \\ &\vdots = \vdots \\ \mathbf{y}_N \mathbf{T}_1 &= \mathbf{f}(\mathbf{x}_N, \beta) \mathbf{T}_1 + \tilde{u}_{N1} \\ \mathbf{y}_N \mathbf{T}_2 &= \mathbf{f}(\mathbf{x}_N, \beta) \mathbf{T}_2 + \tilde{u}_{N2} \\ &\vdots = \vdots \\ \mathbf{y}_N \mathbf{T}_M &= \mathbf{f}(\mathbf{x}_N, \beta) \mathbf{T}_M + \tilde{u}_{NM} \end{aligned} \quad (3)$$

where  $\mathbf{T}_j$  denotes the  $j$ th column of  $\mathbf{T}$ . By construction, all  $\tilde{u}_{ij}$  are independently distributed with unit variance. As a result, by transforming the model in (1) to that shown in (3), we have reduced the multivariate generalized nonlinear least-squares system estimator to a univariate nonlinear least-squares problem; and the same parameter estimation technique used by `nl` can be used here. See [R] `nl` for the details. Moreover, because the  $\tilde{u}_{ij}$  all have variance 1, the final scaled RSS reported by `nlsur` is equal to  $NM$ .

To make the estimator feasible, we require an estimate  $\hat{\Sigma}$  of  $\Sigma$ . `nlsur` first sets  $\hat{\Sigma} = \mathbf{I}$ . Although not efficient, the resulting estimate,  $\hat{\beta}_{\text{NLS}}$ , is consistent. If the `nls` option is specified, estimation is complete. Otherwise, the residuals

$$\hat{\mathbf{u}}_i = \mathbf{y}_i - \mathbf{f}(\mathbf{x}_i, \hat{\beta}_{\text{NLS}})$$

are calculated and used to compute

$$\hat{\Sigma} = \frac{1}{N} \sum_{i=1}^N \hat{\mathbf{u}}_i' \hat{\mathbf{u}}_i$$

With  $\hat{\Sigma}$  in hand, a new estimate  $\hat{\beta}$  is then obtained.

If the `ifgnls` option is specified, the new  $\hat{\beta}$  is used to recompute the residuals and obtain a new estimate of  $\hat{\Sigma}$ , from which  $\hat{\beta}$  can then be reestimated. Iterations stop when the relative change in  $\hat{\beta}$  is less than `eps()`, the relative change in  $\hat{\Sigma}$  is less than `ifgnlseps()`, or if `ifgnlsiterate()` iterations have been performed.



If the `vce(robust)` and `vce(cluster clustvar)` options were not specified, then

$$V(\hat{\beta}) = \left( \sum_{i=1}^N \mathbf{X}_i' \hat{\Sigma}^{-1} \mathbf{X}_i \right)^{-1}$$

where the  $M \times k$  matrix  $\mathbf{X}_i$  has typical element  $X_{ist}$ , the derivative of the  $s$ th element of  $\mathbf{f}$  with respect to the  $t$ th element of  $\beta$ , evaluated at  $\mathbf{x}_i$  and  $\hat{\beta}$ . As a practical matter, once the model is written in the form of (3), the variance–covariance matrix can be calculated via a Gauss–Newton regression; see Davidson and MacKinnon (1993, chap. 6).

If `robust` is specified, then

$$V_R(\hat{\beta}) = \left( \sum_{i=1}^N \mathbf{X}_i' \hat{\Sigma}^{-1} \mathbf{X}_i \right)^{-1} \sum_{i=1}^N \mathbf{X}_i' \hat{\Sigma}^{-1} \hat{\mathbf{u}}_i' \hat{\mathbf{u}}_i \hat{\Sigma}^{-1} \mathbf{X}_i \left( \sum_{i=1}^N \mathbf{X}_i' \hat{\Sigma}^{-1} \mathbf{X}_i \right)^{-1}$$

The cluster–robust variance matrix is

$$V_C(\hat{\beta}) = \left( \sum_{i=1}^N \mathbf{X}_i' \hat{\Sigma}^{-1} \mathbf{X}_i \right)^{-1} \sum_{c=1}^{N_C} \mathbf{w}_c' \mathbf{w}_c \left( \sum_{i=1}^N \mathbf{X}_i' \hat{\Sigma}^{-1} \mathbf{X}_i \right)^{-1}$$

where  $N_C$  is the number of clusters and

$$\mathbf{w}_c = \sum_{j \in C_k} \mathbf{X}_j' \hat{\Sigma}^{-1} \hat{\mathbf{u}}_j'$$

with  $C_k$  denoting the set of observations in the  $k$ th cluster. In evaluating these formulas, we use the value of  $\hat{\Sigma}$  used in calculating the final estimate of  $\hat{\beta}$ . That is, we do not recalculate  $\hat{\Sigma}$  after we obtain the final value of  $\hat{\beta}$ .

The RSS for the  $j$ th equation,  $\text{RSS}_j$ , is

$$\text{RSS}_j = \sum_{i=1}^N (\hat{y}_{ij} - y_{ij})^2$$

where  $\hat{y}_{ij}$  is the predicted value of the  $i$ th observation on the  $j$ th dependent variable; the total sum of squares (TSS) for the  $j$ th equation,  $\text{TSS}_j$ , is

$$\text{TSS}_j = \sum_{i=1}^N (y_{ij} - \bar{y}_j)^2$$

if there is a constant term in the  $j$ th equation, where  $\bar{y}_j$  is the sample mean of the  $j$ th dependent variable, and

$$\text{TSS}_j = \sum_{i=1}^N y_{ij}^2$$

if there is no constant term in the  $j$ th equation; and the model sum of squares (MSS) for the  $j$ th equation,  $MSS_j$ , is  $TSS_j - RSS_j$ .

The  $R^2$  for the  $j$ th equation is  $MSS_j/TSS_j$ . If an equation does not have a constant term, then the reported  $R^2$  for that equation is “uncentered” and based on the latter definition of  $TSS_j$ .

Under the assumption that the  $\mathbf{u}_i$  are independent and identically distributed  $N(\mathbf{0}, \hat{\Sigma})$ , the log likelihood for the model is

$$\ln L = -\frac{MN}{2} \{1 + \ln(2\pi)\} - \frac{N}{2} \ln |\hat{\Sigma}|$$

The log likelihood is reported only when the `ifgnls` option is specified.

## References

- Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.
- . 2004. *Econometric Theory and Methods*. New York: Oxford University Press.
- Deaton, A., and J. Muellbauer. 1980. An almost ideal demand system. *American Economic Review* 70: 312–326.
- Greene, W. H. 1997. *Econometric Analysis*. 3rd ed. Upper Saddle River, NJ: Prentice–Hall.
- . 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- Poi, B. P. 2002. From the help desk: Demand system estimation. *Stata Journal* 2: 403–410.
- . 2008. Demand-system estimation: Update. *Stata Journal* 8: 554–556.
- Zellner, A. 1962. An efficient method of estimating seemingly unrelated regressions and tests for aggregation bias. *Journal of the American Statistical Association* 57: 348–368.
- . 1963. Estimators for seemingly unrelated regression equations: Some exact finite sample results. *Journal of the American Statistical Association* 58: 977–992.
- Zellner, A., and D. S. Huang. 1962. Further properties of efficient estimators for seemingly unrelated regression equations. *International Economic Review* 3: 300–313.

## Also see

- [R] **nl** — Nonlinear least-squares estimation
- [R] **gmm** — Generalized method of moments estimation
- [R] **sureg** — Zellner’s seemingly unrelated regression
- [R] **reg3** — Three-stage estimation for systems of simultaneous equations
- [U] **20 Estimation and postestimation commands**

Description

The following postestimation commands are available for `nlsur`:

| command                           | description                                                                                         |
|-----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>                | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estimates</code>            | cataloging estimation results                                                                       |
| <code>lincom</code>               | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code>               | likelihood-ratio test                                                                               |
| <code>margins</code> <sup>1</sup> | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>                | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>              | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>            | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>                 | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>               | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> You must specify the `variables()` option with `nlsur`.

See the corresponding entries in the *Base Reference Manual* for details.

Syntax for predict

```
predict [type] newvar [if] [in] [, equation(#eqno) yhat residuals]
```

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

Menu

Statistics > Postestimation > Predictions, residuals, etc.

Options for predict

Main

`equation(#eqno)` specifies to which equation you are referring. `equation(#1)` would mean that the calculation is to be made for the first equation, `equation(#2)` would mean the second, and so on. If you do not specify `equation()`, results are the same as if you had specified `equation(#1)`.  
`yhat`, the default, calculates the fitted values for the specified equation.  
`residuals` calculates the residuals for the specified equation.

# Remarks

## Example 1

In [example 2](#) of [\[R\] nlstur](#), we fit a four-factor translog cost function to data for the U.S. economy. The own-price elasticity for a factor measures the percentage change in its usage as a result of a 1% increase in the factor’s price, assuming that output is held constant. For the translog production function, the own-price factor elasticities are

$$\eta_i = \frac{\delta_{ii} + s_i(s_i - 1)}{s_i}$$

Here we compute the elasticity for capital at the sample mean of capital’s factor share. First, we use `summarize` to get the mean of `s_k` and store that value in a scalar:

```
. summarize s_k
+-----+-----+-----+-----+-----+
Variable | Obs Mean Std. Dev. Min Max
+-----+-----+-----+-----+-----+
s_k | 25 .053488 .0044795 .04602 .06185
+-----+-----+-----+-----+-----+
. scalar kmean = r(mean)
```

Now we can use `nlcom` to calculate the elasticity:

```
. nlcom (([dkk]_cons + kmean*(kmean-1)) / kmean)
 _nl_1: ([dkk]_cons + kmean*(kmean-1)) / kmean
+-----+-----+-----+-----+-----+-----+
| | Coef. Std. Err. z P>|z| [95% Conf. Interval]
+-----+-----+-----+-----+-----+-----+
| _nl_1 | -.3952986 .1083535 -3.65 0.000 -.6076676 -.1829295
+-----+-----+-----+-----+-----+-----+
```

If the price of capital increases by 1%, its usage will decrease by about 0.4%. To maintain its current level of output, a firm would increase its usage of other inputs to compensate for the lower capital usage. The standard error reported by `nlcom` reflects the sampling variance of the estimated parameter  $\widehat{\delta_{kk}}$ , but `nlcom` treats the sample mean of `s_k` as a fixed parameter that does not contribute to the sampling variance of the estimated elasticity.



# Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[\[R\] nlstur](#) — Estimation of nonlinear systems of equations

[\[U\] 20 Estimation and postestimation commands](#)

Syntax

```
nptrend varname [if] [in] , by(groupvar) [nodetail score(scorevar)]
```

Menu

Statistics > Nonparametric analysis > Tests of hypotheses > Trend test across ordered groups

Description

nptrend performs a nonparametric test for trend across ordered groups.

Options

Main

by(*groupvar*) is required; it specifies the group on which the data are to be ordered.

nodetail suppresses the listing of group rank sums.

score(*scorevar*) defines scores for groups. When it is not specified, the values of *groupvar* are used for the scores.

Remarks

nptrend performs the nonparametric test for trend across ordered groups developed by [Cuzick \(1985\)](#), which is an extension of the Wilcoxon rank-sum test (`ranksum`; see [\[R\] signrank](#)). A correction for ties is incorporated into the test. `nptrend` is a useful adjunct to the Kruskal–Wallis test; see [\[R\] kwallis](#).

In addition to `nptrend`, the `signtest` and `spearman` commands can be useful for nongrouped data; see [\[R\] signrank](#) and [\[R\] spearman](#). The Cox and Stuart test, for instance, applies the sign test to differences between equally spaced observations of *varname*. The Daniels test calculates Spearman’s rank correlation of *varname* with a time index. Under appropriate conditions, the Daniels test is more powerful than the Cox and Stuart test. See [Conover \(1999\)](#) for a discussion of these tests and their asymptotic relative efficiency.

➤ Example 1

The following data ([Altman 1991](#), 217) show ocular exposure to ultraviolet radiation for 32 pairs of sunglasses classified into three groups according to the amount of visible light transmitted.

| Group | Transmission of visible light | Ocular exposure to ultraviolet radiation    |
|-------|-------------------------------|---------------------------------------------|
| 1     | < 25%                         | 1.4 1.4 1.4 1.6 2.3 2.3                     |
| 2     | 25 to 35%                     | 0.9 1.0 1.1 1.1 1.2 1.2 1.5 1.9 2.2 2.6 2.6 |
|       |                               | 2.6 2.8 2.8 3.2 3.5 4.3 5.1                 |
| 3     | > 35%                         | 0.8 1.7 1.7 1.7 3.4 7.1 8.9 13.5            |

Entering these data into Stata, we have

```
. use http://www.stata-press.com/data/r11/sg
. list, sep(6)
```

|     | group            | exposure |
|-----|------------------|----------|
| 1.  | 1                | 1.4      |
| 2.  | 1                | 1.4      |
| 3.  | 1                | 1.4      |
| 4.  | 1                | 1.6      |
| 5.  | 1                | 2.3      |
| 6.  | 1                | 2.3      |
| 7.  | 2                | .9       |
|     | (output omitted) |          |
| 31. | 3                | 8.9      |
| 32. | 3                | 13.5     |

We use `npntrend` to test for a trend of (increasing) exposure across the three groups by typing

```
. npntrend exposure, by(group)

 group score obs sum of ranks

 1 1 6 76
 2 2 18 290
 3 3 8 162

 z = 1.52
Prob > |z| = 0.129
```

When the groups are given any equally spaced scores (such as  $-1, 0, 1$ ), we will obtain the same answer as above. To illustrate the effect of changing scores, an analysis of these data with scores 1, 2, and 5 (admittedly not sensible here) produces

```
. gen mysc = cond(group==3,5,group)
. npntrend exposure, by(group) score(mysc)

 group score obs sum of ranks

 1 1 6 76
 2 2 18 290
 3 5 8 162

 z = 1.46
Prob > |z| = 0.143
```

This example suggests that the analysis is not all that sensitive to the scores chosen.

❑ Technical note

The grouping variable may be either a string variable or a numeric variable. If it is a string variable and no score variable is specified, the natural numbers 1, 2, 3, ... are assigned to the groups in the sort order of the string variable. This may not always be what you expect. For example, the sort order of the strings “one”, “two”, “three” is “one”, “three”, “two”.



## Saved results

nptrend saves the following in `r()`:

Scalars

|                   |                        |                   |                |
|-------------------|------------------------|-------------------|----------------|
| <code>r(N)</code> | number of observations | <code>r(z)</code> | $z$ statistic  |
| <code>r(p)</code> | two-sided $p$ -value   | <code>r(T)</code> | test statistic |

## Methods and formulas

nptrend is implemented as an ado-file.

nptrend is based on a method in [Cuzick \(1985\)](#). The following description of the statistic is from [Altman \(1991, 215–217\)](#). We have  $k$  groups of sample sizes  $n_i$  ( $i = 1, \dots, k$ ). The groups are given scores,  $l_i$ , which reflect their ordering, such as 1, 2, and 3. The scores do not have to be equally spaced, but they usually are.  $N = \sum n_i$  observations are ranked from 1 to  $N$ , and the sums of the ranks in each group,  $R_i$ , are obtained.  $L$ , the weighted sum of all the group scores, is

$$L = \sum_{i=1}^k l_i n_i$$

The statistic  $T$  is calculated as

$$T = \sum_{i=1}^k l_i R_i$$

Under the null hypothesis, the expected value of  $T$  is  $E(T) = 0.5(N+1)L$ , and its standard error is

$$\text{se}(T) = \sqrt{\frac{n+1}{12} \left( N \sum_{i=1}^k l_i^2 n_i - L^2 \right)}$$

so that the test statistic,  $z$ , is given by  $z = \{T - E(T)\} / \text{se}(T)$ , which has an approximately standard normal distribution when the null hypothesis of no trend is true.

The correction for ties affects the standard error of  $T$ . Let  $\tilde{N}$  be the number of unique values of the variable being tested ( $\tilde{N} \leq N$ ), and let  $t_j$  be the number of times the  $j$ th unique value of the variable appears in the data. Define

$$a = \frac{\sum_{j=1}^{\tilde{N}} t_j(t_j^2 - 1)}{N(N^2 - 1)}$$

The corrected standard error of  $T$  is  $\tilde{\text{se}}(T) = \sqrt{1-a} \text{se}(T)$ .

## Acknowledgments

nptrend was written by K. A. Stepniowska and D. G. Altman ([1992](#)) of the University of Oxford, UK.

## References

- Altman, D. G. 1991. *Practical Statistics for Medical Research*. London: Chapman & Hall/CRC.
- Conover, W. J. 1999. *Practical Nonparametric Statistics*. 3rd ed. New York: Wiley.
- Cuzick, J. 1985. A Wilcoxon-type test for trend. *Statistics in Medicine* 4: 87–90.
- Sasieni, P. 1996. [snp12: Stratified test for trend across ordered groups](#). *Stata Technical Bulletin* 33: 24–27. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 196–200. College Station, TX: Stata Press.
- Sasieni, P., K. A. Stepniowska, and D. G. Altman. 1996. [snp11: Test for trend across ordered groups revisited](#). *Stata Technical Bulletin* 32: 27–29. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 193–196. College Station, TX: Stata Press.
- Stepniowska, K. A., and D. G. Altman. 1992. [snp4: Non-parametric test for trend across ordered groups](#). *Stata Technical Bulletin* 9: 21–22. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, p. 169. College Station, TX: Stata Press.

## Also see

- [R] [kwallis](#) — Kruskal–Wallis equality-of-populations rank test
- [R] [signrank](#) — Equality tests on matched data
- [R] [spearman](#) — Spearman’s and Kendall’s correlations
- [R] [symmetry](#) — Symmetry and marginal homogeneity tests
- [ST] [epitab](#) — Tables for epidemiologists
- [ST] [strate](#) — Tabulate failure rates and rate ratios



Syntax

```
ologit depvar [indepvars] [if] [in] [weight] [, options]
```

| <i>options</i>                            | description                                                                                                                |
|-------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| Model                                     |                                                                                                                            |
| <u>offset</u> ( <i>varname</i> )          | include <i>varname</i> in model with coefficient constrained to 1                                                          |
| <u>constraints</u> ( <i>constraints</i> ) | apply specified linear constraints                                                                                         |
| <u>collinear</u>                          | keep collinear variables                                                                                                   |
| SE/Robust                                 |                                                                                                                            |
| <u>vce</u> ( <i>vcetype</i> )             | <i>vcetype</i> may be <u>oim</u> , <u>robust</u> , <u>cluster</u> <i>clustvar</i> , <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting                                 |                                                                                                                            |
| <u>level</u> (#)                          | set confidence level; default is <u>level</u> (95)                                                                         |
| <u>or</u>                                 | report odds ratios                                                                                                         |
| <u>nocnsreport</u>                        | do not display constraints                                                                                                 |
| <i>display_options</i>                    | control spacing and display of omitted variables and base and empty cells                                                  |
| Maximization                              |                                                                                                                            |
| <i>maximize_options</i>                   | control the maximization process; seldom used                                                                              |
| <sup>†</sup> <u>coeflegend</u>            | display coefficients' legend instead of coefficient table                                                                  |

<sup>†</sup>coeflegend does not appear in the dialog box.  
*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.  
*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.  
bootstrap, by, frapoly, jackknife, mfp, mi estimate, nestreg, rolling, statsby, stepwise, and svy are allowed; see [U] 11.1.10 Prefix commands.  
vce(bootstrap) and vce(jackknife) are not allowed with the mi estimate prefix.  
Weights are not allowed with the bootstrap prefix.  
vce() and weights are not allowed with the svy prefix.  
fweights, iweights, and pweights are allowed; see [U] 11.1.6 weight.  
See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Ordinal outcomes > Ordered logistic regression

Description

`ologit` fits ordered logit models of ordinal variable *depvar* on the independent variables *indepvars*. The actual values taken on by the dependent variable are irrelevant, except that larger values are assumed to correspond to “higher” outcomes.

See [R] **logistic** for a list of related estimation commands.

## Options

### Model

`offset(varname)`, `constraints(constraints)`, `collinear`; see [\[R\] estimation options](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [\[R\] vce\\_option](#).

### Reporting

`level(#)`; see [\[R\] estimation options](#).

`or` reports the estimated coefficients transformed to odds ratios, i.e.,  $e^b$  rather than  $b$ . Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated. `or` may be specified at estimation or when replaying previously estimated results.

`nocnsreport`; see [\[R\] estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [\[R\] estimation options](#).

### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [\[R\] maximize](#). These options are seldom used.

The following option is available with `ologit` but is not shown in the dialog box:

`coeflegend`; see [\[R\] estimation options](#).

## Remarks

Ordered logit models are used to estimate relationships between an ordinal dependent variable and a set of independent variables. An *ordinal* variable is a variable that is categorical and ordered, for instance, “poor”, “good”, and “excellent”, which might indicate a person’s current health status or the repair record of a car. If there are only two outcomes, see [\[R\] logistic](#), [\[R\] logit](#), and [\[R\] probit](#). This entry is concerned only with more than two outcomes. If the outcomes cannot be ordered (e.g., residency in the north, east, south, or west), see [\[R\] mlogit](#). This entry is concerned only with models in which the outcomes can be ordered.

In ordered logit, an underlying score is estimated as a linear function of the independent variables and a set of cutpoints. The probability of observing outcome  $i$  corresponds to the probability that the estimated linear function, plus random error, is within the range of the cutpoints estimated for the outcome:

$$\Pr(\text{outcome}_j = i) = \Pr(\kappa_{i-1} < \beta_1 x_{1j} + \beta_2 x_{2j} + \cdots + \beta_k x_{kj} + u_j \leq \kappa_i)$$

$u_j$  is assumed to be logistically distributed in ordered logit. In either case, we estimate the coefficients  $\beta_1, \beta_2, \dots, \beta_k$  together with the cutpoints  $\kappa_1, \kappa_2, \dots, \kappa_{k-1}$ , where  $k$  is the number of possible outcomes.  $\kappa_0$  is taken as  $-\infty$ , and  $\kappa_k$  is taken as  $+\infty$ . All of this is a direct generalization of the ordinary two-outcome logit model.

## ► Example 1

We wish to analyze the 1977 repair records of 66 foreign and domestic cars. The data are a variation of the automobile dataset described in [U] 1.2.2 Example datasets. The 1977 repair records, like those in 1978, take on values “Poor”, “Fair”, “Average”, “Good”, and “Excellent”. Here is a cross-tabulation of the data:

```
. use http://www.stata-press.com/data/r11/fullauto
(Automobile Models)
```

```
. tabulate rep77 foreign, chi2
```

| Repair<br>Record<br>1977 | Foreign  |         | Total |
|--------------------------|----------|---------|-------|
|                          | Domestic | Foreign |       |
| Poor                     | 2        | 1       | 3     |
| Fair                     | 10       | 1       | 11    |
| Average                  | 20       | 7       | 27    |
| Good                     | 13       | 7       | 20    |
| Excellent                | 0        | 5       | 5     |
| Total                    | 45       | 21      | 66    |

```
Pearson chi2(4) = 13.8619 Pr = 0.008
```

Although it appears that `foreign` takes on the values “Domestic” and “Foreign”, it is actually a numeric variable taking on the values 0 and 1. Similarly, `rep77` takes on the values 1, 2, 3, 4, and 5, corresponding to “Poor”, “Fair”, and so on. The more meaningful words appear because we have attached value labels to the data; see [U] 12.6.3 Value labels.

Because the chi-squared value is significant, we could claim that there is a relationship between `foreign` and `rep77`. Literally, however, we can only claim that the distributions are different; the chi-squared test is not directional. One way to model these data is to model the categorization that took place when the data were created. Cars have a true frequency of repair, which we will assume is given by  $S_j = \beta \text{foreign}_j + u_j$ , and a car is categorized as “poor” if  $S_j \leq \kappa_0$ , as “fair” if  $\kappa_0 < S_j \leq \kappa_1$ , and so on:

```
. ologit rep77 foreign
```

```
Iteration 0: log likelihood = -89.895098
Iteration 1: log likelihood = -85.951765
Iteration 2: log likelihood = -85.908227
Iteration 3: log likelihood = -85.908161
Iteration 4: log likelihood = -85.908161
```

```
Ordered logistic regression
```

```
Number of obs = 66
LR chi2(1) = 7.97
Prob > chi2 = 0.0047
Pseudo R2 = 0.0444
```

```
Log likelihood = -85.908161
```

| rep77   | Coef.     | Std. Err. | z    | P> z  | [95% Conf. Interval] |           |
|---------|-----------|-----------|------|-------|----------------------|-----------|
| foreign | 1.455878  | .5308951  | 2.74 | 0.006 | .4153425             | 2.496413  |
| /cut1   | -2.765562 | .5988208  |      |       | -3.939229            | -1.591895 |
| /cut2   | -.9963603 | .3217706  |      |       | -1.627019            | -.3657016 |
| /cut3   | .9426153  | .3136398  |      |       | .3278925             | 1.557338  |
| /cut4   | 3.123351  | .5423257  |      |       | 2.060412             | 4.18629   |

Our model is  $S_j = 1.46 \text{foreign}_j + u_j$ ; the expected value for foreign cars is 1.46 and, for domestic cars, 0; foreign cars have better repair records.

The estimated cutpoints tell us how to interpret the score. For a foreign car, the probability of a poor record is the probability that  $1.46 + u_j \leq -2.77$ , or equivalently,  $u_j \leq -4.23$ . Making this calculation requires familiarity with the logistic distribution: the probability is  $1/(1 + e^{4.23}) = 0.014$ . On the other hand, for domestic cars, the probability of a poor record is the probability  $u_j \leq -2.77$ , which is 0.059.

This, it seems to us, is a far more reasonable prediction than we would have made based on the table alone. The table showed that 2 of 45 domestic cars had poor records, whereas 1 of 21 foreign cars had poor records—corresponding to probabilities  $2/45 = 0.044$  and  $1/21 = 0.048$ . The predictions from our model imposed a smoothness assumption—foreign cars should not, overall, have better repair records without the difference revealing itself in each category. In our data, the fractions of foreign and domestic cars in the poor category are virtually identical only because of the randomness associated with small samples.

Thus if we were asked to predict the true fractions of foreign and domestic cars that would be classified in the various categories, we would choose the numbers implied by the ordered logit model:

|           | tabulate |         | logit    |         |
|-----------|----------|---------|----------|---------|
|           | Domestic | Foreign | Domestic | Foreign |
| Poor      | 0.044    | 0.048   | 0.059    | 0.014   |
| Fair      | 0.222    | 0.048   | 0.210    | 0.065   |
| Average   | 0.444    | 0.333   | 0.450    | 0.295   |
| Good      | 0.289    | 0.333   | 0.238    | 0.467   |
| Excellent | 0.000    | 0.238   | 0.043    | 0.159   |

See [\[R\] ologit postestimation](#) for a more complete explanation of how to generate predictions from an ordered logit model.



□ Technical note

Here ordered logit provides an alternative to ordinary two-outcome logistic models with an arbitrary dichotomization, which might otherwise have been tempting. We could, for instance, have summarized these data by converting the five-outcome `rep77` variable to a two-outcome variable, combining cars in the average, fair, and poor categories to make one outcome and combining cars in the good and excellent categories to make the second.

Another even less appealing alternative would have been to use ordinary regression, arbitrarily labeling “excellent” as 5, “good” as 4, and so on. The problem is that with different but equally valid labelings (say, 10 for “excellent”), we would obtain different estimates. We would have no way of choosing one metric over another. That assertion is not, however, true of `ologit`. The actual values used to label the categories make no difference other than through the order they imply.

In fact, our labeling was 5 for “excellent”, 4 for “good”, and so on. The words “excellent” and “good” appear in our output because we attached a value label to the variables; see [\[U\] 12.6.3 Value labels](#). If we were to now go back and type `replace rep77=10 if rep77==5`, changing all the 5s to 10s, we would still obtain the same results when we refit our model.



## ► Example 2

In the [example](#) above, we used ordered logit as a way to model a table. We are not, however, limited to including only one explanatory variable or to including only categorical variables. We can explore the relationship of `rep77` with any of the variables in our data. We might, for instance, model `rep77` not only in terms of the origin of manufacture, but also including `length` (a proxy for size) and `mpg`:

```
. ologit rep77 foreign length mpg
Iteration 0: log likelihood = -89.895098
Iteration 1: log likelihood = -78.775147
Iteration 2: log likelihood = -78.254294
Iteration 3: log likelihood = -78.250719
Iteration 4: log likelihood = -78.250719

Ordered logistic regression Number of obs = 66
 LR chi2(3) = 23.29
 Prob > chi2 = 0.0000
 Pseudo R2 = 0.1295

Log likelihood = -78.250719
```

| rep77   | Coef.    | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|---------|----------|-----------|------|-------|----------------------|----------|
| foreign | 2.896807 | .7906411  | 3.66 | 0.000 | 1.347179             | 4.446435 |
| length  | .0828275 | .02272    | 3.65 | 0.000 | .0382972             | .1273579 |
| mpg     | .2307677 | .0704548  | 3.28 | 0.001 | .0926788             | .3688566 |
| /cut1   | 17.92748 | 5.551191  |      |       | 7.047344             | 28.80761 |
| /cut2   | 19.86506 | 5.59648   |      |       | 8.896161             | 30.83396 |
| /cut3   | 22.10331 | 5.708936  |      |       | 10.914               | 33.29262 |
| /cut4   | 24.69213 | 5.890754  |      |       | 13.14647             | 36.2378  |

`foreign` still plays a role—and an even larger role than previously. We find that larger cars tend to have better repair records, as do cars with better mileage ratings.



*(Continued on next page)*

## Saved results

`ologit` saves the following in `e()`:

### Scalars

|                            |                                                |
|----------------------------|------------------------------------------------|
| <code>e(N)</code>          | number of observations                         |
| <code>e(N_cd)</code>       | number of completely determined observations   |
| <code>e(k_cat)</code>      | number of categories                           |
| <code>e(k)</code>          | number of parameters                           |
| <code>e(k_aux)</code>      | number of auxiliary parameters                 |
| <code>e(k_eq)</code>       | number of equations                            |
| <code>e(k_eq_model)</code> | number of equations in model Wald test         |
| <code>e(k_dv)</code>       | number of dependent variables                  |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints |
| <code>e(df_m)</code>       | model degrees of freedom                       |
| <code>e(r2_p)</code>       | pseudo- <i>R</i> -squared                      |
| <code>e(ll)</code>         | log likelihood                                 |
| <code>e(ll_0)</code>       | log likelihood, constant-only model            |
| <code>e(N_clust)</code>    | number of clusters                             |
| <code>e(chi2)</code>       | $\chi^2$                                       |
| <code>e(p)</code>          | significance of model test                     |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                      |
| <code>e(ic)</code>         | number of iterations                           |
| <code>e(rc)</code>         | return code                                    |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                    |

### Macros

|                                 |                                                                          |
|---------------------------------|--------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>ologit</code>                                                      |
| <code>e(cmdline)</code>         | command as typed                                                         |
| <code>e(depvar)</code>          | name of dependent variable                                               |
| <code>e(wtype)</code>           | weight type                                                              |
| <code>e(wexp)</code>            | weight expression                                                        |
| <code>e(title)</code>           | title in estimation output                                               |
| <code>e(clustvar)</code>        | name of cluster variable                                                 |
| <code>e(offset)</code>          | offset                                                                   |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                  |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                           |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                            |
| <code>e(opt)</code>             | type of optimization                                                     |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>       | type of ml method                                                        |
| <code>e(user)</code>            | name of likelihood-evaluator program                                     |
| <code>e(technique)</code>       | maximization technique                                                   |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular              |
| <code>e(crittype)</code>        | optimization criterion                                                   |
| <code>e(properties)</code>      | b V                                                                      |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                           |
| <code>e(marginsprop)</code>     | signals to the <code>margins</code> command                              |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>           |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>           |

|                              |                                              |
|------------------------------|----------------------------------------------|
| Matrices                     |                                              |
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(gradient)</code>     | gradient vector                              |
| <code>e(cat)</code>          | category values                              |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |
| Functions                    |                                              |
| <code>e(sample)</code>       | marks estimation sample                      |

## Methods and formulas

`ologit` is implemented as an ado-file.

See Long and Freese (2006, chap. 5) for a discussion of models for ordinal outcomes and examples that use Stata. Cameron and Trivedi (2005, chap. 15) describe multinomial models, including the model fit by `ologit`. When you have a qualitative dependent variable, several estimation procedures are available. A popular choice is multinomial logistic regression (see [R] `mlogit`), but if you use this procedure when the response variable is ordinal, you are discarding information because multinomial logit ignores the ordered aspect of the outcome. Ordered logit and probit models provide a means to exploit the ordering information.

There is more than one “ordered logit” model. The model fit by `ologit`, which we will call the ordered logit model, is also known as the proportional odds model. Another popular choice, not fit by `ologit`, is known as the stereotype model; see [R] `slogit`. All ordered logit models have been derived by starting with a binary logit/probit model and generalizing it to allow for more than two outcomes.

The proportional-odds ordered logit model is so called because, if we consider the odds  $\text{odds}(k) = P(Y \leq k)/P(Y > k)$ , then  $\text{odds}(k_1)$  and  $\text{odds}(k_2)$  have the same ratio for all independent variable combinations. The model is based on the principle that the only effect of combining adjoining categories in ordered categorical regression problems should be a loss of efficiency in estimating the regression parameters (McCullagh 1980). This model was also described by McKelvey and Zavoina (1975) and, previously, by Aitchison and Silvey (1957) in a different algebraic form. Brant (1990) offers a set of diagnostics for the model.

Peterson and Harrell Jr. (1990) suggest a model that allows nonproportional odds for a subset of the explanatory variables. `ologit` does not allow this, but a model similar to this was implemented by Fu (1998).

The stereotype model rejects the principle on which the ordered logit model is based. Anderson (1984) argues that there are two distinct types of ordered categorical variables: “grouped continuous”, such as income, where the “type a” model applies; and “assessed”, such as extent of pain relief, where the stereotype model applies. Greenland (1985) independently developed the same model. The stereotype model starts with a multinomial logistic regression model and imposes constraints on this model.

Goodness of fit for `ologit` can be evaluated by comparing the likelihood value with that obtained by fitting the model with `mlogit`. Let  $\ln L_1$  be the log-likelihood value reported by `ologit`, and let  $\ln L_0$  be the log-likelihood value reported by `mlogit`. If there are  $p$  independent variables (excluding the constant) and  $k$  categories, `mlogit` will estimate  $p(k - 1)$  additional parameters. We can then perform a “likelihood-ratio test”, i.e., calculate  $-2(\ln L_1 - \ln L_0)$ , and compare it with  $\chi^2\{p(k - 2)\}$ .

This test is suggestive only because the ordered logit model is not nested within the multinomial logit model. A large value of  $-2(\ln L_1 - \ln L_0)$  should, however, be taken as evidence of poorness of fit. Marginally large values, on the other hand, should not be taken too seriously.

The coefficients and cutpoints are estimated using maximum likelihood as described in [R] [maximize](#). In our parameterization, no constant appears, because the effect is absorbed into the cutpoints.

`ologit` and `oprobit` begin by tabulating the dependent variable. Category  $i = 1$  is defined as the minimum value of the variable,  $i = 2$  as the next ordered value, and so on, for the empirically determined  $k$  categories.

The probability of a given observation for ordered logit is

$$\begin{aligned} p_{ij} &= \Pr(y_j = i) = \Pr(\kappa_{i-1} < \mathbf{x}_j\boldsymbol{\beta} + u \leq \kappa_i) \\ &= \frac{1}{1 + \exp(-\kappa_i + \mathbf{x}_j\boldsymbol{\beta})} - \frac{1}{1 + \exp(-\kappa_{i-1} + \mathbf{x}_j\boldsymbol{\beta})} \end{aligned}$$

$\kappa_0$  is defined as  $-\infty$  and  $\kappa_k$  as  $+\infty$ .

For ordered probit, the probability of a given observation is

$$\begin{aligned} p_{ij} &= \Pr(y_j = i) = \Pr(\kappa_{i-1} < \mathbf{x}_j\boldsymbol{\beta} + u \leq \kappa_i) \\ &= \Phi(\kappa_i - \mathbf{x}_j\boldsymbol{\beta}) - \Phi(\kappa_{i-1} - \mathbf{x}_j\boldsymbol{\beta}) \end{aligned}$$

where  $\Phi(\cdot)$  is the standard normal cumulative distribution function.

The log likelihood is

$$\ln L = \sum_{j=1}^N w_j \sum_{i=1}^k I_i(y_j) \ln p_{ij}$$

where  $w_j$  is an optional weight and

$$I_i(y_j) = \begin{cases} 1, & \text{if } y_j = i \\ 0, & \text{otherwise} \end{cases}$$

`ologit` and `oprobit` support the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

These commands also support estimation with survey data. For details on VCEs with survey data, see [SVY] [variance estimation](#).

## References

- Aitchison, J., and S. D. Silvey. 1957. The generalization of probit analysis to the case of multiple responses. *Biometrika* 44: 131–140.
- Anderson, J. A. 1984. Regression and ordered categorical variables (with discussion). *Journal of the Royal Statistical Society, Series B* 46: 1–30.
- Brant, R. 1990. Assessing proportionality in the proportional odds model for ordinal logistic regression. *Biometrics* 46: 1171–1178.



- Cameron, A. C., and P. K. Trivedi. 2005. *Microeconometrics: Methods and Applications*. New York: Cambridge University Press.
- Fu, V. K. 1998. [sg88: Estimating generalized ordered logit models](#). *Stata Technical Bulletin* 44: 27–30. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 160–164. College Station, TX: Stata Press.
- Goldstein, R. 1997. [sg59: Index of ordinal variation and Neyman–Barton GOF](#). *Stata Technical Bulletin* 33: 10–12. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 145–147. College Station, TX: Stata Press.
- Greenland, S. 1985. An application of logistic models to the analysis of ordinal responses. *Biometrical Journal* 27: 189–197.
- Kleinbaum, D. G., and M. Klein. 2002. *Logistic Regression: A Self-Learning Text*. 2nd ed. New York: Springer.
- Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.
- Long, J. S., and J. Freese. 2006. *Regression Models for Categorical Dependent Variables Using Stata*. 2nd ed. College Station, TX: Stata Press.
- Lunt, M. 2001. [sg163: Stereotype ordinal regression](#). *Stata Technical Bulletin* 61: 12–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 298–307. College Station, TX: Stata Press.
- McCullagh, P. 1977. A logistic model for paired comparisons with ordered categorical data. *Biometrika* 64: 449–453.
- . 1980. Regression models for ordinal data (with discussion). *Journal of the Royal Statistical Society, Series B* 42: 109–142.
- McCullagh, P., and J. A. Nelder. 1989. *Generalized Linear Models*. 2nd ed. London: Chapman & Hall/CRC.
- McKelvey, R. D., and W. Zavoina. 1975. A statistical model for the analysis of ordinal level dependent variables. *Journal of Mathematical Sociology* 4: 103–120.
- Miranda, A., and S. Rabe-Hesketh. 2006. [Maximum likelihood estimation of endogenous switching and sample selection models for binary, ordinal, and count variables](#). *Stata Journal* 6: 285–308.
- Peterson, B., and F. E. Harrell Jr. 1990. Partial proportional odds models for ordinal response variables. *Applied Statistics* 39: 205–217.
- Williams, R. 2006. [Generalized ordered logit/partial proportional odds models for ordinal dependent variables](#). *Stata Journal* 6: 58–82.
- Wolfe, R. 1998. [sg86: Continuation-ratio models for ordinal response data](#). *Stata Technical Bulletin* 44: 18–21. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 149–153. College Station, TX: Stata Press.
- Wolfe, R., and W. W. Gould. 1998. [sg76: An approximate likelihood-ratio test for ordinal response models](#). *Stata Technical Bulletin* 42: 24–27. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 199–204. College Station, TX: Stata Press.
- Xu, J., and J. S. Long. 2005. [Confidence intervals for predicted outcomes in regression models for categorical outcomes](#). *Stata Journal* 5: 537–559.

## Also see

- [R] [ologit postestimation](#) — Postestimation tools for ologit
- [R] [clogit](#) — Conditional (fixed-effects) logistic regression
- [R] [logistic](#) — Logistic regression, reporting odds ratios
- [R] [logit](#) — Logistic regression, reporting coefficients
- [R] [mlogit](#) — Multinomial (polytomous) logistic regression
- [R] [oprobit](#) — Ordered probit regression
- [R] [rologit](#) — Rank-ordered logistic regression
- [R] [slogit](#) — Stereotype logistic regression
- [SVY] [svy estimation](#) — Estimation commands for survey data
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `ologit`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>            | link test for model specification                                                                   |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Syntax for predict

```
predict [type] newvars [if] [in] [, statistic outcome(outcome) nooffset]

predict [type] { stub*|newvarreg newvarκ1 ... newvarκk-1 } [if] [in] , scores
```

where  $k$  is the number of outcomes in the model.

| statistic         | description                             |
|-------------------|-----------------------------------------|
| Main              |                                         |
| <code>pr</code>   | predicted probabilities; the default    |
| <code>xb</code>   | linear prediction                       |
| <code>stdp</code> | standard error of the linear prediction |

If you do not specify `outcome()`, `pr` (with one new variable specified) assumes `outcome(#1)`.  
You specify one or  $k$  new variables with `pr`, where  $k$  is the number of outcomes.  
You specify one new variable with `xb` and `stdp`.  
These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

### Main

**pr**, the default, calculates the predicted probabilities. If you do not also specify the `outcome()` option, you must specify  $k$  new variables, where  $k$  is the number of categories of the dependent variable. Say that you fit a model by typing `ologit result x1 x2`, and `result` takes on three values. Then you could type `predict p1 p2 p3` to obtain all three predicted probabilities. If you specify the `outcome()` option, you must specify one new variable. Say that `result` takes on the values 1, 2, and 3. Typing `predict p1, outcome(1)` would produce the same `p1`.

**xb** calculates the linear prediction. You specify one new variable, for example, `predict linear, xb`. The linear prediction is defined, ignoring the contribution of the estimated cutpoints.

**stdp** calculates the standard error of the linear prediction. You specify one new variable, for example, `predict se, stdp`.

**outcome(outcome)** specifies for which outcome the predicted probabilities are to be calculated. `outcome()` should contain either one value of the dependent variable or one of #1, #2, ..., with #1 meaning the first category of the dependent variable, #2 meaning the second category, etc.

**nooffset** is relevant only if you specified `offset(varname)` for `ologit`. It modifies the calculations made by `predict` so that they ignore the offset variable; the linear prediction is treated as  $\mathbf{x}_j\mathbf{b}$  rather than as  $\mathbf{x}_j\mathbf{b} + \text{offset}_j$ .

**scores** calculates equation-level score variables. The number of score variables created will equal the number of outcomes in the model. If the number of outcomes in the model was  $k$ , then

the first new variable will contain  $\partial \ln L / \partial (\mathbf{x}_j\mathbf{b})$ ;

the second new variable will contain  $\partial \ln L / \partial \kappa_1$ ;

the third new variable will contain  $\partial \ln L / \partial \kappa_2$ ;

...

and the  $k$ th new variable will contain  $\partial \ln L / \partial \kappa_{k-1}$ , where  $\kappa_i$  refers to the  $i$ th cutpoint.

## Remarks

See [U] [20 Estimation and postestimation commands](#) for instructions on obtaining the variance–covariance matrix of the estimators, predicted values, and hypothesis tests. Also see [R] [lrtest](#) for performing likelihood-ratio tests.

### ► Example 1

In [example 2](#) of [R] [ologit](#), we fit the model `ologit rep77 foreign length mpg`. The `predict` command can be used to obtain the predicted probabilities.

We type `predict` followed by the names of the new variables to hold the predicted probabilities, ordering the names from low to high. In our data, the lowest outcome is “poor”, and the highest is “excellent”. We have five categories, so we must type five names following `predict`; the choice of names is up to us:

```
. predict poor fair avg good exc
(option pr assumed; predicted probabilities)
```

```
. list exc good make model rep78 if rep77>=., sep(4) divider
```

|     | exc      | good     | make    | model   | rep78   |
|-----|----------|----------|---------|---------|---------|
| 3.  | .0033341 | .0393056 | AMC     | Spirit  | .       |
| 10. | .0098392 | .1070041 | Buick   | Opel    | .       |
| 32. | .0023406 | .0279497 | Ford    | Fiesta  | Good    |
| 44. | .015697  | .1594413 | Merc.   | Monarch | Average |
| 53. | .065272  | .4165188 | Peugeot | 604     | .       |
| 56. | .005187  | .059727  | Plym.   | Horizon | Average |
| 57. | .0261461 | .2371826 | Plym.   | Sapporo | .       |
| 63. | .0294961 | .2585825 | Pont.   | Phoenix | .       |

The eight cars listed were introduced after 1977, so they do not have 1977 repair records in our data. We predicted what their 1977 repair records might have been using the fitted model. We see that, based on its characteristics, the Peugeot 604 had about a  $41.65 + 6.53 \approx 48.2\%$  chance of a good or excellent repair record. The Ford Fiesta, which had only a 3% chance of a good or excellent repair record, in fact, had a good record when it was introduced in the following year.



□ Technical note

For ordered logit, `predict, xb` produces  $S_j = x_{1j}\beta_1 + x_{2j}\beta_2 + \cdots + x_{kj}\beta_k$ . The ordered-logit predictions are then the probability that  $S_j + u_j$  lies between a pair of cutpoints,  $\kappa_{i-1}$  and  $\kappa_i$ . Some handy formulas are

$$\begin{aligned} \Pr(S_j + u_j < \kappa) &= 1/(1 + e^{S_j - \kappa}) \\ \Pr(S_j + u_j > \kappa) &= 1 - 1/(1 + e^{S_j - \kappa}) \\ \Pr(\kappa_1 < S_j + u_j < \kappa_2) &= 1/(1 + e^{S_j - \kappa_2}) - 1/(1 + e^{S_j - \kappa_1}) \end{aligned}$$

Rather than using `predict` directly, we could calculate the predicted probabilities by hand. If we wished to obtain the predicted probability that the repair record is excellent and the probability that it is good, we look back at `ologit`’s output to obtain the cutpoints. We find that “good” corresponds to the interval  $\text{/cut3} < S_j + u < \text{/cut4}$  and “excellent” to the interval  $S_j + u > \text{/cut4}$ :

```
. predict score, xb
. generate probgood = 1/(1+exp(score-_b[/cut4])) - 1/(1+exp(score-_b[/cut3]))
. generate probexc = 1 - 1/(1+exp(score-_b[/cut4]))
```

The results of our calculation will be the same as those produced in the previous example. We refer to the estimated cutpoints just as we would any coefficient, so `_b[/cut3]` refers to the value of the `/cut3` coefficient; see [U] 13.5 Accessing coefficients and standard errors.



Methods and formulas

All postestimation commands listed above are implemented as ado-files.

---

## Also see

[R] [ologit](#) — Ordered logistic regression

[U] [20 Estimation and postestimation commands](#)

# Title

oneway — One-way analysis of variance

## Syntax

`oneway response_var factor_var [if] [in] [weight] [, options]`

| options                    | description                                                                         |
|----------------------------|-------------------------------------------------------------------------------------|
| Main                       |                                                                                     |
| <code>bonferroni</code>    | Bonferroni multiple-comparison test                                                 |
| <code>scheffe</code>       | Schéffé multiple-comparison test                                                    |
| <code>sidak</code>         | Šidák multiple-comparison test                                                      |
| <code>tabulate</code>      | produce summary table                                                               |
| <code>[no] means</code>    | include or suppress means; default is <code>means</code>                            |
| <code>[no] standard</code> | include or suppress standard deviations; default is <code>standard</code>           |
| <code>[no] freq</code>     | include or suppress frequencies; default is <code>freq</code>                       |
| <code>[no] obs</code>      | include or suppress number of obs; default is <code>obs</code> if data are weighted |
| <code>noanova</code>       | suppress the ANOVA table                                                            |
| <code>no label</code>      | show numeric codes, not labels                                                      |
| <code>wrap</code>          | do not break wide tables                                                            |
| <code>missing</code>       | treat missing values as categories                                                  |

by is allowed; see [\[D\] by](#).  
aweights and fweights are allowed; see [\[U\] 11.1.6 weight](#).

## Menu

Statistics > Linear models and related > ANOVA/MANOVA > One-way ANOVA

## Description

The `oneway` command reports one-way analysis-of-variance (ANOVA) models and performs multiple-comparison tests.

If you wish to fit more complicated ANOVA layouts or wish to fit analysis-of-covariance (ANCOVA) models, see [\[R\] anova](#).

See [\[D\] encode](#) for examples of fitting ANOVA models on string variables.

See [\[R\] loneway](#) for an alternative `oneway` command with slightly different features.

## Options

Main

`bonferroni` reports the results of a Bonferroni multiple-comparison test.

`scheffe` reports the results of a Scheffé multiple-comparison test.

`sidak` reports the results of a Šidák multiple-comparison test.

`tabulate` produces a table of summary statistics of the *response\_var* by levels of the *factor\_var*.

The table includes the mean, standard deviation, frequency, and, if the data are weighted, the number of observations. Individual elements of the table may be included or suppressed by using the `[no]means`, `[no]standard`, `[no]freq`, and `[no]obs` options. For example, typing

```
oneway response factor, tabulate means standard
```

produces a summary table that contains only the means and standard deviations. You could achieve the same result by typing

```
oneway response factor, tabulate nofreq
```

`[no]means` includes or suppresses only the means from the table produced by the `tabulate` option. See `tabulate` above.

`[no]standard` includes or suppresses only the standard deviations from the table produced by the `tabulate` option. See `tabulate` above.

`[no]freq` includes or suppresses only the frequencies from the table produced by the `tabulate` option. See `tabulate` above.

`[no]obs` includes or suppresses only the reported number of observations from the table produced by the `tabulate` option. If the data are not weighted, only the frequency is reported. If the data are weighted, the frequency refers to the sum of the weights. See `tabulate` above.

`noanova` suppresses the display of the ANOVA table.

`no label` causes the numeric codes to be displayed rather than the value labels in the ANOVA and multiple-comparison test tables.

`wrap` requests that Stata not break up wide tables to make them more readable.

`missing` requests that missing values of *factor\_var* be treated as a category rather than as observations to be omitted from the analysis.

## Remarks

Remarks are presented under the following headings:

*Introduction*

*Obtaining observed means*

*Multiple-comparison tests*

*Weighted data*

## Introduction

The `oneway` command reports one-way ANOVA models. To perform a one-way layout of a variable called `endog` on `exog`, type `oneway endog exog`.

### ► Example 1

We run an experiment varying the amount of fertilizer used in growing apple trees. We test four concentrations, using each concentration in three groves of 12 trees each. Later in the year, we measure the average weight of the fruit.

If all had gone well, we would have had 3 observations on the average weight for each of the four concentrations. Instead, two of the groves were mistakenly leveled by a confused man on a large bulldozer. We are left with the following dataset:

```
. use http://www.stata-press.com/data/r11/apple
(Apple trees)

. describe
Contains data from http://www.stata-press.com/data/r11/apple.dta
 obs: 10 Apple trees
 vars: 2 16 Jan 2009 11:23
 size: 140 (99.9% of memory free)
```

| variable name | storage type | display format | value label | variable label          |
|---------------|--------------|----------------|-------------|-------------------------|
| treatment     | int          | %8.0g          |             | Fertilizer              |
| weight        | double       | %10.0g         |             | Average weight in grams |

Sorted by:

```
. list, abbreviate(10)
```

|     | treatment | weight |
|-----|-----------|--------|
| 1.  | 1         | 117.5  |
| 2.  | 1         | 113.8  |
| 3.  | 1         | 104.4  |
| 4.  | 2         | 48.9   |
| 5.  | 2         | 50.4   |
| 6.  | 2         | 58.9   |
| 7.  | 3         | 70.4   |
| 8.  | 3         | 86.9   |
| 9.  | 4         | 87.7   |
| 10. | 4         | 67.3   |

To obtain the one-way ANOVA results, we type

```
. oneway weight treatment
```

| Analysis of Variance |            |    |            |       |          |
|----------------------|------------|----|------------|-------|----------|
| Source               | SS         | df | MS         | F     | Prob > F |
| Between groups       | 5295.54433 | 3  | 1765.18144 | 21.46 | 0.0013   |
| Within groups        | 493.591667 | 6  | 82.2652778 |       |          |
| Total                | 5789.136   | 9  | 643.237333 |       |          |

Bartlett's test for equal variances: chi2(3) = 1.3900 Prob>chi2 = 0.708

We find significant (at better than the 1% level) differences among the four concentrations.

❑ Technical note

Rather than using the `oneway` command, we could have performed this analysis by using `anova`. [Example 1](#) in [\[R\] anova](#) repeats this same analysis. You may wish to compare the output.

You will find the `oneway` command quicker than the `anova` command, and, as you will learn, `oneway` allows you to perform multiple-comparison tests. On the other hand, `anova` will let you generate predictions, examine the covariance matrix of the estimators, and perform more general hypothesis tests.





## □ Technical note

Although the output is a usual ANOVA table, let's run through it anyway. The between-group sum of squares for the model is 5295.5 with 3 degrees of freedom, resulting in a mean square of  $5295.5/3 \approx 1765.2$ . The corresponding  $F$  statistic is 21.46 and has a significance level of 0.0013. Thus the model appears to be significant at the 0.13% level.

The second line summarizes the within-group (residual) variation. The within-group sum of squares is 493.59 with 6 degrees of freedom, resulting in a mean squared error of 82.27.

The between- and residual-group variations sum to the total sum of squares (TSS), which is reported as 5789.1 in the last line of the table. This is the TSS of `weight` after removal of the mean. Similarly, the between plus residual degrees of freedom sum to the total degrees of freedom, 9. Remember that there are 10 observations. Subtracting 1 for the mean, we are left with 9 total degrees of freedom.

At the bottom of the table, Bartlett's test for equal variances is reported. The value of the statistic is 1.39. The corresponding significance level ( $\chi^2$  with 3 degrees of freedom) is 0.708, so we cannot reject the assumption that the variances are homogeneous.

□

## Obtaining observed means

### ▷ Example 2

We typed `oneway weight treatment` to obtain an ANOVA table of weight of fruit by fertilizer concentration. Although we obtained the table, we obtained no information on which fertilizer seems to work the best. If we add the `tabulate` option, we obtain that additional information:

```
. oneway weight treatment, tabulate
```

| Fertilizer | Summary of Average weight in grams |           |       |
|------------|------------------------------------|-----------|-------|
|            | Mean                               | Std. Dev. | Freq. |
| 1          | 111.9                              | 6.7535176 | 3     |
| 2          | 52.733333                          | 5.3928966 | 3     |
| 3          | 78.65                              | 11.667262 | 2     |
| 4          | 77.5                               | 14.424978 | 2     |
| Total      | 80.62                              | 25.362124 | 10    |

| Source         | Analysis of Variance |    |            | F     | Prob > F |
|----------------|----------------------|----|------------|-------|----------|
|                | SS                   | df | MS         |       |          |
| Between groups | 5295.54433           | 3  | 1765.18144 | 21.46 | 0.0013   |
| Within groups  | 493.591667           | 6  | 82.2652778 |       |          |
| Total          | 5789.136             | 9  | 643.237333 |       |          |

```
Bartlett's test for equal variances: chi2(3) = 1.3900 Prob>chi2 = 0.708
```

We find that the average weight was largest when we used fertilizer concentration 1.

◀

## Multiple-comparison tests

### ➤ Example 3

`oneway` can also perform multiple-comparison tests using either Bonferroni, Scheffé, or Šidák normalizations. For instance, to obtain the Bonferroni multiple-comparison test, we specify the `bonferroni` option:

```
. oneway weight treatment, bonferroni
```

| Source         | Analysis of Variance |    |            | F     | Prob > F |
|----------------|----------------------|----|------------|-------|----------|
|                | SS                   | df | MS         |       |          |
| Between groups | 5295.54433           | 3  | 1765.18144 | 21.46 | 0.0013   |
| Within groups  | 493.591667           | 6  | 82.2652778 |       |          |
| Total          | 5789.136             | 9  | 643.237333 |       |          |

Bartlett's test for equal variances: `chi2(3) = 1.3900 Prob>chi2 = 0.708`

Comparison of Average weight in grams by Fertilizer  
(Bonferroni)

| Row Mean-<br>Col Mean | 1                 | 2                | 3              |
|-----------------------|-------------------|------------------|----------------|
| 2                     | -59.1667<br>0.001 |                  |                |
| 3                     | -33.25<br>0.042   | 25.9167<br>0.122 |                |
| 4                     | -34.4<br>0.036    | 24.7667<br>0.146 | -1.15<br>1.000 |

The results of the Bonferroni test are presented as a matrix. The first entry,  $-59.17$ , represents the difference between fertilizer concentrations 2 and 1 (labeled “Row Mean – Col Mean” in the upper stub of the table). Remember that in the previous example we requested the `tabulate` option. Looking back, we find that the means of concentrations 1 and 2 are 111.90 and 52.73, respectively. Thus  $52.73 - 111.90 = -59.17$ .

Underneath that number is reported “0.001”. This is the Bonferroni-adjusted significance of the difference. The difference is significant at the 0.1% level. Looking down the column, we see that concentration 3 is also worse than concentration 1 (4.2% level), as is concentration 4 (3.6% level).

On the basis of this evidence, we would use concentration 1 if we grew apple trees.



### ➤ Example 4

We can just as easily obtain the Scheffé-adjusted significance levels. Rather than specifying the `bonferroni` option, we specify the `scheffe` option.

We will also add the `noanova` option to prevent Stata from redisplaying the ANOVA table:

```
. oneway weight treatment, noanova scheffe
```

Comparison of Average weight in grams by Fertilizer  
(Scheffe)

| Row Mean-<br>Col Mean | 1                 | 2                | 3              |
|-----------------------|-------------------|------------------|----------------|
| 2                     | -59.1667<br>0.001 |                  |                |
| 3                     | -33.25<br>0.039   | 25.9167<br>0.101 |                |
| 4                     | -34.4<br>0.034    | 24.7667<br>0.118 | -1.15<br>0.999 |

The differences are the same as those we obtained in the Bonferroni output, but the significance levels are not. According to the Bonferroni-adjusted numbers, the significance of the difference between fertilizer concentrations 1 and 3 is 4.2%. The Scheffé-adjusted significance level is 3.9%.

We will leave it to you to decide which results are more accurate.

◀

## ► Example 5

Let's conclude this example by obtaining the Šidák-adjusted multiple-comparison tests. We do this to illustrate Stata's capabilities to calculate these results, because searching across adjustment methods until you find the results you want is not a valid technique for obtaining significance levels.

```
. oneway weight treatment, noanova sidak
```

Comparison of Average weight in grams by Fertilizer  
(Sidak)

| Row Mean-<br>Col Mean | 1                 | 2                | 3              |
|-----------------------|-------------------|------------------|----------------|
| 2                     | -59.1667<br>0.001 |                  |                |
| 3                     | -33.25<br>0.041   | 25.9167<br>0.116 |                |
| 4                     | -34.4<br>0.035    | 24.7667<br>0.137 | -1.15<br>1.000 |

We find results that are similar to the Bonferroni-adjusted numbers.

◀

Henry Scheffé (1907–1977) was born in New York. He studied mathematics at the University of Wisconsin, gaining a doctorate with a dissertation on differential equations. He taught mathematics at Wisconsin, Oregon State University, and Reed College, but his interests changed to statistics and he joined Wilks at Princeton. After periods at Syracuse, UCLA, and Columbia, Scheffé settled in Berkeley from 1953. His research increasingly focused on linear models and particularly ANOVA, on which he produced a celebrated monograph. His death was the result of a bicycle accident.

## Weighted data

### ► Example 6

`oneway` can work with both weighted and unweighted data. Let’s assume that we wish to perform a one-way layout of the death rate on the four census regions of the United States using state data. Our data contain three variables, `drate` (the death rate), `region` (the region), and `pop` (the population of the state).

To fit the model, we type `oneway drate region [weight=pop]`, although we typically abbreviate `weight` as `w`. We will also add the `tabulate` option to demonstrate how the table of summary statistics differs for weighted data:

```
. use http://www.stata-press.com/data/r11/census8
(1980 Census data by state)
. oneway drate region [w=pop], tabulate
(analytic weights assumed)
```

| Census<br>region | Summary of Death Rate |           |           | Obs. |
|------------------|-----------------------|-----------|-----------|------|
|                  | Mean                  | Std. Dev. | Freq.     |      |
| NE               | 97.15                 | 5.82      | 49135283  | 9    |
| N Cntrl          | 88.10                 | 5.58      | 58865670  | 12   |
| South            | 87.05                 | 10.40     | 74734029  | 16   |
| West             | 75.65                 | 8.23      | 43172490  | 13   |
| Total            | 87.34                 | 10.43     | 2.259e+08 | 50   |

| Source         | Analysis of Variance |    |            | F     | Prob > F |
|----------------|----------------------|----|------------|-------|----------|
|                | SS                   | df | MS         |       |          |
| Between groups | 2360.92281           | 3  | 786.974272 | 12.17 | 0.0000   |
| Within groups  | 2974.09635           | 46 | 64.6542685 |       |          |
| Total          | 5335.01916           | 49 | 108.877942 |       |          |

Bartlett’s test for equal variances: `chi2(3) = 5.4971 Prob>chi2 = 0.139`

When the data are weighted, the summary table has four columns rather than three. The column labeled “`Freq.`” reports the sum of the weights. The overall frequency is  $2.259 \times 10^8$ , meaning that there are approximately 226 million people in the United States.

The ANOVA table is appropriately weighted. Also see [U] 11.1.6 [weight](#).



## Saved results

`oneway` saves the following in `r()`:

|                      |                                 |                          |                                  |
|----------------------|---------------------------------|--------------------------|----------------------------------|
| Scalars              |                                 |                          |                                  |
| <code>r(N)</code>    | number of observations          | <code>r(df_m)</code>     | between-group degrees of freedom |
| <code>r(F)</code>    | <i>F</i> statistic              | <code>r(rss)</code>      | within-group sum of squares      |
| <code>r(df_r)</code> | within-group degrees of freedom | <code>r(chi2bart)</code> | Bartlett’s $\chi^2$              |
| <code>r(mss)</code>  | between-group sum of squares    | <code>r(df_bart)</code>  | Bartlett’s degrees of freedom    |

## Methods and formulas

Methods and formulas are presented under the following headings:

*One-way analysis of variance*

*Bartlett's test*

*Multiple-comparison tests*

## One-way analysis of variance

The model of one-way ANOVA is

$$y_{ij} = \mu + \alpha_i + \epsilon_{ij}$$

for levels  $i = 1, \dots, k$  and observations  $j = 1, \dots, n_i$ . Define  $\bar{y}_i$  as the (weighted) mean of  $y_{ij}$  over  $j$  and  $\bar{y}$  as the overall (weighted) mean of  $y_{ij}$ . Define  $w_{ij}$  as the weight associated with  $y_{ij}$ , which is 1 if the data are unweighted.  $w_{ij}$  is normalized to sum to  $n = \sum_i n_i$  if `aweights` are used and is otherwise not normalized.  $w_i$  refers to  $\sum_j w_{ij}$ , and  $w$  refers to  $\sum_i w_i$ .

The between-group sum of squares is then

$$S_1 = \sum_i w_i (\bar{y}_i - \bar{y})^2$$

The TSS is

$$S = \sum_i \sum_j w_{ij} (y_{ij} - \bar{y})^2$$

The within-group sum of squares is given by  $S_e = S - S_1$ .

The between-group mean square is  $s_1^2 = S_1/(k-1)$ , and the within-group mean square is  $s_e^2 = S_e/(w-k)$ . The test statistic is  $F = s_1^2/s_e^2$ . See, for instance, [Snedecor and Cochran \(1989\)](#).

## Bartlett's test

Bartlett's test assumes that you have  $m$  independent, normal, random samples and tests the hypothesis  $\sigma_1^2 = \sigma_2^2 = \dots = \sigma_m^2$ . The test statistic,  $M$ , is defined as

$$M = \frac{(T-m) \ln \hat{\sigma}^2 - \sum (T_i-1) \ln \hat{\sigma}_i^2}{1 + \frac{1}{3(m-1)} \left\{ \left( \sum \frac{1}{T_i-1} \right) - \frac{1}{T-m} \right\}}$$

where there are  $T$  overall observations,  $T_i$  observations in the  $i$ th group, and

$$(T_i-1) \hat{\sigma}_i^2 = \sum_{j=1}^{T_i} (y_{ij} - \bar{y}_i)^2$$

$$(T-m) \hat{\sigma}^2 = \sum_{i=1}^m (T_i-1) \hat{\sigma}_i^2$$

An approximate test of the homogeneity of variance is based on the statistic  $M$  with critical values obtained from the  $\chi^2$  distribution of  $m-1$  degrees of freedom. See [Bartlett \(1937\)](#) or [Draper and Smith \(1998, 56–57\)](#).

## Multiple-comparison tests

Let's begin by reviewing the logic behind these adjustments. The “standard”  $t$  statistic for the comparison of two means is

$$t = \frac{\bar{y}_i - \bar{y}_j}{s \sqrt{\frac{1}{n_i} + \frac{1}{n_j}}}$$

where  $s$  is the overall standard deviation,  $\bar{y}_i$  is the measured average of  $y$  in group  $i$ , and  $n_i$  is the number of observations in the group. We perform hypothesis tests by calculating this  $t$  statistic. We simultaneously choose a critical level,  $\alpha$ , and look up the  $t$  statistic corresponding to that level in a table. We reject the hypothesis if our calculated  $t$  exceeds the value we looked up. Alternatively, because we have a computer at our disposal, we calculate the significance level  $e$  corresponding to our calculated  $t$  statistic, and if  $e < \alpha$ , we reject the hypothesis.

This logic works well when we are performing one test. Now consider what happens when we perform several separate tests, say,  $n$  of them. Let's assume, just for discussion, that we set  $\alpha$  equal to 0.05 and that we will perform six tests. For each test, we have a 0.05 probability of falsely rejecting the equality-of-means hypothesis. Overall, then, our chances of falsely rejecting *at least one* of the hypotheses is  $1 - (1 - 0.05)^6 \approx 0.26$  if the tests are independent.

The idea behind multiple-comparison tests is to control for the fact that we will perform multiple tests and to reduce our overall chances of falsely rejecting each hypothesis to  $\alpha$  rather than letting our chances increase with each additional test. (See [Miller Jr. \[1981\]](#) and [Hochberg and Tamhane \[1987\]](#) for rather advanced texts on multiple-comparison procedures.)

The Bonferroni adjustment (see [Miller Jr. \[1981\]](#); also see [van Belle et al. \[2004, 534–537\]](#)) does this by (falsely but approximately) asserting that the critical level we should use,  $a$ , is the true critical level,  $\alpha$ , divided by the number of tests,  $n$ ; that is,  $a = \alpha/n$ . For instance, if we are going to perform six tests, each at the 0.05 significance level, we want to adopt a critical level of  $0.05/6 \approx 0.00833$ .

We can just as easily apply this logic to  $e$ , the significance level associated with our  $t$  statistic, as to our critical level  $\alpha$ . If a comparison has a calculated significance of  $e$ , then its “real” significance, adjusted for the fact of  $n$  comparisons, is  $n \times e$ . If a comparison has a significance level of, say, 0.012, and we perform six tests, then its “real” significance is 0.072. If we adopt a critical level of 0.05, we cannot reject the hypothesis. If we adopt a critical level of 0.10, we can reject it.

Of course, this calculation can go above 1, but that just means that there is no  $\alpha < 1$  for which we could reject the hypothesis. (This situation arises because of the crude nature of the Bonferroni adjustment.) Stata handles this case by simply calling the significance level 1. Thus the formula for the Bonferroni significance level is

$$e_b = \min(1, en)$$

where  $n = k(k-1)/2$  is the number of comparisons.

The Šidák adjustment ([Šidák \[1967\]](#); also see [Winer, Brown, and Michels \[1991, 165–166\]](#)) is slightly different and provides a tighter bound. It starts with the assertion that

$$a = 1 - (1 - \alpha)^{1/n}$$

Turning this formula around and substituting calculated significance levels, we obtain

$$e_s = \min\left\{1, 1 - (1 - e)^n\right\}$$

For example, if the calculated significance is 0.012 and we perform six tests, the “real” significance is approximately 0.07.

The Scheffé test (Scheffé [1953, 1959]; also see Kuehl [2000, 97–98]) differs in derivation, but it attacks the same problem. Let there be  $k$  means for which we want to make all the pairwise tests. Two means are declared significantly different if

$$t \geq \sqrt{(k-1)F(\alpha; k-1, \nu)}$$

where  $F(\alpha; k-1, \nu)$  is the  $\alpha$ -critical value of the  $F$  distribution with  $k-1$  numerator and  $\nu$  denominator degrees of freedom. Scheffé's test has the nicety that it never declares a contrast significant if the overall  $F$  test is not significant.

Turning the test around, Stata calculates a significance level

$$\hat{e} = F\left(\frac{t^2}{k-1}, k-1, \nu\right)$$

For instance, you have a calculated  $t$  statistic of 4.0 with 50 degrees of freedom. The simple  $t$  test says that the significance level is 0.00021. The  $F$  test equivalent, 16 with 1 and 50 degrees of freedom, says the same. If you are comparing three means, however, you calculate an  $F$  test of 8.0 with 2 and 50 degrees of freedom, which says that the significance level is 0.0010.

## References

- Acocck, A. C. 2008. *A Gentle Introduction to Stata*. 2nd ed. College Station, TX: Stata Press.
- Altman, D. G. 1991. *Practical Statistics for Medical Research*. London: Chapman & Hall/CRC.
- Bartlett, M. S. 1937. Properties of sufficiency and statistical tests. *Proceedings of the Royal Society, Series A* 160: 268–282.
- Daniel, C., and E. L. Lehmann. 1979. Henry Scheffé 1907–1977. *Annals of Statistics* 7: 1149–1161.
- Draper, N., and H. Smith. 1998. *Applied Regression Analysis*. 3rd ed. New York: Wiley.
- Hochberg, Y., and A. C. Tamhane. 1987. *Multiple Comparison Procedures*. New York: Wiley.
- Kuehl, R. O. 2000. *Design of Experiments: Statistical Principles of Research Design and Analysis*. 2nd ed. Belmont, CA: Duxbury.
- Marchenko, Y. 2006. Estimating variance components in Stata. *Stata Journal* 6: 1–21.
- Miller Jr., R. G. 1981. *Simultaneous Statistical Inference*. 2nd ed. New York: Springer.
- Scheffé, H. 1953. A method for judging all contrasts in the analysis of variance. *Biometrika* 40: 87–104.
- . 1959. *The Analysis of Variance*. New York: Wiley.
- Šidák, Z. 1967. Rectangular confidence regions for the means of multivariate normal distributions. *Journal of the American Statistical Association* 62: 626–633.
- Snedecor, G. W., and W. G. Cochran. 1989. *Statistical Methods*. 8th ed. Ames, IA: Iowa State University Press.
- van Belle, G., L. D. Fisher, P. J. Heagerty, and T. S. Lumley. 2004. *Biostatistics: A Methodology for the Health Sciences*. 2nd ed. New York: Wiley.
- Winer, B. J., D. R. Brown, and K. M. Michels. 1991. *Statistical Principles in Experimental Design*. 3rd ed. New York: McGraw–Hill.

## Also see

[R] [anova](#) — Analysis of variance and covariance

[R] [loneway](#) — Large one-way ANOVA, random effects, and reliability

Syntax

```
oprobit depvar [indepvars] [if] [in] [weight] [, options]
```

| <i>options</i>                               | description                                                                                                                                              |
|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                        |                                                                                                                                                          |
| <code>offset(<i>varname</i>)</code>          | include <i>varname</i> in model with coefficient constrained to 1                                                                                        |
| <code>constraints(<i>constraints</i>)</code> | apply specified linear constraints                                                                                                                       |
| <code>collinear</code>                       | keep collinear variables                                                                                                                                 |
| SE/Robust                                    |                                                                                                                                                          |
| <code>vce(<i>vcetype</i>)</code>             | <i>vcetype</i> may be <code>oim</code> , <code>robust</code> , <code>cluster <i>clustvar</i></code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                                    |                                                                                                                                                          |
| <code>level(#)</code>                        | set confidence level; default is <code>level(95)</code>                                                                                                  |
| <code>nocnsreport</code>                     | do not display constraints                                                                                                                               |
| <code>display_options</code>                 | control spacing and display of omitted variables and base and empty cells                                                                                |
| Maximization                                 |                                                                                                                                                          |
| <code>maximize_options</code>                | control the maximization process; seldom used                                                                                                            |
| <sup>†</sup> <code>coeflegend</code>         | display coefficients' legend instead of coefficient table                                                                                                |

<sup>†</sup>`coeflegend` does not appear in the dialog box.

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

`bootstrap`, `by`, `frapoly`, `jackknife`, `mfp`, `mi estimate`, `nestreg`, `rolling`, `statsby`, `stepwise`, and `svy` are allowed; see [U] 11.1.10 Prefix commands.

`vce(bootstrap)` and `vce(jackknife)` are not allowed with the `mi estimate` prefix.

Weights are not allowed with the `bootstrap` prefix.

`vce()` and weights are not allowed with the `svy` prefix.

`fweights`, `iweights`, and `pweights` are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Ordinal outcomes > Ordered probit regression

Description

`oprobit` fits ordered probit models of ordinal variable *depvar* on the independent variables *indepvars*. The actual values taken on by the dependent variable are irrelevant, except that larger values are assumed to correspond to “higher” outcomes.

See [R] **logistic** for a list of related estimation commands.



## Options

### Model

`offset(varname)`, `constraints(constraints)`, `collinear`; see [\[R\] estimation options](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [\[R\] vce\\_option](#).

### Reporting

`level(#)`; see [\[R\] estimation options](#).

`nocnsreport`; see [\[R\] estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [\[R\] estimation options](#).

### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [\[R\] maximize](#).

The following option is available with `oprobit` but is not shown in the dialog box:

`coeflegend`; see [\[R\] estimation options](#).

## Remarks

An ordered probit model is used to estimate relationships between an ordinal dependent variable and a set of independent variables. An *ordinal* variable is a variable that is categorical and ordered, for instance, “poor”, “good”, and “excellent”, which might indicate a person’s current health status or the repair record of a car. If there are only two outcomes, see [\[R\] logistic](#), [\[R\] logit](#), and [\[R\] probit](#). This entry is concerned only with more than two outcomes. If the outcomes cannot be ordered (e.g., residency in the north, east, south, or west), see [\[R\] mlogit](#). This entry is concerned only with models in which the outcomes can be ordered.

In ordered probit, an underlying score is estimated as a linear function of the independent variables and a set of cutpoints. The probability of observing outcome  $i$  corresponds to the probability that the estimated linear function, plus random error, is within the range of the cutpoints estimated for the outcome:

$$\Pr(\text{outcome}_j = i) = \Pr(\kappa_{i-1} < \beta_1 x_{1j} + \beta_2 x_{2j} + \cdots + \beta_k x_{kj} + u_j \leq \kappa_i)$$

$u_j$  is assumed to be normally distributed. In either case, we estimate the coefficients  $\beta_1, \beta_2, \dots, \beta_k$  together with the cutpoints  $\kappa_1, \kappa_2, \dots, \kappa_{I-1}$ , where  $I$  is the number of possible outcomes.  $\kappa_0$  is taken as  $-\infty$ , and  $\kappa_I$  is taken as  $+\infty$ . All of this is a direct generalization of the ordinary two-outcome probit model.

➤ Example 1

In [example 2](#) of [\[R\] ologit](#), we use a variation of the automobile dataset (see [\[U\] 1.2.2 Example datasets](#)) to analyze the 1977 repair records of 66 foreign and domestic cars. We use ordered logit to explore the relationship of `rep77` in terms of `foreign` (origin of manufacture), `length` (a proxy for size), and `mpg`. Here we fit the same model using ordered probit rather than ordered logit:

```
. use http://www.stata-press.com/data/r11/fullauto
(Automobile Models)
. oprobit rep77 foreign length mpg

Iteration 0: log likelihood = -89.895098
Iteration 1: log likelihood = -78.106316
Iteration 2: log likelihood = -78.020086
Iteration 3: log likelihood = -78.020025
Iteration 4: log likelihood = -78.020025

Ordered probit regression Number of obs = 66
 LR chi2(3) = 23.75
 Prob > chi2 = 0.0000
Log likelihood = -78.020025 Pseudo R2 = 0.1321
```

| rep77   | Coef.    | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|---------|----------|-----------|------|-------|----------------------|----------|
| foreign | 1.704861 | .4246796  | 4.01 | 0.000 | .8725037             | 2.537217 |
| length  | .0468675 | .012648   | 3.71 | 0.000 | .022078              | .0716571 |
| mpg     | .1304559 | .0378628  | 3.45 | 0.001 | .0562463             | .2046656 |
| /cut1   | 10.1589  | 3.076754  |      |       | 4.128577             | 16.18923 |
| /cut2   | 11.21003 | 3.107527  |      |       | 5.119389             | 17.30067 |
| /cut3   | 12.54561 | 3.155233  |      |       | 6.361467             | 18.72975 |
| /cut4   | 13.98059 | 3.218793  |      |       | 7.671874             | 20.28931 |

We find that foreign cars have better repair records, as do larger cars and cars with better mileage ratings.



## Saved results

oprobit saves the following in `e()`:

### Scalars

|                            |                                                |
|----------------------------|------------------------------------------------|
| <code>e(N)</code>          | number of observations                         |
| <code>e(N_cd)</code>       | number of completely determined observations   |
| <code>e(k_cat)</code>      | number of categories                           |
| <code>e(k)</code>          | number of parameters                           |
| <code>e(k_aux)</code>      | number of auxiliary parameters                 |
| <code>e(k_eq)</code>       | number of equations                            |
| <code>e(k_eq_model)</code> | number of equations in model Wald test         |
| <code>e(k_dv)</code>       | number of dependent variables                  |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints |
| <code>e(df_m)</code>       | model degrees of freedom                       |
| <code>e(r2_p)</code>       | pseudo- <i>R</i> -squared                      |
| <code>e(ll)</code>         | log likelihood                                 |
| <code>e(ll_0)</code>       | log likelihood, constant-only model            |
| <code>e(N_clust)</code>    | number of clusters                             |
| <code>e(chi2)</code>       | $\chi^2$                                       |
| <code>e(p)</code>          | significance of model test                     |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                      |
| <code>e(ic)</code>         | number of iterations                           |
| <code>e(rc)</code>         | return code                                    |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                    |

### Macros

|                                 |                                                                          |
|---------------------------------|--------------------------------------------------------------------------|
| <code>e(cmd)</code>             | oprobit                                                                  |
| <code>e(cmdline)</code>         | command as typed                                                         |
| <code>e(depvar)</code>          | name of dependent variable                                               |
| <code>e(wtype)</code>           | weight type                                                              |
| <code>e(wexp)</code>            | weight expression                                                        |
| <code>e(title)</code>           | title in estimation output                                               |
| <code>e(clustvar)</code>        | name of cluster variable                                                 |
| <code>e(offset)</code>          | offset                                                                   |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                  |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                           |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                            |
| <code>e(opt)</code>             | type of optimization                                                     |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>       | type of ml method                                                        |
| <code>e(user)</code>            | name of likelihood-evaluator program                                     |
| <code>e(technique)</code>       | maximization technique                                                   |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular              |
| <code>e(crittype)</code>        | optimization criterion                                                   |
| <code>e(properties)</code>      | b V                                                                      |
| <code>e(predict)</code>         | program used to implement predict                                        |
| <code>e(marginsprop)</code>     | signals to the margins command                                           |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>           |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>           |

## Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(gradient)</code>     | gradient vector                              |
| <code>e(cat)</code>          | category values                              |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`oprobit` is implemented as an ado-file.

See [Methods and formulas](#) of [R] **ologit**.

## References

- Aitchison, J., and S. D. Silvey. 1957. The generalization of probit analysis to the case of multiple responses. *Biometrika* 44: 131–140.
- Cameron, A. C., and P. K. Trivedi. 2005. *Microeconometrics: Methods and Applications*. New York: Cambridge University Press.
- Chiburis, R., and M. Lokshin. 2007. [Maximum likelihood and two-step estimation of an ordered-probit selection model](#). *Stata Journal* 7: 167–182.
- Goldstein, R. 1997. [sg59: Index of ordinal variation and Neyman–Barton GOF](#). *Stata Technical Bulletin* 33: 10–12. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 145–147. College Station, TX: Stata Press.
- Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.
- Long, J. S., and J. Freese. 2006. *Regression Models for Categorical Dependent Variables Using Stata*. 2nd ed. College Station, TX: Stata Press.
- Miranda, A., and S. Rabe-Hesketh. 2006. [Maximum likelihood estimation of endogenous switching and sample selection models for binary, ordinal, and count variables](#). *Stata Journal* 6: 285–308.
- Stewart, M. B. 2004. [Semi-nonparametric estimation of extended ordered probit models](#). *Stata Journal* 4: 27–39.
- Wolfe, R. 1998. [sg86: Continuation-ratio models for ordinal response data](#). *Stata Technical Bulletin* 44: 18–21. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 149–153. College Station, TX: Stata Press.
- Wolfe, R., and W. W. Gould. 1998. [sg76: An approximate likelihood-ratio test for ordinal response models](#). *Stata Technical Bulletin* 42: 24–27. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 199–204. College Station, TX: Stata Press.
- Xu, J., and J. S. Long. 2005. [Confidence intervals for predicted outcomes in regression models for categorical outcomes](#). *Stata Journal* 5: 537–559.

## Also see

[R] [oprobit postestimation](#) — Postestimation tools for oprobit

[R] [logistic](#) — Logistic regression, reporting odds ratios

[R] [mlogit](#) — Multinomial (polytomous) logistic regression

[R] [mprobit](#) — Multinomial probit regression

[R] [ologit](#) — Ordered logistic regression

[R] [probit](#) — Probit regression

[SVY] [svy estimation](#) — Estimation commands for survey data

[U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `oprobit`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>            | link test for model specification                                                                   |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic outcome(outcome) nooffset]

predict [type] { stub*|newvarreg newvarκ1 ... newvarκk-1 } [if] [in] , scores
```

where  $k$  is the number of outcomes in the model.

| statistic         | description                             |
|-------------------|-----------------------------------------|
| Main              |                                         |
| <code>pr</code>   | predicted probabilities; the default    |
| <code>xb</code>   | linear prediction                       |
| <code>stdp</code> | standard error of the linear prediction |

If you do not specify `outcome()`, `pr` (with one new variable specified) assumes `outcome(#1)`.  
You specify one or  $k$  new variables with `pr`, where  $k$  is the number of outcomes.  
You specify one new variable with `xb` and `stdp`.  
These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

**pr**, the default, calculates the predicted probabilities. If you do not also specify the `outcome()` option, you must specify  $k$  new variables, where  $k$  is the number of categories of the dependent variable. Say that you fit a model by typing `oprobit result x1 x2`, and `result` takes on three values. Then you could type `predict p1 p2 p3` to obtain all three predicted probabilities. If you specify the `outcome()` option, you must specify one new variable. Say that `result` takes on values 1, 2, and 3. Typing `predict p1, outcome(1)` would produce the same `p1`.

**xb** calculates the linear prediction. You specify one new variable, for example, `predict linear, xb`. The linear prediction is defined ignoring the contribution of the estimated cutpoints.

**stdp** calculates the standard error of the linear prediction. You specify one new variable, for example, `predict se, stdp`.

**outcome(outcome)** specifies for which outcome the predicted probabilities are to be calculated. `outcome()` should contain either one value of the dependent variable or one of `#1`, `#2`, ..., with `#1` meaning the first category of the dependent variable, `#2` meaning the second category, etc.

**nooffset** is relevant only if you specified `offset(varname)` for `oprobit`. It modifies the calculations made by `predict` so that they ignore the offset variable; the linear prediction is treated as  $\mathbf{x}_j\mathbf{b}$  rather than as  $\mathbf{x}_j\mathbf{b} + \text{offset}_j$ .

**scores** calculates equation-level score variables. The number of score variables created will equal the number of outcomes in the model. If the number of outcomes in the model was  $k$ , then

the first new variable will contain  $\partial \ln L / \partial (\mathbf{x}_j\mathbf{b})$ ;

the second new variable will contain  $\partial \ln L / \partial \kappa_1$ ;

the third new variable will contain  $\partial \ln L / \partial \kappa_2$ ;

...

and the  $k$ th new variable will contain  $\partial \ln L / \partial \kappa_{k-1}$ , where  $\kappa_i$  refers to the  $i$ th cutpoint.

## Remarks

See [\[U\] 20 Estimation and postestimation commands](#) for instructions on obtaining the variance–covariance matrix of the estimators, predicted values, and hypothesis tests. Also see [\[R\] lrtest](#) for performing likelihood-ratio tests.

### ► Example 1

In [example 1](#) of [\[R\] oprobit](#), we fit the model `oprobit rep77 foreign length mpg`. The `predict` command can be used to obtain the predicted probabilities. We type `predict` followed by the names of the new variables to hold the predicted probabilities, ordering the names from low to high. In our data, the lowest outcome is “poor” and the highest is “excellent”. We have five categories, so we must type five names following `predict`; the choice of names is up to us:

```
. predict poor fair avg good exc
(option pr assumed; predicted probabilities)
. list make model exc good if rep77>=., sep(4) divider
```

|     | make    | model   | exc      | good     |
|-----|---------|---------|----------|----------|
| 3.  | AMC     | Spirit  | .0006044 | .0351813 |
| 10. | Buick   | Opel    | .0043803 | .1133763 |
| 32. | Ford    | Fiesta  | .0002927 | .0222789 |
| 44. | Merc.   | Monarch | .0093209 | .1700846 |
| 53. | Peugeot | 604     | .0734199 | .4202766 |
| 56. | Plym.   | Horizon | .001413  | .0590294 |
| 57. | Plym.   | Sapporo | .0197543 | .2466034 |
| 63. | Pont.   | Phoenix | .0234156 | .266771  |



□ Technical note

For ordered probit, `predict, xb` produces  $S_j = x_{1j}\beta_1 + x_{2j}\beta_2 + \cdots + x_{kj}\beta_k$ . Ordered probit is identical to ordered logit, except that we use different distribution functions for calculating probabilities. The ordered-probit predictions are then the probability that  $S_j + u_j$  lies between a pair of cutpoints  $\kappa_{i-1}$  and  $\kappa_i$ . The formulas for ordered probit are

$$\begin{aligned}\Pr(S_j + u < \kappa) &= \Phi(\kappa - S_j) \\ \Pr(S_j + u > \kappa) &= 1 - \Phi(\kappa - S_j) = \Phi(S_j - \kappa) \\ \Pr(\kappa_1 < S_j + u < \kappa_2) &= \Phi(\kappa_2 - S_j) - \Phi(\kappa_1 - S_j)\end{aligned}$$

Rather than using `predict` directly, we could calculate the predicted probabilities by hand.

```
. predict pscore, xb
. generate probexc = normal(pscore-_b[/cut4])
. generate probgood = normal(_b[/cut4]-pscore) - normal(_b[/cut3]-pscore)
```



Methods and formulas

All postestimation tools listed above are implemented as ado-files.

Also see

- [R] [oprobit](#) — Ordered probit regression
- [U] [20 Estimation and postestimation commands](#)



## Syntax

*Orthogonalize variables*

```
orthog [varlist] [if] [in] [weight], generate(newvarlist) [matrix(matname)]
```

*Compute orthogonal polynomial*

```
orthpoly varname [if] [in] [weight],
{ generate(newvarlist) | poly(matname) } [degree(#)]
```

`orthpoly` requires that `generate(newvarlist)` or `poly(matname)`, or both, be specified.

*varlist* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*iweights*, *fweights*, *pweights*, and *aweights* are allowed, see [U] 11.1.6 weight.

## Menu

### orthog

Data > Create or change data > Other variable-creation commands > Orthogonalize variables

### orthpoly

Data > Create or change data > Other variable-creation commands > Orthogonal polynomials

## Description

`orthog` orthogonalizes a set of variables, creating a new set of orthogonal variables (all of type `double`), using a modified Gram–Schmidt procedure (Golub and Van Loan 1996). The order of the variables determines the orthogonalization; hence, the “most important” variables should be listed first.

Execution time is proportional to the square of the number of variables. With many (>10) variables, `orthog` will be fairly slow.

`orthpoly` computes orthogonal polynomials for one variable.

## Options for orthog

### Main

`generate(newvarlist)` is required. `generate()` creates new orthogonal variables of type `double`.

For `orthog`, *newvarlist* will contain the orthogonalized *varlist*. If *varlist* contains *d* variables, then so will *newvarlist*. *newvarlist* can be specified by giving a list of exactly *d* new variable names, or it can be abbreviated using the styles *newvar1–newvard* or *newvar\**. For these two styles of abbreviation, new variables *newvar1*, *newvar2*, . . . , *newvard* are generated.

`matrix(matname)` creates a  $(d+1) \times (d+1)$  matrix containing the matrix  $R$  defined by  $X = QR$ , where  $X$  is the  $N \times (d+1)$  matrix representation of *varlist* plus a column of ones and  $Q$  is the  $N \times (d+1)$  matrix representation of *newvarlist* plus a column of ones ( $d$  = number of variables in *varlist*, and  $N$  = number of observations).

## Options for orthopoly

### Main

`generate(newvarlist)` or `poly()`, or both, must be specified. `generate()` creates new orthogonal variables of type `double`. *newvarlist* will contain orthogonal polynomials of degree 1, 2, ...,  $d$  evaluated at *varname*, where  $d$  is as specified by `degree(d)`. *newvarlist* can be specified by giving a list of exactly  $d$  new variable names, or it can be abbreviated using the styles *newvar1-newvard* or *newvar\**. For these two styles of abbreviation, new variables *newvar1*, *newvar2*, ..., *newvard* are generated.

`poly(matname)` creates a  $(d+1) \times (d+1)$  matrix called *matname* containing the coefficients of the orthogonal polynomials. The orthogonal polynomial of degree  $i \leq d$  is

$$\text{matname}[i, d+1] + \text{matname}[i, 1]*\text{varname} + \text{matname}[i, 2]*\text{varname}^2 + \dots + \text{matname}[i, i]*\text{varname}^i$$

The coefficients corresponding to the constant term are placed in the last column of the matrix. The last row of the matrix is all zeros, except for the last column, which corresponds to the constant term.

`degree(#)` specifies the highest-degree polynomial to include. Orthogonal polynomials of degree 1, 2, ...,  $d = \#$  are computed. The default is  $d = 1$ .

## Remarks

Orthogonal variables are useful for two reasons. The first is numerical accuracy for highly collinear variables. Stata's `regress` and other estimation commands can face much collinearity and still produce accurate results. But, at some point, these commands will drop variables because of collinearity. If you know with certainty that the variables are not perfectly collinear, you may want to retain all their effects in the model. If you use `orthog` or `orthpoly` to produce a set of orthogonal variables, all variables will be present in the estimation results.

Users are more likely to find orthogonal variables useful for the second reason: ease of interpreting results. `orthog` and `orthpoly` create a set of variables such that the “effects” of all the preceding variables have been removed from each variable. For example, if we issue the command

```
. orthog x1 x2 x3, generate(q1 q2 q3)
```

the effect of the constant is removed from *x1* to produce *q1*; the constant and *x1* are removed from *x2* to produce *q2*; and finally the constant, *x1*, and *x2* are removed from *x3* to produce *q3*. Hence,

$$q1 = r_{01} + r_{11} x1$$

$$q2 = r_{02} + r_{12} x1 + r_{22} x2$$

$$q3 = r_{03} + r_{13} x1 + r_{23} x2 + r_{33} x3$$

This effect can be generalized and written in matrix notation as

$$X = QR$$

where  $X$  is the  $N \times (d + 1)$  matrix representation of *varlist* plus a column of ones, and  $Q$  is the  $N \times (d + 1)$  matrix representation of *newvarlist* plus a column of ones ( $d$  = number of variables in *varlist* and  $N$  = number of observations). The  $(d + 1) \times (d + 1)$  matrix  $R$  is a permuted upper-triangular matrix, i.e.,  $R$  would be upper triangular if the constant were first, but the constant is last, so the first row/column has been permuted with the last row/column. Because Stata's estimation commands list the constant term last, this allows  $R$ , obtained via the `matrix()` option, to be used to transform estimation results.

## ► Example 1

Consider Stata's `auto.dta` dataset. Suppose that we postulate a model in which `price` depends on the car's `length`, `weight`, `headroom`, and trunk size (`trunk`). These predictors are collinear, but not extremely so—the correlations are not that close to 1:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. correlate length weight headroom trunk
(obs=74)
```

|          | length | weight | headroom | trunk  |
|----------|--------|--------|----------|--------|
| length   | 1.0000 |        |          |        |
| weight   | 0.9460 | 1.0000 |          |        |
| headroom | 0.5163 | 0.4835 | 1.0000   |        |
| trunk    | 0.7266 | 0.6722 | 0.6620   | 1.0000 |

`regress` certainly has no trouble fitting this model:

```
. regress price length weight headroom trunk
```

| Source   | SS        | df | MS         |
|----------|-----------|----|------------|
| Model    | 236016580 | 4  | 59004145   |
| Residual | 399048816 | 69 | 5783316.17 |
| Total    | 635065396 | 73 | 8699525.97 |

```
Number of obs = 74
F(4, 69) = 10.20
Prob > F = 0.0000
R-squared = 0.3716
Adj R-squared = 0.3352
Root MSE = 2404.9
```

| price    | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| length   | -101.7092 | 42.12534  | -2.41 | 0.018 | -185.747             | -17.67147 |
| weight   | 4.753066  | 1.120054  | 4.24  | 0.000 | 2.518619             | 6.987512  |
| headroom | -711.5679 | 445.0204  | -1.60 | 0.114 | -1599.359            | 176.2236  |
| trunk    | 114.0859  | 109.9488  | 1.04  | 0.303 | -105.2559            | 333.4277  |
| _cons    | 11488.47  | 4543.902  | 2.53  | 0.014 | 2423.638             | 20553.31  |

However, we may believe a priori that `length` is the most important predictor, followed by `weight`, `headroom`, and `trunk`. We would like to remove the “effect” of `length` from all the other predictors, remove `weight` from `headroom` and `trunk`, and remove `headroom` from `trunk`. We can do this by running `orthog`, and then we fit the model again using the orthogonal variables:

(Continued on next page)

```
. orthog length weight headroom trunk, gen(olength oweight oheadroom otrunk)
> matrix(R)

. regress price olength oweight oheadroom otrunk
```

| Source   | SS        | df | MS         |                 |        |  |
|----------|-----------|----|------------|-----------------|--------|--|
| Model    | 236016580 | 4  | 59004145   | Number of obs = | 74     |  |
| Residual | 399048816 | 69 | 5783316.17 | F( 4, 69) =     | 10.20  |  |
|          |           |    |            | Prob > F =      | 0.0000 |  |
|          |           |    |            | R-squared =     | 0.3716 |  |
|          |           |    |            | Adj R-squared = | 0.3352 |  |
| Total    | 635065396 | 73 | 8699525.97 | Root MSE =      | 2404.9 |  |

| price     | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|-----------|-----------|-----------|-------|-------|----------------------|----------|
| olength   | 1265.049  | 279.5584  | 4.53  | 0.000 | 707.3454             | 1822.753 |
| oweight   | 1175.765  | 279.5584  | 4.21  | 0.000 | 618.0617             | 1733.469 |
| oheadroom | -349.9916 | 279.5584  | -1.25 | 0.215 | -907.6955            | 207.7122 |
| otrunk    | 290.0776  | 279.5584  | 1.04  | 0.303 | -267.6262            | 847.7815 |
| _cons     | 6165.257  | 279.5584  | 22.05 | 0.000 | 5607.553             | 6722.961 |

Using the matrix *R*, we can transform the results obtained using the orthogonal predictors back to the metric of original predictors:

```
. matrix b = e(b)*inv(R)'
. matrix list b
b[1,5]
 length weight headroom trunk _cons
y1 -101.70924 4.7530659 -711.56789 114.08591 11488.475
```



□ Technical note

The matrix *R* obtained using the `matrix()` option with `orthog` can also be used to recover *X* (the original *varlist*) from *Q* (the orthogonalized *newvarlist*), one variable at a time. Continuing with the previous example, we illustrate how to recover the `trunk` variable:

```
. matrix C = R[1..., "trunk"]'
. matrix score double rtrunk = C
. compare rtrunk trunk
```

|                 | count | minimum  | average  | maximum  |
|-----------------|-------|----------|----------|----------|
| rtrunk>trunk    | 74    | 1.42e-14 | 2.27e-14 | 3.55e-14 |
| jointly defined | 74    | 1.42e-14 | 2.27e-14 | 3.55e-14 |
| total           | 74    |          |          |          |

Here the recovered variable `rtrunk` is almost exactly the same as the original `trunk` variable. When you are orthogonalizing many variables, this procedure can be performed to check the numerical soundness of the orthogonalization. Because of the ordering of the orthogonalization procedure, the last variable and the variables near the end of the *varlist* are the most important ones to check.



The `orthpoly` command effectively does for polynomial terms what the `orthog` command does for an arbitrary set of variables.

## ► Example 2

Again consider the `auto.dta` dataset. Suppose that we wish to fit the model

$$\text{mpg} = \beta_0 + \beta_1 \text{weight} + \beta_2 \text{weight}^2 + \beta_3 \text{weight}^3 + \beta_4 \text{weight}^4 + \epsilon$$

We will first compute the regression with natural polynomials:

```
. gen double w1 = weight
. gen double w2 = w1*w1
. gen double w3 = w2*w1
. gen double w4 = w3*w1
. correlate w1-w4
(obs=74)
```

|    | w1     | w2     | w3     | w4     |
|----|--------|--------|--------|--------|
| w1 | 1.0000 |        |        |        |
| w2 | 0.9915 | 1.0000 |        |        |
| w3 | 0.9665 | 0.9916 | 1.0000 |        |
| w4 | 0.9279 | 0.9679 | 0.9922 | 1.0000 |

```
. regress mpg w1-w4
```

| Source   | SS         | df | MS         |  |  |  |
|----------|------------|----|------------|--|--|--|
| Model    | 1652.73666 | 4  | 413.184164 |  |  |  |
| Residual | 790.722803 | 69 | 11.4597508 |  |  |  |
| Total    | 2443.45946 | 73 | 33.4720474 |  |  |  |

|  |                 |        |
|--|-----------------|--------|
|  | Number of obs = | 74     |
|  | F( 4, 69) =     | 36.06  |
|  | Prob > F =      | 0.0000 |
|  | R-squared =     | 0.6764 |
|  | Adj R-squared = | 0.6576 |
|  | Root MSE =      | 3.3852 |

| mpg   | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|-------|-----------|-----------|-------|-------|----------------------|----------|
| w1    | .0289302  | .1161939  | 0.25  | 0.804 | -.2028704            | .2607307 |
| w2    | -.0000229 | .0000566  | -0.40 | 0.687 | -.0001359            | .0000901 |
| w3    | 5.74e-09  | 1.19e-08  | 0.48  | 0.631 | -1.80e-08            | 2.95e-08 |
| w4    | -4.86e-13 | 9.14e-13  | -0.53 | 0.596 | -2.31e-12            | 1.34e-12 |
| _cons | 23.94421  | 86.60667  | 0.28  | 0.783 | -148.8314            | 196.7198 |

Some of the correlations among the powers of `weight` are very large, but this does not create any problems for `regress`. However, we may wish to look at the quadratic trend with the constant removed, the cubic trend with the quadratic and constant removed, etc. `orthpoly` will generate polynomial terms with this property:

```
. orthpoly weight, generate(pw*) deg(4) poly(P)
. regress mpg pw1-pw4
```

| Source   | SS         | df | MS         |  |  |  |
|----------|------------|----|------------|--|--|--|
| Model    | 1652.73666 | 4  | 413.184164 |  |  |  |
| Residual | 790.722803 | 69 | 11.4597508 |  |  |  |
| Total    | 2443.45946 | 73 | 33.4720474 |  |  |  |

|  |                 |        |
|--|-----------------|--------|
|  | Number of obs = | 74     |
|  | F( 4, 69) =     | 36.06  |
|  | Prob > F =      | 0.0000 |
|  | R-squared =     | 0.6764 |
|  | Adj R-squared = | 0.6576 |
|  | Root MSE =      | 3.3852 |

| mpg   | Coef.     | Std. Err. | t      | P> t  | [95% Conf. Interval] |           |
|-------|-----------|-----------|--------|-------|----------------------|-----------|
| pw1   | -4.638252 | .3935245  | -11.79 | 0.000 | -5.423312            | -3.853192 |
| pw2   | .8263545  | .3935245  | 2.10   | 0.039 | .0412947             | 1.611414  |
| pw3   | -.3068616 | .3935245  | -0.78  | 0.438 | -1.091921            | .4781982  |
| pw4   | -.209457  | .3935245  | -0.53  | 0.596 | -.9945168            | .5756028  |
| _cons | 21.2973   | .3935245  | 54.12  | 0.000 | 20.51224             | 22.08236  |

Compare the  $p$ -values of the terms in the natural-polynomial regression with those in the orthogonal-polynomial regression. With orthogonal polynomials, it is easy to see that the pure cubic and quartic trends are not significant and that the constant, linear, and quadratic terms each have  $p < 0.05$ .

The matrix  $P$  obtained with the `poly()` option can be used to transform coefficients for orthogonal polynomials to coefficients for natural polynomials:

```
. orthpoly weight, poly(P) deg(4)
. matrix b = e(b)*P
. matrix list b
b[1,5]
 deg1 deg2 deg3 deg4 _cons
y1 .02893016 -.00002291 5.745e-09 -4.862e-13 23.944212
```

◀

## Methods and formulas

`orthog` and `orthpoly` are implemented as ado-files.

`orthog`'s orthogonalization can be written in matrix notation as

$$X = QR$$

where  $X$  is the  $N \times (d + 1)$  matrix representation of *varlist* plus a column of ones and  $Q$  is the  $N \times (d + 1)$  matrix representation of *newvarlist* plus a column of ones ( $d$  = number of variables in *varlist*, and  $N$  = number of observations). The  $(d + 1) \times (d + 1)$  matrix  $R$  is a permuted upper-triangular matrix; i.e.,  $R$  would be upper triangular if the constant were first, but the constant is last, so the first row/column has been permuted with the last row/column.

$Q$  and  $R$  are obtained using a modified Gram–Schmidt procedure; see [Golub and Van Loan \(1996, 218–219\)](#) for details. The traditional Gram–Schmidt procedure is notoriously unsound, but the modified procedure is good. `orthog` performs two passes of this procedure.

`orthpoly` uses the Christoffel–Darboux recurrence formula ([Abramowitz and Stegun 1972](#)).

Both `orthog` and `orthpoly` normalize the orthogonal variables such that

$$Q'WQ = MI$$

where  $W = \text{diag}(w_1, w_2, \dots, w_N)$  with weights  $w_1, w_2, \dots, w_N$  (all 1 if weights are not specified), and  $M$  is the sum of the weights (the number of observations if weights are not specified).

## References

- Abramowitz, M., and I. A. Stegun, ed. 1972. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. 10th ed. Washington, DC: National Bureau of Standards.
- Golub, G. H., and C. F. Van Loan. 1996. *Matrix Computations*. 3rd ed. Baltimore: Johns Hopkins University Press.
- Sribney, W. M. 1995. [sg37: Orthogonal polynomials](#). *Stata Technical Bulletin* 25: 17–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 96–98. College Station, TX: Stata Press.

## Also see

[R] [regress](#) — Linear regression

# Title

pcorr — Partial and semipartial correlation coefficients

## Syntax

```
pcorr varname1 varlist [if] [in] [weight]
```

*varname*<sub>1</sub> and *varlist* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*by* is allowed; see [D] *by*.

*aweight*s and *fweight*s are allowed; see [U] 11.1.6 *weight*.

## Menu

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Partial correlations

## Description

`pcorr` displays the partial and semipartial correlation coefficients of *varname*<sub>1</sub> with each variable in *varlist* after removing the effects of all other variables in *varlist*. The squared correlations and corresponding significance are also reported.

## Remarks

Assume that *y* is determined by  $x_1, x_2, \dots, x_k$ . The partial correlation between *y* and  $x_1$  is an attempt to estimate the correlation that would be observed between *y* and  $x_1$  if the other *x*'s did not vary. The semipartial correlation, also called part correlation, between *y* and  $x_1$  is an attempt to estimate the correlation that would be observed between *y* and  $x_1$  after the effects of all other *x*'s are removed from  $x_1$  but not from *y*.

Both squared correlations estimate the proportion of the variance of *y* that is explained by each predictor. The squared semipartial correlation between *y* and  $x_1$  represents the proportion of variance in *y* that is explained by  $x_1$  only. This squared correlation can also be interpreted as the decrease in the model's  $R^2$  value that results from removing  $x_1$  from the full model. Thus one could use the squared semipartial correlations as criteria for model selection. The squared partial correlation between *y* and  $x_1$  represents the proportion of variance in *y* not associated with any other *x*'s that is explained by  $x_1$ . Thus the squared partial correlation gives an estimate of how much of the variance of *y* not explained by the other *x*'s is explained by  $x_1$ .

### ► Example 1

Using our automobile dataset (described in [U] 1.2.2 Example datasets), we can obtain the simple correlations between *price*, *mpg*, *weight*, and *foreign* from `correlate` (see [R] `correlate`):



```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. correlate price mpg weight foreign
(obs=74)
```

|         | price   | mpg     | weight  | foreign |
|---------|---------|---------|---------|---------|
| price   | 1.0000  |         |         |         |
| mpg     | -0.4686 | 1.0000  |         |         |
| weight  | 0.5386  | -0.8072 | 1.0000  |         |
| foreign | 0.0487  | 0.3934  | -0.5928 | 1.0000  |

Although `correlate` gave us the full correlation matrix, our interest is in just the first column. We find, for instance, that the higher the mpg, the lower the price. We obtain the partial and semipartial correlation coefficients by using `pcorr`:

```
. pcorr price mpg weight foreign
(obs=74)
```

| Partial and semipartial correlations of price with |                  |                      |                    |                        |                       |
|----------------------------------------------------|------------------|----------------------|--------------------|------------------------|-----------------------|
| Variable                                           | Partial<br>Corr. | Semipartial<br>Corr. | Partial<br>Corr.^2 | Semipartial<br>Corr.^2 | Significance<br>Value |
| mpg                                                | 0.0352           | 0.0249               | 0.0012             | 0.0006                 | 0.7693                |
| weight                                             | 0.5488           | 0.4644               | 0.3012             | 0.2157                 | 0.0000                |
| foreign                                            | 0.5402           | 0.4541               | 0.2918             | 0.2062                 | 0.0000                |

We now find that the partial and semipartial correlations of `price` with `mpg` are near 0. In the simple correlations, we found that `price` and `foreign` were virtually uncorrelated. In the partial and semipartial correlations, we find that `price` and `foreign` are positively correlated. The nonsignificance of `mpg` tells us that the amount in which  $R^2$  decreases by removing `mpg` from the model is not significant. We find that removing either `weight` or `foreign` results in a significant drop in the  $R^2$  of the model.

◀

## □ Technical note

Use caution when interpreting the above results. As we said at the outset, the partial and semipartial correlation coefficients are an *attempt* to estimate the correlation that would be observed if the effects of all other variables were taken out of both  $y$  and  $x$  or only  $x$ . `pcorr` makes it too easy to ignore the fact that we are fitting a model. In the example above, the model is

$$\text{price} = \beta_0 + \beta_1 \text{mpg} + \beta_2 \text{weight} + \beta_3 \text{foreign} + \epsilon$$

which is, in all honesty, a rather silly model. Even if we accept the implied economic assumptions of the model—that consumers value `mpg`, `weight`, and `foreign`—do we really believe that consumers place equal value on every extra 1,000 pounds of weight? That is, have we correctly parameterized the model? If we have not, then the estimated partial and semipartial correlation coefficients may not represent what they claim to represent. Partial and semipartial correlation coefficients are a reasonable way to summarize data if we are convinced that the underlying model is reasonable. We should not, however, pretend that there is no underlying model and that these correlation coefficients are unaffected by the assumptions and parameterization.

□

## Saved results

`pcorr` saves the following in `r()`:

|                         |                                            |
|-------------------------|--------------------------------------------|
| Scalars                 |                                            |
| <code>r(N)</code>       | number of observations                     |
| <code>r(df)</code>      | degrees of freedom                         |
| Matrices                |                                            |
| <code>r(p_corr)</code>  | partial correlation coefficient vector     |
| <code>r(sp_corr)</code> | semipartial correlation coefficient vector |

## Methods and formulas

`pcorr` is implemented as an ado-file.

Results are obtained by fitting a linear regression of *varname*<sub>1</sub> on *varlist*; see [R] [regress](#). The partial correlation coefficient between *varname*<sub>1</sub> and each variable in *varlist* is then calculated as

$$\frac{t}{\sqrt{t^2 + n - k}}$$

([Greene 2008](#), 30), where *t* is the *t* statistic, *n* is the number of observations, and *k* is the number of independent variables, including the constant but excluding any dropped variables.

The semipartial correlation coefficient between *varname*<sub>1</sub> and each variable in *varlist* is calculated as

$$\text{sign}(t) \sqrt{\frac{t^2(1 - R^2)}{n - k}}$$

(Cohen et al. [2003](#), 89), where *R*<sup>2</sup> is the model *R*<sup>2</sup> value, and *t*, *n*, and *k* are as described above.

The significance is given by  $2\Pr(t_{n-k} > |t|)$ , where *t*<sub>*n*−*k*</sub> follows a Student's *t* distribution with *n* − *k* degrees of freedom.

## Acknowledgment

The addition of semipartial correlation coefficients to `pcorr` is based on the `pcorr2` command by Richard Williams, University of Notre Dame.

## References

- Cohen, J., P. Cohen, S. G. West, and L. S. Aiken. 2003. *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. 3rd ed. Hillsdale, NJ: Erlbaum.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice-Hall.

## Also see

- [R] [correlate](#) — Correlations (covariances) of variables or coefficients
- [R] [spearman](#) — Spearman's and Kendall's correlations

Syntax

Compute permutation test

```
permute permvar exp_list [, options] : command
```

Report saved results

```
permute [varlist] [using filename] [, display_options]
```

| options                                | description                                                                                                                    |
|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| Main                                   |                                                                                                                                |
| <u>reps</u> (#)                        | perform # random permutations; default is <code>reps(100)</code>                                                               |
| <u>left</u>   <u>right</u>             | compute one-sided <i>p</i> -values; default is two-sided                                                                       |
| Options                                |                                                                                                                                |
| <u>strata</u> ( <i>varlist</i> )       | permute within strata                                                                                                          |
| <u>saving</u> ( <i>filename</i> , ...) | save results to <i>filename</i> ; save statistics in double precision;<br>save results to <i>filename</i> every # replications |
| Reporting                              |                                                                                                                                |
| <u>level</u> (#)                       | set confidence level; default is <code>level(95)</code>                                                                        |
| <u>noheader</u>                        | suppress table header                                                                                                          |
| <u>nolegend</u>                        | suppress table legend                                                                                                          |
| <u>verbose</u>                         | display full table legend                                                                                                      |
| <u>nodrop</u>                          | do not drop observations                                                                                                       |
| <u>nodots</u>                          | suppress replication dots                                                                                                      |
| <u>noisily</u>                         | display any output from <i>command</i>                                                                                         |
| <u>trace</u>                           | trace <i>command</i>                                                                                                           |
| <u>title</u> ( <i>text</i> )           | use <i>text</i> as title for permutation results                                                                               |
| Advanced                               |                                                                                                                                |
| <u>eps</u> (#)                         | numerical tolerance; seldom used                                                                                               |
| <u>nowarn</u>                          | do not warn when <code>e(sample)</code> is not set                                                                             |
| <u>force</u>                           | do not check for <i>weights</i> or <code>svy</code> commands; seldom used                                                      |
| <u>reject</u> ( <i>exp</i> )           | identify invalid results                                                                                                       |
| <u>seed</u> (#)                        | set random-number seed to #                                                                                                    |

*weights* are not allowed in *command*.

| <i>display_options</i>    | description                                             |
|---------------------------|---------------------------------------------------------|
| <code>left   right</code> | compute one-sided $p$ -values; default is two-sided     |
| <code>level(#)</code>     | set confidence level; default is <code>level(95)</code> |
| <code>noheader</code>     | suppress table header                                   |
| <code>nolegend</code>     | suppress table legend                                   |
| <code>verbose</code>      | display full table legend                               |
| <code>title(text)</code>  | use <i>text</i> as title for results                    |
| <code>eps(#)</code>       | numerical tolerance; seldom used                        |

*exp\_list* contains      (*name*: *elist*)  
                          *elist*  
                          *eexp*

*elist* contains        *newvar* = (*exp*)  
                          (*exp*)

*eexp* is                *specname*  
                          [*eqno*]*specname*

*specname* is         \_b  
                          \_b[]  
                          \_se  
                          \_se[]

*eqno* is             ##  
                          *name*

*exp* is a standard Stata expression; see [\[U\] 13 Functions and expressions](#).

Distinguish between `[]`, which are to be typed, and `[][]`, which indicate optional arguments.

## Menu

Statistics > Resampling > Permutation tests

## Description

`permute` estimates  $p$ -values for permutation tests on the basis of Monte Carlo simulations. Typing

```
. permute permvar exp_list, reps(#): command
```

randomly permutes the values in *permvar* # times, each time executing *command* and collecting the associated values from the expression in *exp\_list*.

These  $p$ -value estimates can be one-sided:  $\Pr(T^* \leq T)$  or  $\Pr(T^* \geq T)$ . The default is two-sided:  $\Pr(|T^*| \geq |T|)$ . Here  $T^*$  denotes the value of the statistic from a randomly permuted dataset, and  $T$  denotes the statistic as computed on the original data.

*permvar* identifies the variable whose observed values will be randomly permuted.

*command* defines the statistical command to be executed. Most Stata commands and user-written programs can be used with `permute`, as long as they follow standard Stata syntax; see [U] [11 Language syntax](#). The `by` prefix may not be part of *command*.

*exp\_list* specifies the statistics to be retrieved after the execution of *command*.

`permute` may be used for replaying results, but this feature is appropriate only when a dataset generated by `permute` is currently in memory or is identified by the `using` option. The variables specified in *varlist* in this context must be present in the respective dataset.

## Options

### Main

`reps(#)` specifies the number of random permutations to perform. The default is 100.

`left` or `right` requests that one-sided *p*-values be computed. If `left` is specified, an estimate of  $\Pr(T^* \leq T)$  is produced, where  $T^*$  is the test statistic and  $T$  is its observed value. If `right` is specified, an estimate of  $\Pr(T^* \geq T)$  is produced. By default, two-sided *p*-values are computed; that is,  $\Pr(|T^*| \geq |T|)$  is estimated.

### Options

`strata(varlist)` specifies that the permutations be performed within each stratum defined by the values of *varlist*.

`saving(filename[, suboptions])` creates a Stata data file (`.dta` file) consisting of, for each statistic in *exp\_list*, a variable containing the permutation replicates.

`double` specifies that the results for each replication be stored as `doubles`, meaning 8-byte reals.

By default, they are stored as `floats`, meaning 4-byte reals.

`every(#)` specifies that results are to be written to disk every *#*th replication. `every()` should be specified only in conjunction with `saving()` when *command* takes a long time for each replication. This will allow recovery of partial results should some other software crash your computer. See [P] [postfile](#).

`replace` indicates that *filename* may exist, and, if it does, it should be overwritten. This option does not appear in the dialog box.

### Reporting

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see [R] [level](#).

`noheader` suppresses display of the table header. This option implies `nolegend`.

`nolegend` suppresses display of the table legend. The table legend identifies the rows of the table with the expressions they represent.

`verbose` requests that the full table legend be displayed. By default, coefficients and standard errors are not displayed.

`nodrop` prevents `permute` from dropping observations outside the `if` and `in` qualifiers. `nodrop` will also cause `permute` to ignore the contents of `e(sample)` if it exists as a result of running *command*. By default, `permute` temporarily drops out-of-sample observations.

`nodots` suppresses display of the replication dots. By default, one dot character is displayed for each successful replication. A red ‘x’ is displayed if *command* returns an error or if one of the values in *exp\_list* is missing.

`noisily` requests that any output from *command* be displayed. This option implies `nodots`.

`trace` causes a trace of the execution of *command* to be displayed. This option implies `noisily`.

`title(text)` specifies a title to be displayed above the table of permutation results; the default title is `Monte Carlo permutation results`.

#### Advanced

`eps(#)` specifies the numerical tolerance for testing  $|T^*| \geq |T|$ ,  $T^* \leq T$ , or  $T^* \geq T$ . These are considered true if, respectively,  $|T^*| \geq |T| - \#$ ,  $T^* \leq T + \#$ , or  $T^* \geq T - \#$ . The default is `1e-7`. You will not have to specify `eps()` under normal circumstances.

`nowarn` suppresses the printing of a warning message when *command* does not set `e(sample)`.

`force` suppresses the restriction that *command* may not specify weights or be a `svy` command. `permute` is not suited for weighted estimation, thus `permute` should not be used with weights or `svy`. `permute` reports an error when it encounters weights or `svy` in *command* if the `force` option is not specified. This is a seldom used option, so use it only if you know what you are doing!

`reject(exp)` identifies an expression that indicates when results should be rejected. When *exp* is true, the resulting values are reset to missing values.

`seed(#)` sets the random-number seed. Specifying this option is equivalent to typing the following command prior to calling `permute`:

```
. set seed #
```

## Remarks

Permutation tests determine the significance of the observed value of a test statistic in light of rearranging the order (permuting) of the observed values of a variable.

### ► Example 1

Suppose that we conducted an experiment to determine the effect of a treatment on the development of cells. Further suppose that we are restricted to six experimental units because of the extreme cost of the experiment. Thus three units are to be given a placebo, and three units are given the treatment. The measurement is the number of newly developed healthy cells. The following listing gives the hypothetical data, along with some summary statistics.

```
. input y treatment
 y treatment
1. 7 0
2. 9 0
3. 11 0
4. 10 1
5. 12 1
6. 14 1
7. end
. sort treatment
```

```
. summarize y
```

| Variable | Obs | Mean | Std. Dev. | Min | Max |
|----------|-----|------|-----------|-----|-----|
| y        | 6   | 10.5 | 2.428992  | 7   | 14  |

```
. by treatment: summarize y
```

---

```
-> treatment = 0
```

| Variable | Obs | Mean | Std. Dev. | Min | Max |
|----------|-----|------|-----------|-----|-----|
| y        | 3   | 9    | 2         | 7   | 11  |

---

```
-> treatment = 1
```

| Variable | Obs | Mean | Std. Dev. | Min | Max |
|----------|-----|------|-----------|-----|-----|
| y        | 3   | 12   | 2         | 10  | 14  |

Clearly, there are more cells in the treatment group than in the placebo group, but a statistical test is needed to conclude that the treatment does affect the development of cells. If the sum of the treatment measures is our test statistic, we can use `permute` to determine the probability of observing 36 or more cells, given the observed data and assuming that there is no effect due to the treatment.

```
. set seed 1234
. permute y sum=r(sum), saving(permdish) right nodrop nowarn: sum y if treatment
(running summarize on estimation sample)

Permutation replications (100)
-----|----- 1 -----|----- 2 -----|----- 3 -----|----- 4 -----|----- 5
.....|..... 50
.....|..... 100

Monte Carlo permutation results Number of obs = 6

 command: summarize y if treatment
 sum: r(sum)
 permute var: y
```

| T   | T(obs) | c  | n   | p=c/n  | SE(p)  | [95% Conf. Interval] |          |
|-----|--------|----|-----|--------|--------|----------------------|----------|
| sum | 36     | 10 | 100 | 0.1000 | 0.0300 | .0490047             | .1762226 |

Note: confidence interval is with respect to  $p=c/n$ .

Note:  $c = \#\{T \geq T(\text{obs})\}$

We see that 10 of the 100 randomly permuted datasets yielded sums from the treatment group larger than or equal to the observed sum of 36. Thus the evidence is not strong enough, at the 5% level, to reject the null hypothesis that there is no effect of the treatment.

Because of the small size of this experiment, we could have calculated the exact permutation  $p$ -value from all possible permutations. There are six units, but we want the sum of the treatment units. Thus there are  $\binom{6}{3} = 20$  permutation sums from the possible unique permutations.

(Continued on next page)

|                  |                  |                  |                   |
|------------------|------------------|------------------|-------------------|
| 7 + 9 + 10 = 26  | 7 + 10 + 12 = 29 | 9 + 10 + 11 = 30 | 9 + 12 + 14 = 35  |
| 7 + 9 + 11 = 27  | 7 + 10 + 14 = 31 | 9 + 10 + 12 = 31 | 10 + 11 + 12 = 33 |
| 7 + 9 + 12 = 28  | 7 + 11 + 12 = 30 | 9 + 10 + 14 = 33 | 10 + 11 + 14 = 35 |
| 7 + 9 + 14 = 30  | 7 + 11 + 14 = 32 | 9 + 11 + 12 = 32 | 10 + 12 + 14 = 36 |
| 7 + 10 + 11 = 28 | 7 + 12 + 14 = 33 | 9 + 11 + 14 = 34 | 11 + 12 + 14 = 37 |

Two of the 20 permutation sums are greater than or equal to 36. Thus the exact *p*-value for this permutation test is 0.1. Tied values will decrease the number of unique permutations.

When the `saving()` option is supplied, `permute` saves the values of the permutation statistic to the indicated file, in our case, `permdish.dta`. This file can be used to replay the result of `permute`. The `level()` option controls the confidence level of the confidence interval for the permutation *p*-value. This confidence interval is calculated using `cii` with the reported `n` (number of nonmissing replications) and `c` (the counter for events of significance).

```
. permute using permdish, level(80)
Monte Carlo permutation results Number of obs = 6
 command: summarize y if treatment
 sum: r(sum)
 permute var: y
```

| T   | T(obs) | c  | n   | p=c/n  | SE(p)  | [80% Conf. Interval] |
|-----|--------|----|-----|--------|--------|----------------------|
| sum | 36     | 10 | 100 | 0.1000 | 0.0300 | .0631113 .1498826    |

Note: confidence interval is with respect to  $p=c/n$ .  
Note:  $c = \#\{|T| \geq |T(\text{obs})|\}$

➤ Example 2

Consider some fictional data from a randomized complete-block design in which we wish to determine the significance of five treatments.

```
. use http://www.stata-press.com/data/r11/permute1, clear
. list y treatment in 1/10, abbrev(10)
```

|     | y        | treatment |
|-----|----------|-----------|
| 1.  | 4.407557 | 1         |
| 2.  | 5.693386 | 1         |
| 3.  | 7.099699 | 1         |
| 4.  | 3.12132  | 1         |
| 5.  | 5.242648 | 1         |
| 6.  | 4.280349 | 2         |
| 7.  | 4.508785 | 2         |
| 8.  | 4.079967 | 2         |
| 9.  | 5.904368 | 2         |
| 10. | 3.010556 | 2         |



These data may be analyzed using `anova`.

```
. anova y treatment subject
```

|           | Number of obs = | 50      | R-squared =     | 0.3544 |          |
|-----------|-----------------|---------|-----------------|--------|----------|
|           | Root MSE =      | .914159 | Adj R-squared = | 0.1213 |          |
| Source    | Partial SS      | df      | MS              | F      | Prob > F |
| Model     | 16.5182188      | 13      | 1.27063221      | 1.52   | 0.1574   |
| treatment | 13.0226706      | 9       | 1.44696341      | 1.73   | 0.1174   |
| subject   | 3.49554813      | 4       | .873887032      | 1.05   | 0.3973   |
| Residual  | 30.0847503      | 36      | .835687509      |        |          |
| Total     | 46.6029691      | 49      | .951081002      |        |          |

Suppose that we want to compute the significance of the  $F$  statistic for `treatment` by using `permute`. All we need to do is write a short program that will save the result of this statistic for `permute` to use. For example,

```
program panova, rclass
 version 11
 args response fac_intrst fac_other
 anova 'response' 'fac_intrst' 'fac_other'
 return scalar Fmodel = e(F)
 test 'fac_intrst'
 return scalar F = r(F)
end
```

Now in `panova`, `test` saves the  $F$  statistic for the factor of interest in `r(F)`. This is different from `e(F)`, which is the overall model  $F$  statistic for the model fit by `anova` that `panova` saves in `r(Fmodel)`. In the following example, we use the `strata()` option so that the treatments are randomly rearranged within each subject. It should not be too surprising that the estimated  $p$ -values are equal for this example, because the two  $F$  statistics are equivalent when controlling for differences between subjects. However, we would not expect to always get the same  $p$ -values every time we reran `permute`.

```
. set seed 1234
. permute treatmentF=r(F) modelF=e(F), reps(1000) strata(subject)
> saving(permanova) nodots: panova y treatment subject
```

Monte Carlo permutation results

|                    |                            |                 |    |
|--------------------|----------------------------|-----------------|----|
| Number of strata = | 5                          | Number of obs = | 50 |
| command:           | panova y treatment subject |                 |    |
| treatmentF:        | r(F)                       |                 |    |
| modelF:            | e(F)                       |                 |    |
| permute var:       | treatment                  |                 |    |

| T          | T(obs)   | c   | n    | p=c/n  | SE(p)  | [95% Conf. Interval] |
|------------|----------|-----|------|--------|--------|----------------------|
| treatmentF | 1.731465 | 118 | 1000 | 0.1180 | 0.0102 | .0986525 .1396277    |
| modelF     | 1.520463 | 118 | 1000 | 0.1180 | 0.0102 | .0986525 .1396277    |

Note: confidence intervals are with respect to  $p=c/n$ .

Note:  $c = \#\{|T| \geq |T(\text{obs})|\}$

➤ Example 3

As a final example, let’s consider estimating the  $p$ -value of the  $Z$  statistic returned by `ranksum`. Suppose that we collected data from some experiment: `y` is some measure we took on 17 individuals, and `group` identifies the group that an individual belongs to.

```
. use http://www.stata-press.com/data/r11/permute2, clear
. list
```

|     | group | y  |
|-----|-------|----|
| 1.  | 1     | 6  |
| 2.  | 1     | 11 |
| 3.  | 1     | 20 |
| 4.  | 1     | 2  |
| 5.  | 1     | 9  |
| 6.  | 1     | 5  |
| 7.  | 0     | 2  |
| 8.  | 0     | 1  |
| 9.  | 0     | 6  |
| 10. | 0     | 0  |
| 11. | 0     | 2  |
| 12. | 0     | 3  |
| 13. | 0     | 3  |
| 14. | 0     | 12 |
| 15. | 0     | 4  |
| 16. | 0     | 1  |
| 17. | 0     | 5  |

Next we analyze the data using `ranksum` and notice that the observed value of the test statistic (saved as `r(z)`) is  $-2.02$  with an approximate  $p$ -value of  $0.0434$ .

```
. ranksum y, by(group)
Two-sample Wilcoxon rank-sum (Mann-Whitney) test
```

| group    | obs | rank sum | expected |
|----------|-----|----------|----------|
| 0        | 11  | 79       | 99       |
| 1        | 6   | 74       | 54       |
| combined | 17  | 153      | 153      |

```
unadjusted variance 99.00
adjustment for ties -0.97

adjusted variance 98.03
Ho: y(group==0) = y(group==1)
 z = -2.020
 Prob > |z| = 0.0434
```

The observed value of the rank-sum statistic is 79, with an expected value (under the null hypothesis of no group effect) of 99. There are 17 observations, so the permutation distribution contains  $\binom{17}{6} = 12,376$  possible values of the rank-sum statistic if we ignore ties. With ties, we have fewer possible values but still too many to want to count them. Thus we use `permute` with 10,000 replications and see that the Monte Carlo permutation test agrees with the result of the test based on the normal approximation.

```
. set seed 18385766
. permute y z=r(z), reps(10000) nowarn nodots: ranksum y, by(group)
Monte Carlo permutation results Number of obs = 17
 command: ranksum y, by(group)
 z: r(z)
 permute var: y
```

| T | T(obs)    | c   | n     | p=c/n  | SE(p)  | [95% Conf. Interval] |
|---|-----------|-----|-------|--------|--------|----------------------|
| z | -2.020002 | 468 | 10000 | 0.0468 | 0.0021 | .0427429 .0511236    |

Note: confidence interval is with respect to  $p=c/n$ .  
Note:  $c = \#\{|T| \geq |T(\text{obs})|\}$



For an application of a permutation test to a problem in epidemiology, see [Hayes and Moulton \(2009, 190–193\)](#).

□ Technical note

`permute` reports confidence intervals for  $p$  to emphasize that it is based on the binomial estimator for proportions. When the variability implied by the confidence interval makes conclusions difficult, you may increase the number of replications to determine more precisely the significance of the test statistic of interest. In other words, the value of  $p$  from `permute` will converge to the true permutation  $p$ -value as the number of replications gets arbitrarily large.



## Saved results

`permute` saves the following in `r()`:

|                         |                                          |                        |                                                                                |
|-------------------------|------------------------------------------|------------------------|--------------------------------------------------------------------------------|
| Scalars                 |                                          |                        |                                                                                |
| <code>r(N)</code>       | sample size                              | <code>r(k_exp)</code>  | number of standard expressions                                                 |
| <code>r(N_reps)</code>  | number of requested replications         | <code>r(k_eexp)</code> | number of <code>_b/_se</code> expressions                                      |
| <code>r(level)</code>   | confidence level                         |                        |                                                                                |
| Macros                  |                                          |                        |                                                                                |
| <code>r(cmd)</code>     | <code>permute</code>                     | <code>r(left)</code>   | left or empty                                                                  |
| <code>r(command)</code> | <i>command</i> following colon           | <code>r(right)</code>  | right or empty                                                                 |
| <code>r(permvar)</code> | permutation variable                     | <code>r(seed)</code>   | initial random-number seed                                                     |
| <code>r(title)</code>   | title in output                          | <code>r(event)</code>  | $T \leq T(\text{obs}), T \geq T(\text{obs}),$<br>or $ T  \leq  T(\text{obs}) $ |
| <code>r(exp#)</code>    | <i>#</i> th expression                   |                        |                                                                                |
| Matrices                |                                          |                        |                                                                                |
| <code>r(b)</code>       | observed statistics                      | <code>r(p)</code>      | observed proportions                                                           |
| <code>r(c)</code>       | count when <code>r(event)</code> is true | <code>r(se)</code>     | standard errors of observed proportions                                        |
| <code>r(reps)</code>    | number of nonmissing results             | <code>r(ci)</code>     | confidence intervals of observed proportions                                   |

## Methods and formulas

`permute` is implemented as an ado-file.

## References

- Good, P. I. 2006. *Resampling Methods: A Practical Guide to Data Analysis*. 3rd ed. Boston: Birkhäuser.
- Hayes, R. J., and L. H. Moulton. 2009. *Cluster Randomised Trials*. Boca Raton, FL: Chapman & Hall/CRC.
- Kaiser, J. 2007. An exact and a Monte Carlo proposal to the Fisher–Pitman permutation tests for paired replicates and for independent samples. *Stata Journal* 7: 402–412.
- Kaiser, J., and M. G. Lacy. 2009. A general-purpose method for two-group randomization tests. *Stata Journal* 9: 70–85.

## Also see

- [R] **bootstrap** — Bootstrap sampling and estimation
- [R] **jackknife** — Jackknife estimation
- [R] **simulate** — Monte Carlo simulations

Description

The term pk refers to pharmacokinetic data and the Stata commands, all of which begin with the letters pk, designed to do some of the analyses commonly performed in the pharmaceutical industry. The system is intended for the analysis of pharmacokinetic data, although some of the commands are for general use.

The pk commands are

|            |                                |                                              |
|------------|--------------------------------|----------------------------------------------|
| pkexamine  | [R] <a href="#">pkexamine</a>  | Calculate pharmacokinetic measures           |
| pksumm     | [R] <a href="#">pksumm</a>     | Summarize pharmacokinetic data               |
| pkshape    | [R] <a href="#">pkshape</a>    | Reshape (pharmacokinetic) Latin-square data  |
| pkcross    | [R] <a href="#">pkcross</a>    | Analyze crossover experiments                |
| pkequiv    | [R] <a href="#">pkequiv</a>    | Perform bioequivalence tests                 |
| pkcollapse | [R] <a href="#">pkcollapse</a> | Generate pharmacokinetic measurement dataset |

Remarks

Several types of clinical trials are commonly performed in the pharmaceutical industry. Examples include combination trials, multicenter trials, equivalence trials, and active control trials. For each type of trial, there is an optimal study design for estimating the effects of interest. Currently, the pk system can be used to analyze equivalence trials, which are usually conducted using a crossover design; however, it is possible to use a parallel design and still draw conclusions about equivalence.

Equivalence trials assess bioequivalence between two drugs. Although proving that two drugs behave the same is impossible, the United States Food and Drug Administration believes that if the absorption properties of two drugs are similar, the two drugs will produce similar effects and have similar safety profiles. Generally, the goal of an equivalence trial is to assess the equivalence of a generic drug to an existing drug. This goal is commonly accomplished by comparing a confidence interval about the difference between a pharmacokinetic measurement of two drugs with a confidence limit constructed from U.S. federal regulations. If the confidence interval is entirely within the confidence limit, the drugs are declared bioequivalent. Another approach to assessing bioequivalence is to use the method of interval hypotheses testing. `pkequiv` is used to conduct these tests of bioequivalence.

Several pharmacokinetic measures can be used to ascertain how available a drug is for cellular absorption. The most common measure is the area under the time-versus-concentration curve (AUC). Another common measure of drug availability is the maximum concentration ( $C_{\text{max}}$ ) achieved by the drug during the follow-up period. Stata reports these and other less common measures of drug availability, including the time at which the maximum drug concentration was observed and the duration of the period during which the subject was being measured. Stata also reports the elimination rate, that is, the rate at which the drug is metabolized, and the drug's half-life, that is, the time it takes for the drug concentration to fall to one-half of its maximum concentration.

`pkexamine` computes and reports all the pharmacokinetic measures that Stata produces, including four calculations of the area under the time-versus-concentration curve. The standard area under the curve from 0 to the maximum observed time ( $\text{AUC}_{0,t_{\text{max}}}$ ) is computed using cubic splines or the

trapezoidal rule. Additionally, `pkexamine` also computes the area under the curve from 0 to infinity by extending the standard time-versus-concentration curve from the maximum observed time by using three different methods. The first method simply extends the standard curve by using a least-squares linear fit through the last few data points. The second method extends the standard curve by fitting a decreasing exponential curve through the last few data points. Finally, the third method extends the curve by fitting a least-squares linear regression line on the log concentration. The mathematical details of these extensions are described in *Methods and formulas* of [R] `pkexamine`.

Data from an equivalence trial may also be analyzed using methods appropriate to the particular study design. When you have a crossover design, `pkcross` can be used to fit an appropriate ANOVA model. As an aside, a crossover design is simply a restricted Latin square; therefore, `pkcross` can also be used to analyze any Latin-square design.

There are some practical concerns when dealing with data from equivalence trials. Primarily, the data must be organized in a manner that Stata can use. The `pk` commands include `pkcollapse` and `pkshape`, which are designed to help transform data from a common format to one that is suitable for analysis with Stata.

In the following example, we illustrate several different data formats that are often encountered in pharmaceutical research and describe how these formats can be transformed to formats that can be analyzed with Stata.

➤ Example 1

Assume that we have one subject and are interested in determining the drug profile for that subject. A reasonable experiment would be to give the subject the drug and then measure the concentration of the drug in the subject’s blood over a given period. For example, here is a part of a dataset from [Chow and Liu \(2009, 13\)](#):

```
. use http://www.stata-press.com/data/r11/auc
. list, abbrev(14)
```

|     | id | time | concentration |
|-----|----|------|---------------|
| 1.  | 1  | 0    | 0             |
| 2.  | 1  | .5   | 0             |
| 3.  | 1  | 1    | 2.8           |
| 4.  | 1  | 1.5  | 4.4           |
| 5.  | 1  | 2    | 4.4           |
| 6.  | 1  | 3    | 4.7           |
| 7.  | 1  | 4    | 4.1           |
| 8.  | 1  | 6    | 4             |
| 9.  | 1  | 8    | 3.6           |
| 10. | 1  | 12   | 3             |
| 11. | 1  | 16   | 2.5           |
| 12. | 1  | 24   | 2             |
| 13. | 1  | 32   | 1.6           |

Examining these data, we notice that the concentration quickly increases, plateaus for a short period, and then slowly decreases over time. `pkexamine` is used to calculate the pharmacokinetic measures of interest. `pkexamine` is explained in detail in [R] `pkexamine`. The output is

```
. pkexamine time conc
```

```

Maximum concentration = 4.7
Time of maximum concentration = 3
Time of last observation (Tmax) = 32
Elimination rate = 0.0279
Half life = 24.8503

```

Area under the curve

| AUC [0, Tmax] | AUC [0, inf.)<br>Linear of log conc. | AUC [0, inf.)<br>Linear fit | AUC [0, inf.)<br>Exponential fit |
|---------------|--------------------------------------|-----------------------------|----------------------------------|
| 85.24         | 142.603                              | 107.759                     | 142.603                          |

Fit based on last 3 points.

Clinical trials, however, require that data be collected on more than one subject. There are several ways to enter raw measured data collected on several subjects. It would be reasonable to enter for each subject the drug concentration value at specific points in time. Such data could be

| id | conc1 | conc2 | conc3 | conc4 | conc5 | conc6 | conc7 |
|----|-------|-------|-------|-------|-------|-------|-------|
| 1  | 0     | 1     | 4     | 7     | 5     | 3     | 1     |
| 2  | 0     | 2     | 6     | 5     | 4     | 3     | 2     |
| 3  | 0     | 1     | 2     | 3     | 5     | 4     | 1     |

where conc1 is the concentration at the first measured time, conc2 is the concentration at the second measured time, etc. This format requires that each drug concentration measurement be made at the same time on each subject. Another more flexible way to enter the data is to have an observation with three variables for each time measurement on a subject. Each observation would have a subject ID, the time at which the measurement was made, and the corresponding drug concentration at that time. The data would be

(Continued on next page)

```
. use http://www.stata-press.com/data/r11/pkdata
. list id concA time, sepby(id)
```

|      | id               | concA    | time |
|------|------------------|----------|------|
| 1.   | 1                | 0        | 0    |
| 2.   | 1                | 3.073403 | .5   |
| 3.   | 1                | 5.188444 | 1    |
| 4.   | 1                | 5.898577 | 1.5  |
| 5.   | 1                | 5.096378 | 2    |
| 6.   | 1                | 6.094085 | 3    |
| 7.   | 1                | 5.158772 | 4    |
| 8.   | 1                | 5.7065   | 6    |
| 9.   | 1                | 5.272467 | 8    |
| 10.  | 1                | 4.4576   | 12   |
| 11.  | 1                | 5.146423 | 16   |
| 12.  | 1                | 4.947427 | 24   |
| 13.  | 1                | 1.920421 | 32   |
| 14.  | 2                | 0        | 0    |
| 15.  | 2                | 2.48462  | .5   |
| 16.  | 2                | 4.883569 | 1    |
| 17.  | 2                | 7.253442 | 1.5  |
| 18.  | 2                | 5.849345 | 2    |
| 19.  | 2                | 6.761085 | 3    |
| 20.  | 2                | 4.33839  | 4    |
| 21.  | 2                | 5.04199  | 6    |
| 22.  | 2                | 4.25128  | 8    |
| 23.  | 2                | 6.205004 | 12   |
| 24.  | 2                | 5.566165 | 16   |
| 25.  | 2                | 3.689007 | 24   |
| 26.  | 2                | 3.644063 | 32   |
| 27.  | 3                | 0        | 0    |
|      | (output omitted) |          |      |
| 207. | 20               | 4.673281 | 24   |
| 208. | 20               | 3.487347 | 32   |

Stata expects the data to be organized in the second form. If your data are organized as described in the first dataset, you will need to use `reshape` to change the data to the second form; see [\[D\] reshape](#). Because the data in the second (or long) format contain information for one drug on several subjects, `pksum` can be used to produce summary statistics of the pharmacokinetic measurements. The output is

```
. pksumm id time concA
.....
```

| Summary statistics for the pharmacokinetic measures |        |        |           |          |          |         |
|-----------------------------------------------------|--------|--------|-----------|----------|----------|---------|
| Number of observations = 16                         |        |        |           |          |          |         |
| Measure                                             | Mean   | Median | Variance  | Skewness | Kurtosis | p-value |
| auc                                                 | 151.63 | 152.18 | 127.58    | -0.34    | 2.07     | 0.55    |
| aucline                                             | 397.09 | 219.83 | 178276.59 | 2.69     | 9.61     | 0.00    |
| aucexp                                              | 668.60 | 302.96 | 720356.98 | 2.67     | 9.54     | 0.00    |
| auclog                                              | 665.95 | 298.03 | 752573.34 | 2.71     | 9.70     | 0.00    |
| half                                                | 90.68  | 29.12  | 17750.70  | 2.36     | 7.92     | 0.00    |
| ke                                                  | 0.02   | 0.02   | 0.00      | 0.88     | 3.87     | 0.08    |
| cmax                                                | 7.37   | 7.42   | 0.40      | -0.64    | 2.75     | 0.36    |
| tomc                                                | 3.38   | 3.00   | 7.25      | 2.27     | 7.70     | 0.00    |
| tmax                                                | 32.00  | 32.00  | 0.00      | .        | .        | .       |



Until now, we have been concerned with the profile of only one drug. We have characterized the profile of that drug by individual subjects by using `pkexamine` and by a group of subjects by using `pksumm`. The goal of an equivalence trial, however, is to compare two drugs, which we will do in the rest of this example.

For equivalence trials, the study design most often used is the crossover design. For a complete discussion of crossover designs, see [Ratkowsky, Evans, and Alldredge \(1993\)](#).

In brief, crossover designs require that each subject be given both treatments at two different times. The order in which the treatments are applied changes between groups. For example, if we had 20 subjects numbered 1–20, the first 10 would receive treatment A during the first period of the study, and then they would be given treatment B. The second 10 subjects would be given treatment B during the first period of the study, and then they would be given treatment A. Each subject in the study will have four variables that describe the observation: a subject identifier, a sequence identifier that indicates the order of treatment, and two outcome variables, one for each treatment. The outcome variables for each subject are the pharmacokinetic measures. The data must be transformed from a series of measurements on individual subjects to data containing the pharmacokinetic measures for each subject. In Stata parlance, this is referred to as a collapse, which can be done with `pkcollapse`; see [\[R\] pkcollapse](#).

Here is a part of our data:

```
. list, sepby(id)
```

|                  | id | seq | time | concA    | concB    |
|------------------|----|-----|------|----------|----------|
| 1.               | 1  | 1   | 0    | 0        | 0        |
| 2.               | 1  | 1   | .5   | 3.073403 | 3.712592 |
| 3.               | 1  | 1   | 1    | 5.188444 | 6.230602 |
| 4.               | 1  | 1   | 1.5  | 5.898577 | 7.885944 |
| 5.               | 1  | 1   | 2    | 5.096378 | 9.241735 |
| 6.               | 1  | 1   | 3    | 6.094085 | 13.10507 |
| 7.               | 1  | 1   | 4    | 5.158772 | .169429  |
| 8.               | 1  | 1   | 6    | 5.7065   | 8.759894 |
| 9.               | 1  | 1   | 8    | 5.272467 | 7.985409 |
| 10.              | 1  | 1   | 12   | 4.4576   | 7.740126 |
| 11.              | 1  | 1   | 16   | 5.146423 | 7.607208 |
| 12.              | 1  | 1   | 24   | 4.947427 | 7.588428 |
| 13.              | 1  | 1   | 32   | 1.920421 | 2.791115 |
| 14.              | 2  | 1   | 0    | 0        | 0        |
| 15.              | 2  | 1   | .5   | 2.48462  | .9209593 |
| 16.              | 2  | 1   | 1    | 4.883569 | 5.925818 |
| 17.              | 2  | 1   | 1.5  | 7.253442 | 8.710549 |
| 18.              | 2  | 1   | 2    | 5.849345 | 10.90552 |
| 19.              | 2  | 1   | 3    | 6.761085 | 8.429898 |
| 20.              | 2  | 1   | 4    | 4.33839  | 5.573152 |
| 21.              | 2  | 1   | 6    | 5.04199  | 6.32341  |
| 22.              | 2  | 1   | 8    | 4.25128  | .5251224 |
| 23.              | 2  | 1   | 12   | 6.205004 | 7.415988 |
| 24.              | 2  | 1   | 16   | 5.566165 | 6.323938 |
| 25.              | 2  | 1   | 24   | 3.689007 | 1.133553 |
| 26.              | 2  | 1   | 32   | 3.644063 | 5.759489 |
| 27.              | 3  | 1   | 0    | 0        | 0        |
| (output omitted) |    |     |      |          |          |
| 207.             | 20 | 2   | 24   | 4.673281 | 6.059818 |
| 208.             | 20 | 2   | 32   | 3.487347 | 5.213639 |

This format is similar to the second format described above, except that now we have measurements for two drugs at each time for each subject. We transform these data with `pkcollapse`:

```
. pkcollapse time concA concB, id(id) keep(seq) stat(auc)
.....
. list, sep(8) abbrev(10)
```

|     | id | seq | auc_concA | auc_concB |
|-----|----|-----|-----------|-----------|
| 1.  | 1  | 1   | 150.9643  | 218.5551  |
| 2.  | 2  | 1   | 146.7606  | 133.3201  |
| 3.  | 3  | 1   | 160.6548  | 126.0635  |
| 4.  | 4  | 1   | 157.8622  | 96.17461  |
| 5.  | 5  | 1   | 133.6957  | 188.9038  |
| 6.  | 7  | 1   | 160.639   | 223.6922  |
| 7.  | 8  | 1   | 131.2604  | 104.0139  |
| 8.  | 9  | 1   | 168.5186  | 237.8962  |
| 9.  | 10 | 2   | 137.0627  | 139.7382  |
| 10. | 12 | 2   | 153.4038  | 202.3942  |
| 11. | 13 | 2   | 163.4593  | 136.7848  |
| 12. | 14 | 2   | 146.0462  | 104.5191  |
| 13. | 15 | 2   | 158.1457  | 165.8654  |
| 14. | 18 | 2   | 147.1977  | 139.235   |
| 15. | 19 | 2   | 164.9988  | 166.2391  |
| 16. | 20 | 2   | 145.3823  | 158.5146  |

For this example, we chose to use the AUC for two drugs as our pharmacokinetic measure. We could have used any of the measures computed by `pkexamine`. In addition to the AUCs, the dataset also contains a sequence variable for each subject indicating when each treatment was administered.

The data produced by `pkcollapse` are in what Stata calls wide format; that is, there is one observation per subject containing two or more outcomes. To use `pkcross` and `pkequiv`, we need to transform these data to long format. This goal can be accomplished using `pkshape`; see [\[R\] pkshape](#).

Consider the first subject in the dataset. This subject is in sequence one, which means that treatment A was applied during the first period of the study and treatment B was applied in the second period of the study. We need to split the first observation into two observations so that the outcome measure is only in one variable. Also we need two new variables, one indicating the treatment the subject received and another recording the period of the study when the subject received that treatment. We might expect the expansion of the first subject to be

| id | sequence | auc      | treat | period |
|----|----------|----------|-------|--------|
| 1  | 1        | 150.9643 | A     | 1      |
| 1  | 1        | 218.5551 | B     | 2      |

We see that subject number 1 was in sequence 1, had an AUC of 150.9643 when treatment A was applied in the first period of the study, and had an AUC of 218.5551 when treatment B was applied.

Similarly, the expansion of subject 10 (the first subject in sequence 2) would be

| id | sequence | auc      | treat | period |
|----|----------|----------|-------|--------|
| 10 | 2        | 137.0627 | B     | 1      |
| 10 | 2        | 139.7382 | A     | 2      |

Here treatment B was applied to the subject during the first period of the study, and treatment A was applied to the subject during the second period of the study.

An additional complication is common in crossover study designs. The treatment applied in the first period of the study might still have some effect on the outcome in the second period. In this example,

each subject was given one treatment followed by another treatment. To get accurate estimates of treatment effects, it is necessary to account for the effect that the first treatment has in the second period of the study. This is called the carryover effect. We must, therefore, have a variable that indicates which treatment was applied in the first treatment period. `pkshape` creates a variable that indicates the carryover effect. For treatments applied during the first treatment period, there will never be a carryover effect. Thus the expanded data created by `pkshape` for subject 1 will be

| id | sequence | outcome  | treat | period | carry |
|----|----------|----------|-------|--------|-------|
| 1  | 1        | 150.9643 | A     | 1      | 0     |
| 1  | 1        | 218.5551 | B     | 2      | A     |

and the data for subject 10 will be

| id | sequence | outcome  | treat | period | carry |
|----|----------|----------|-------|--------|-------|
| 10 | 2        | 137.0627 | B     | 1      | 0     |
| 10 | 2        | 139.7382 | A     | 2      | B     |

We `pkshape` the data:

```
. pkshape id seq auc*, order(ab ba)
. sort id sequence period
. list, sep(16)
```

|     | id | sequence | outcome  | treat | carry | period |
|-----|----|----------|----------|-------|-------|--------|
| 1.  | 1  | 1        | 150.9643 | 1     | 0     | 1      |
| 2.  | 1  | 1        | 218.5551 | 2     | 1     | 2      |
| 3.  | 2  | 1        | 146.7606 | 1     | 0     | 1      |
| 4.  | 2  | 1        | 133.3201 | 2     | 1     | 2      |
| 5.  | 3  | 1        | 160.6548 | 1     | 0     | 1      |
| 6.  | 3  | 1        | 126.0635 | 2     | 1     | 2      |
| 7.  | 4  | 1        | 157.8622 | 1     | 0     | 1      |
| 8.  | 4  | 1        | 96.17461 | 2     | 1     | 2      |
| 9.  | 5  | 1        | 133.6957 | 1     | 0     | 1      |
| 10. | 5  | 1        | 188.9038 | 2     | 1     | 2      |
| 11. | 7  | 1        | 160.639  | 1     | 0     | 1      |
| 12. | 7  | 1        | 223.6922 | 2     | 1     | 2      |
| 13. | 8  | 1        | 131.2604 | 1     | 0     | 1      |
| 14. | 8  | 1        | 104.0139 | 2     | 1     | 2      |
| 15. | 9  | 1        | 168.5186 | 1     | 0     | 1      |
| 16. | 9  | 1        | 237.8962 | 2     | 1     | 2      |
| 17. | 10 | 2        | 137.0627 | 2     | 0     | 1      |
| 18. | 10 | 2        | 139.7382 | 1     | 2     | 2      |
| 19. | 12 | 2        | 153.4038 | 2     | 0     | 1      |
| 20. | 12 | 2        | 202.3942 | 1     | 2     | 2      |
| 21. | 13 | 2        | 163.4593 | 2     | 0     | 1      |
| 22. | 13 | 2        | 136.7848 | 1     | 2     | 2      |
| 23. | 14 | 2        | 146.0462 | 2     | 0     | 1      |
| 24. | 14 | 2        | 104.5191 | 1     | 2     | 2      |
| 25. | 15 | 2        | 158.1457 | 2     | 0     | 1      |
| 26. | 15 | 2        | 165.8654 | 1     | 2     | 2      |
| 27. | 18 | 2        | 147.1977 | 2     | 0     | 1      |
| 28. | 18 | 2        | 139.235  | 1     | 2     | 2      |
| 29. | 19 | 2        | 164.9988 | 2     | 0     | 1      |
| 30. | 19 | 2        | 166.2391 | 1     | 2     | 2      |
| 31. | 20 | 2        | 145.3823 | 2     | 0     | 1      |
| 32. | 20 | 2        | 158.5146 | 1     | 2     | 2      |

As an aside, crossover designs do not require that each subject receive each treatment, but if they do, the crossover design is referred to as a complete crossover design.

The last dataset is organized in a manner that can be analyzed with Stata. To fit an ANOVA model to these data, we can use `anova` or `pkcross`. To conduct equivalence tests, we can use `pkequiv`. This example is further analyzed in [\[R\] pkcross](#) and [\[R\] pkequiv](#).



## References

- Chow, S. C., and J. P. Liu. 2009. *Design and Analysis of Bioavailability and Bioequivalence Studies*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC.
- Ratkowsky, D. A., M. A. Evans, and J. R. Alldredge. 1993. *Cross-over Experiments: Design, Analysis, and Application*. New York: Dekker.

Syntax

```
pkcollapse time concentration [if] , id(id_var) [options]
```

| options                  | description                                                     |
|--------------------------|-----------------------------------------------------------------|
| Main                     |                                                                 |
| *id(id_var)              | subject ID variable                                             |
| stat(measures)           | create specified measures; default is all                       |
| trapezoid                | use trapezoidal rule; default is cubic splines                  |
| fit(#)                   | use # points to estimate AUC <sub>0,∞</sub> ; default is fit(3) |
| keep(varlist)            | keep variables in varlist                                       |
| force                    | force collapse                                                  |
| nodots                   | suppress dots during calculation                                |
| *id(id_var) is required. |                                                                 |

| measures | description                                                                        |
|----------|------------------------------------------------------------------------------------|
| auc      | area under the concentration-time curve (AUC <sub>0,∞</sub> )                      |
| aucline  | area under the concentration-time curve from 0 to ∞ using a linear extension       |
| aucexp   | area under the concentration-time curve from 0 to ∞ using an exponential extension |
| auclog   | area under the log-concentration-time curve extended with a linear fit             |
| half     | half-life of the drug                                                              |
| ke       | elimination rate                                                                   |
| cmax     | maximum concentration                                                              |
| tmax     | time at last concentration                                                         |
| tomc     | time of maximum concentration                                                      |

Menu

Statistics > Epidemiology and related > Other > Generate pharmacokinetic measurement dataset

Description

pkcollapse generates new variables with the pharmacokinetic summary measures of interest. pkcollapse is one of the pk commands. Please read [\[R\] pk](#) before reading this entry.

# Options

Main

`id(id_var)` is required and specifies the variable that contains the subject ID over which `pkcollapse` is to operate.

`stat(measures)` specifies the measures to be generated. The default is to generate all the measures.

`trapezoid` tells Stata to use the trapezoidal rule when calculating the AUC. The default is to use cubic splines, which give better results for most functions. When the curve is irregular, `trapezoid` may give better results.

`fit(#)` specifies the number of points to use in estimating the  $AUC_{0,\infty}$ . The default is `fit(3)`, the last three points. This number should be viewed as a minimum; the appropriate number of points will depend on your data.

`keep(varlist)` specifies the variables to be kept during the collapse. Variables not specified with the `keep()` option will be dropped. When `keep()` is specified, the keep variables are checked to ensure that all values of the variables are the same within *id\_var*.

`force` forces the collapse, even when the values of the `keep()` variables are different within the *id\_var*.

`nodots` suppresses the display of dots during calculation.

# Remarks

`pkcollapse` generates all the summary pharmacokinetic measures.

## ► Example 1

We demonstrate the use of `pkcollapse` with the data described in [R] [pk](#). We have drug concentration data on 15 subjects. Each subject is measured at 13 time points over a 32-hour period. Some of the records are

```
. use http://www.stata-press.com/data/r11/pkdata
. list, sep(0)
```

|      | id               | seq | time | concA    | concB    |
|------|------------------|-----|------|----------|----------|
| 1.   | 1                | 1   | 0    | 0        | 0        |
| 2.   | 1                | 1   | .5   | 3.073403 | 3.712592 |
| 3.   | 1                | 1   | 1    | 5.188444 | 6.230602 |
| 4.   | 1                | 1   | 1.5  | 5.898577 | 7.885944 |
| 5.   | 1                | 1   | 2    | 5.096378 | 9.241735 |
| 6.   | 1                | 1   | 3    | 6.094085 | 13.10507 |
|      | (output omitted) |     |      |          |          |
| 14.  | 2                | 1   | 0    | 0        | 0        |
| 15.  | 2                | 1   | .5   | 2.48462  | .9209593 |
| 16.  | 2                | 1   | 1    | 4.883569 | 5.925818 |
| 17.  | 2                | 1   | 1.5  | 7.253442 | 8.710549 |
| 18.  | 2                | 1   | 2    | 5.849345 | 10.90552 |
| 19.  | 2                | 1   | 3    | 6.761085 | 8.429898 |
|      | (output omitted) |     |      |          |          |
| 207. | 20               | 2   | 24   | 4.673281 | 6.059818 |
| 208. | 20               | 2   | 32   | 3.487347 | 5.213639 |

Although `pksumm` allows us to view all the pharmacokinetic measures, we can create a dataset with the measures by using `pkcollapse`.

```
. pkcollapse time concA concB, id(id) stat(auc) keep(seq)
.....
. list, sep(8) abbrev(10)
```

|     | id | seq | auc_concA | auc_concB |
|-----|----|-----|-----------|-----------|
| 1.  | 1  | 1   | 150.9643  | 218.5551  |
| 2.  | 2  | 1   | 146.7606  | 133.3201  |
| 3.  | 3  | 1   | 160.6548  | 126.0635  |
| 4.  | 4  | 1   | 157.8622  | 96.17461  |
| 5.  | 5  | 1   | 133.6957  | 188.9038  |
| 6.  | 7  | 1   | 160.639   | 223.6922  |
| 7.  | 8  | 1   | 131.2604  | 104.0139  |
| 8.  | 9  | 1   | 168.5186  | 237.8962  |
| 9.  | 10 | 2   | 137.0627  | 139.7382  |
| 10. | 12 | 2   | 153.4038  | 202.3942  |
| 11. | 13 | 2   | 163.4593  | 136.7848  |
| 12. | 14 | 2   | 146.0462  | 104.5191  |
| 13. | 15 | 2   | 158.1457  | 165.8654  |
| 14. | 18 | 2   | 147.1977  | 139.235   |
| 15. | 19 | 2   | 164.9988  | 166.2391  |
| 16. | 20 | 2   | 145.3823  | 158.5146  |

The resulting dataset, which we will call `pkdata2`, contains 1 observation per subject. This dataset is in wide format. If we want to use `pkcross` or `pkequiv`, we must transform these data to long format, which we do in the [last example](#) of [\[R\] pkshape](#).



## Methods and formulas

`pkcollapse` is implemented as an ado-file.

The statistics generated by `pkcollapse` are described in [\[R\] pkexamine](#).

## Also see

[\[R\] pk](#) — Pharmacokinetic (biopharmaceutical) data

Syntax

```
pkcross outcome [if] [in] [, options]
```

| options                         | description                                                                                                                     |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| Model                           |                                                                                                                                 |
| <code>sequence(varname)</code>  | sequence variable; default is <code>sequence(sequence)</code>                                                                   |
| <code>treatment(varname)</code> | treatment variable; default is <code>treatment(treat)</code>                                                                    |
| <code>period(varname)</code>    | period variable; default is <code>period(period)</code>                                                                         |
| <code>id(varname)</code>        | ID variable                                                                                                                     |
| <code>carryover(varname)</code> | name of carryover variable; default is <code>carryover(carry)</code>                                                            |
| <code>carryover(none)</code>    | omit carryover effects from model; default is <code>carryover(carry)</code>                                                     |
| <code>model(string)</code>      | specify the model to fit                                                                                                        |
| <code>sequential</code>         | estimate sequential instead of partial sums of squares                                                                          |
| Parameterization                |                                                                                                                                 |
| <code>param(3)</code>           | estimate mean and the period, treatment, and sequence effects; assume no carryover effects exist; the default                   |
| <code>param(1)</code>           | estimate mean and the period, treatment, and carryover effects; assume no sequence effects exist                                |
| <code>param(2)</code>           | estimate mean, period and treatment effects, and period-by-treatment interaction; assume no sequence or carryover effects exist |
| <code>param(4)</code>           | estimate mean, period and treatment effects, and period-by-treatment interaction; assume no period or crossover effects exist   |

Menu

Statistics > Epidemiology and related > Other > Analyze crossover experiments

Description

pkcross analyzes data from a crossover design experiment. When analyzing pharmaceutical trial data, if the treatment, carryover, and sequence variables are known, the omnibus test for separability of the treatment and carryover effects is calculated.

pkcross is one of the pk commands. Please read [R] **pk** before reading this entry.

Options

Model

`sequence(varname)` specifies the variable that contains the sequence in which the treatment was administered. If this option is not specified, `sequence(sequence)` is assumed.



`treatment(varname)` specifies the variable that contains the treatment information. If this option is not specified, `treatment(treat)` is assumed.

`period(varname)` specifies the variable that contains the period information. If this option is not specified, `period(period)` is assumed.

`id(varname)` specifies the variable that contains the subject identifiers. If this option is not specified, `id(id)` is assumed.

`carryover(varname|none)` specifies the variable that contains the carryover information. If `carry(none)` is specified, the carryover effects are omitted from the model. If this option is not specified, `carryover(carry)` is assumed.

`model(string)` specifies the model to be fit. For higher-order crossover designs, this option can be useful if you want to fit a model other than the default. However, `anova` (see [R] [anova](#)) can also be used to fit a crossover model. The default model for higher-order crossover designs is outcome predicted by sequence, period, treatment, and carryover effects. By default, the model statement is `model(sequence period treat carry)`.

`sequential` specifies that sequential sums of squares be estimated.

#### Parameterization

`param(#)` specifies which of the four parameterizations to use for the analysis of a  $2 \times 2$  crossover experiment. This option is ignored with higher-order crossover designs. The default is `param(3)`. See the technical note for  $2 \times 2$  crossover designs for more details.

`param(3)` estimates the overall mean, the period effects, the treatment effects, and the sequence effects, assuming that no carryover effects exist. This is the default parameterization.

`param(1)` estimates the overall mean, the period effects, the treatment effects, and the carryover effects, assuming that no sequence effects exist.

`param(2)` estimates the overall mean, the period effects, the treatment effects, and the period-by-treatment interaction, assuming that no sequence or carryover effects exist.

`param(4)` estimates the overall mean, the sequence effects, the treatment effects, and the sequence-by-treatment interaction, assuming that no period or crossover effects exist. When the sequence by treatment is equivalent to the period effect, this reduces to the third parameterization.

## Remarks

`pkcross` is designed to analyze crossover experiments. Use `pkshape` first to reshape your data; see [R] [pkshape](#). `pkcross` assumes that the data were reshaped by `pkshape` or are organized in the same manner as produced with `pkshape`. Washout periods are indicated by the number 0. See the technical note in this entry for more information on analyzing  $2 \times 2$  crossover experiments.

### □ Technical note

The  $2 \times 2$  crossover design cannot be used to estimate more than four parameters because there are only four pieces of information (the four cell means) collected. `pkcross` uses ANOVA models to analyze the data, so one of the four parameters must be the overall mean of the model, leaving just 3 degrees of freedom to estimate the remaining effects (period, sequence, treatment, and carryover). Thus the model is overparameterized. Estimation of treatment and carryover effects requires the assumption of either no period effects or no sequence effects. Some researchers maintain that it estimating carryover effects at the expense of other effects is a bad idea. This is a limitation of this

design. pkcross implements four parameterizations for this model. They are numbered sequentially from one to four and are described in [Options](#).



➤ Example 1

Consider the example data published in [Chow and Liu \(2009, 71\)](#) and described in [\[R\] pkshape](#). We have entered and reshaped the data with pkshape and have variables that identify the subjects, periods, treatments, sequence, and carryover treatment. To compute the ANOVA table, use pkcross:

```
. use http://www.stata-press.com/data/r11/chowliu
. pkshape id seq period1 period2, order(ab ba)
. pkcross outcome

 sequence variable = sequence
 period variable = period
 treatment variable = treat
 carryover variable = carry
 id variable = id

Analysis of variance (ANOVA) for a 2x2 crossover study
Source of Variation	SS df MS F Prob > F
Intersubjects
 Sequence effect | 276.00 1 276.00 0.37 0.5468
 Residuals | 16211.49 22 736.89 4.41 0.0005
Intrasubjects
 Treatment effect | 62.79 1 62.79 0.38 0.5463
 Period effect | 35.97 1 35.97 0.22 0.6474
 Residuals | 3679.43 22 167.25
Total | 20265.68 47
Omnibus measure of separability of treatment and carryover = 29.2893%
```

There is evidence of intersubject variability, but there are no other significant effects. The omnibus test for separability is a measure reflecting the degree to which the study design allows the treatment effects to be estimated independently of the carryover effects. The measure of separability of the treatment and carryover effects indicates approximately 29% separability, which can be interpreted as the degree to which the treatment and carryover effects are orthogonal. This is a characteristic of the design of the study. For a complete discussion, see [Ratkowsky, Evans, and Alldredge \(1993\)](#). Compared to the output in [Chow and Liu \(2009\)](#), the sequence effect is mislabeled as a carryover effect. See [Ratkowsky, Evans, and Alldredge \(1993, sec. 3.2\)](#) for a complete discussion of the mislabeling.

By specifying `param(1)`, we obtain parameterization 1 for this model.

```
. pkcross outcome, param(1)
```

```
sequence variable = sequence
period variable = period
treatment variable = treat
carryover variable = carry
id variable = id
```

| Source of Variation | Partial SS | df | MS     | F    | Prob > F |
|---------------------|------------|----|--------|------|----------|
| Treatment effect    | 301.04     | 1  | 301.04 | 0.67 | 0.4189   |
| Period effect       | 255.62     | 1  | 255.62 | 0.57 | 0.4561   |
| Carryover effect    | 276.00     | 1  | 276.00 | 0.61 | 0.4388   |
| Residuals           | 19890.92   | 44 | 452.07 |      |          |
| Total               | 20265.68   | 47 |        |      |          |

Omnibus measure of separability of treatment and carryover = 29.2893%

◀

## ► Example 2

Consider the case of a two-treatment, four-sequence, two-period crossover design. This design is commonly referred to as Balaam's design (Balaam 1968). Ratkowsky, Evans, and Alldredge (1993, 140) published the following data from an amantadine trial, originally published by Taka and Armitage (1983):

```
. use http://www.stata-press.com/data/r11/balaam, clear
. list, sep(0)
```

|     | id | seq | period1 | period2 | period3 |
|-----|----|-----|---------|---------|---------|
| 1.  | 1  | -ab | 9       | 8.75    | 8.75    |
| 2.  | 2  | -ab | 12      | 10.5    | 9.75    |
| 3.  | 3  | -ab | 17      | 15      | 18.5    |
| 4.  | 4  | -ab | 21      | 21      | 21.5    |
| 5.  | 1  | -ba | 23      | 22      | 18      |
| 6.  | 2  | -ba | 15      | 15      | 13      |
| 7.  | 3  | -ba | 13      | 14      | 13.75   |
| 8.  | 4  | -ba | 24      | 22.75   | 21.5    |
| 9.  | 5  | -ba | 18      | 17.75   | 16.75   |
| 10. | 1  | -aa | 14      | 12.5    | 14      |
| 11. | 2  | -aa | 27      | 24.25   | 22.5    |
| 12. | 3  | -aa | 19      | 17.25   | 16.25   |
| 13. | 4  | -aa | 30      | 28.25   | 29.75   |
| 14. | 1  | -bb | 21      | 20      | 19.51   |
| 15. | 2  | -bb | 11      | 10.5    | 10      |
| 16. | 3  | -bb | 20      | 19.5    | 20.75   |
| 17. | 4  | -bb | 25      | 22.5    | 23.5    |

The sequence identifier must be a string with zeros to indicate washout or baseline periods, or a number. If the sequence identifier is numeric, the `order` option must be specified with `pkshape`. If the sequence identifier is a string, `pkshape` will create sequence, period, and treatment identifiers without the `order` option. In this example, the dash is used to indicate a baseline period, which is an invalid code for this purpose. As a result, the data must be encoded; see [D] [encode](#).

```
. encode seq, gen(num_seq)
. pkshape id num_seq period1 period2 period3, order(0aa 0ab 0ba 0bb)
. pkcross outcome, se
 sequence variable = sequence
 period variable = period
 treatment variable = treat
 carryover variable = carry
 id variable = id
```

| Analysis of variance (ANOVA) for a crossover study |         |    |       |       |          |
|----------------------------------------------------|---------|----|-------|-------|----------|
| Source of Variation                                | SS      | df | MS    | F     | Prob > F |
| Intersubjects                                      |         |    |       |       |          |
| Sequence effect                                    | 285.82  | 3  | 95.27 | 1.01  | 0.4180   |
| Residuals                                          | 1221.49 | 13 | 93.96 | 59.96 | 0.0000   |
| Intrasubjects                                      |         |    |       |       |          |
| Period effect                                      | 15.13   | 2  | 7.56  | 6.34  | 0.0048   |
| Treatment effect                                   | 8.48    | 1  | 8.48  | 8.86  | 0.0056   |
| Carryover effect                                   | 0.11    | 1  | 0.11  | 0.12  | 0.7366   |
| Residuals                                          | 29.56   | 30 | 0.99  |       |          |
| Total                                              | 1560.59 | 50 |       |       |          |

Omnibus measure of separability of treatment and carryover = 64.6447%

In this example, the sequence specifier used dashes instead of zeros to indicate a baseline period during which no treatment was given. For pkcross to work, we need to encode the string sequence variable and then use the order option with pkshape. A word of caution: encode does not necessarily choose the first sequence to be sequence 1, as in this example. Always double-check the sequence numbering when using encode.



### ► Example 3

Continuing with the example from [R] **pkshape**, we fit an ANOVA model.

```
. use http://www.stata-press.com/data/r11/pkdata3, clear
. list, sep(8)
```

|     | id | sequence | outcome  | treat | carry | period |
|-----|----|----------|----------|-------|-------|--------|
| 1.  | 1  | 1        | 150.9643 | A     | 0     | 1      |
| 2.  | 2  | 1        | 146.7606 | A     | 0     | 1      |
| 3.  | 3  | 1        | 160.6548 | A     | 0     | 1      |
| 4.  | 4  | 1        | 157.8622 | A     | 0     | 1      |
| 5.  | 5  | 1        | 133.6957 | A     | 0     | 1      |
| 6.  | 7  | 1        | 160.639  | A     | 0     | 1      |
| 7.  | 8  | 1        | 131.2604 | A     | 0     | 1      |
| 8.  | 9  | 1        | 168.5186 | A     | 0     | 1      |
|     |    |          |          |       |       |        |
| 9.  | 10 | 2        | 137.0627 | B     | 0     | 1      |
| 10. | 12 | 2        | 153.4038 | B     | 0     | 1      |
| 11. | 13 | 2        | 163.4593 | B     | 0     | 1      |
| 12. | 14 | 2        | 146.0462 | B     | 0     | 1      |
| 13. | 15 | 2        | 158.1457 | B     | 0     | 1      |
| 14. | 18 | 2        | 147.1977 | B     | 0     | 1      |
| 15. | 19 | 2        | 164.9988 | B     | 0     | 1      |
| 16. | 20 | 2        | 145.3823 | B     | 0     | 1      |
|     |    |          |          |       |       |        |
| 17. | 1  | 1        | 218.5551 | B     | A     | 2      |
| 18. | 2  | 1        | 133.3201 | B     | A     | 2      |
| 19. | 3  | 1        | 126.0635 | B     | A     | 2      |
| 20. | 4  | 1        | 96.17461 | B     | A     | 2      |
| 21. | 5  | 1        | 188.9038 | B     | A     | 2      |
| 22. | 7  | 1        | 223.6922 | B     | A     | 2      |
| 23. | 8  | 1        | 104.0139 | B     | A     | 2      |
| 24. | 9  | 1        | 237.8962 | B     | A     | 2      |
|     |    |          |          |       |       |        |
| 25. | 10 | 2        | 139.7382 | A     | B     | 2      |
| 26. | 12 | 2        | 202.3942 | A     | B     | 2      |
| 27. | 13 | 2        | 136.7848 | A     | B     | 2      |
| 28. | 14 | 2        | 104.5191 | A     | B     | 2      |
| 29. | 15 | 2        | 165.8654 | A     | B     | 2      |
| 30. | 18 | 2        | 139.235  | A     | B     | 2      |
| 31. | 19 | 2        | 166.2391 | A     | B     | 2      |
| 32. | 20 | 2        | 158.5146 | A     | B     | 2      |

(Continued on next page)

The ANOVA model is fit using pkcross:

```
. pkcross outcome
 sequence variable = sequence
 period variable = period
 treatment variable = treat
 carryover variable = carry
 id variable = id
```

| Analysis of variance (ANOVA) for a 2x2 crossover study |          |    |         |      |          |
|--------------------------------------------------------|----------|----|---------|------|----------|
| Source of Variation                                    | SS       | df | MS      | F    | Prob > F |
| Intersubjects                                          |          |    |         |      |          |
| Sequence effect                                        | 378.04   | 1  | 378.04  | 0.29 | 0.5961   |
| Residuals                                              | 17991.26 | 14 | 1285.09 | 1.40 | 0.2691   |
| Intrasubjects                                          |          |    |         |      |          |
| Treatment effect                                       | 455.04   | 1  | 455.04  | 0.50 | 0.4931   |
| Period effect                                          | 419.47   | 1  | 419.47  | 0.46 | 0.5102   |
| Residuals                                              | 12860.78 | 14 | 918.63  |      |          |
| Total                                                  | 32104.59 | 31 |         |      |          |

Omnibus measure of separability of treatment and carryover = 29.2893%

➤ Example 4

Consider the case of a six-treatment crossover trial in which the squares are not variance balanced. The following dataset is from a partially balanced crossover trial published by [Patterson and Lucas \(1962\)](#) and reproduced in [Ratkowsky, Evans, and Alldredge \(1993, 231\)](#):

```
. use http://www.stata-press.com/data/r11/nobalance
. list, sep(4)
```

|     | cow | seq  | period1 | period2 | period3 | period4 | block |
|-----|-----|------|---------|---------|---------|---------|-------|
| 1.  | 1   | adbe | 38.7    | 37.4    | 34.3    | 31.3    | 1     |
| 2.  | 2   | baed | 48.9    | 46.9    | 42      | 39.6    | 1     |
| 3.  | 3   | ebda | 34.6    | 32.3    | 28.5    | 27.1    | 1     |
| 4.  | 4   | deab | 35.2    | 33.5    | 28.4    | 25.1    | 1     |
| 5.  | 1   | dafc | 32.9    | 33.1    | 27.5    | 25.1    | 2     |
| 6.  | 2   | fdca | 30.4    | 29.5    | 26.7    | 23.1    | 2     |
| 7.  | 3   | cfad | 30.8    | 29.3    | 26.4    | 23.2    | 2     |
| 8.  | 4   | acdf | 25.7    | 26.1    | 23.4    | 18.7    | 2     |
| 9.  | 1   | efbc | 25.4    | 26      | 23.9    | 19.9    | 3     |
| 10. | 2   | becf | 21.8    | 23.9    | 21.7    | 17.6    | 3     |
| 11. | 3   | fceb | 21.4    | 22      | 19.4    | 16.6    | 3     |
| 12. | 4   | cbfe | 22.8    | 21      | 18.6    | 16.1    | 3     |

When there is no variance balance in the design, a square or blocking variable is needed to indicate in which treatment cell a sequence was observed, but the mechanical steps are the same.

```
. pkshape cow seq period1 period2 period3 period4
. pkcross outcome, model(block cow|block period|block treat carry) se
```

|              | Number of obs = | 48      | R-squared =     | 0.9965  |          |
|--------------|-----------------|---------|-----------------|---------|----------|
|              | Root MSE =      | .740408 | Adj R-squared = | 0.9903  |          |
| Source       | Seq. SS         | df      | MS              | F       | Prob > F |
| Model        | 2650.1331       | 30      | 88.3377701      | 161.14  | 0.0000   |
| block        | 1607.01128      | 2       | 803.505642      | 1465.71 | 0.0000   |
| cow block    | 628.706274      | 9       | 69.8562527      | 127.43  | 0.0000   |
| period block | 408.031253      | 9       | 45.3368059      | 82.70   | 0.0000   |
| treat        | 2.50000057      | 5       | .500000114      | 0.91    | 0.4964   |
| carry        | 3.88428906      | 5       | .776857812      | 1.42    | 0.2680   |
| Residual     | 9.31945887      | 17      | .548203463      |         |          |
| Total        | 2659.45256      | 47      | 56.584097       |         |          |

When the model statement is used and the omnibus measure of separability is desired, specify the variables in the `treatment()`, `carryover()`, and `sequence()` options to `pkcross`.

◀

## Methods and formulas

`pkcross` is implemented as an ado-file.

`pkcross` uses ANOVA to fit models for crossover experiments; see [\[R\] anova](#).

The omnibus measure of separability is

$$S = 100(1 - V)\%$$

where  $V$  is Cramér's  $V$  and is defined as

$$V = \left\{ \frac{\frac{\chi^2}{N}}{\min(r-1, c-1)} \right\}^{\frac{1}{2}}$$

The  $\chi^2$  is calculated as

$$\chi^2 = \sum_i \sum_j \left\{ \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \right\}$$

where  $O$  and  $E$  are the observed and expected counts in a table of the number of times each treatment is followed by the other treatments.

## References

- Balaam, L. N. 1968. A two-period design with  $t^2$  experimental units. *Biometrics* 24: 61–73.
- Chow, S. C., and J. P. Liu. 2009. *Design and Analysis of Bioavailability and Bioequivalence Studies*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC.

- Kutner, M. H., C. J. Nachtsheim, J. Neter, and W. Li. 2005. *Applied Linear Statistical Models*. 5th ed. New York: McGraw-Hill/Irwin.
- Patterson, H. D., and H. L. Lucas. 1962. Change-over designs. Technical Bulletin 147, North Carolina Agricultural Experiment Station and the USDA.
- Ratkowsky, D. A., M. A. Evans, and J. R. Alldredge. 1993. *Cross-over Experiments: Design, Analysis, and Application*. New York: Dekker.
- Taka, M. T., and P. Armitage. 1983. Autoregressive models in clinical trials. *Communications in Statistics, Theory and Methods* 12: 865–876.

## Also see

[\[R\] pk](#) — Pharmacokinetic (biopharmaceutical) data



Syntax

pkequiv *outcome treatment period sequence id* [*if*] [*in*] [*, options*]

| <i>options</i>                   | description                                                       |
|----------------------------------|-------------------------------------------------------------------|
| Options                          |                                                                   |
| <u>compare</u> ( <i>string</i> ) | compare the two specified values of the treatment variable        |
| <u>limit</u> (#)                 | equivalence limit (between 0.10 and 0.99); default is 0.2         |
| <u>level</u> (#)                 | set confidence level; default is <code>level(90)</code>           |
| <u>fieller</u>                   | calculate confidence interval by Fieller's theorem                |
| <u>symmetric</u>                 | calculate symmetric equivalence interval                          |
| <u>anderson</u>                  | Anderson and Hauck hypothesis test for bioequivalence             |
| <u>tost</u>                      | two one-sided hypothesis tests for bioequivalence                 |
| <u>noboot</u>                    | do not estimate probability that CI lies within confidence limits |

Menu

Statistics > Epidemiology and related > Other > Bioequivalence tests

Description

pkequiv performs bioequivalence testing for two treatments. By default, pkequiv calculates a standard confidence interval symmetric about the difference between the two treatment means. pkequiv also calculates confidence intervals symmetric about zero and intervals based on Fieller's theorem. Also, pkequiv can perform interval hypothesis tests for bioequivalence.

pkequiv is one of the pk commands. Please read [\[R\]](#) **pk** before reading this entry.

Options

Options

- `compare(string)` specifies the two treatments to be tested for equivalence. Sometimes there may be more than two treatments, but the equivalence can be determined only between any two treatments.
- `limit(#)` specifies the equivalence limit. The default is 0.2. The equivalence limit can be changed only symmetrically; that is, it is not possible to have a 0.15 lower limit and a 0.2 upper limit in the same test.
- `level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(90)`. This setting is not controlled by the `set level` command.
- `fieller` specifies that an equivalence interval on the basis of Fieller's theorem be calculated.
- `symmetric` specifies that a symmetric equivalence interval be calculated.
- `anderson` specifies that the [Anderson and Hauck \(1983\)](#) hypothesis test for bioequivalence be computed. This option is ignored when calculating equivalence intervals based on Fieller's theorem or when calculating a confidence interval that is symmetric about zero.

`tost` specifies that the two one-sided hypothesis tests for bioequivalence be computed. This option is ignored when calculating equivalence intervals based on Fieller’s theorem or when calculating a confidence interval that is symmetric about zero.

`noboot` prevents the estimation of the probability that the confidence interval lies within the confidence limits. If this option is not specified, this probability is estimated by resampling the data.

Remarks

`pkequiv` is designed to conduct tests for bioequivalence based on data from a crossover experiment. `pkequiv` requires that the user specify the *outcome*, *treatment*, *period*, *sequence*, and *id* variables. The data must be in the same format as that produced by `pkshape`; see [R] `pkshape`.

➤ Example 1

Continuing with `example 4` from [R] `pkshape`, we will conduct equivalence testing.

```
. use http://www.stata-press.com/data/r11/pkdata3
. list, sep(4)
```

|     | id | sequence | outcome  | treat | carry | period |
|-----|----|----------|----------|-------|-------|--------|
| 1.  | 1  | 1        | 150.9643 | A     | 0     | 1      |
| 2.  | 2  | 1        | 146.7606 | A     | 0     | 1      |
| 3.  | 3  | 1        | 160.6548 | A     | 0     | 1      |
| 4.  | 4  | 1        | 157.8622 | A     | 0     | 1      |
| 5.  | 5  | 1        | 133.6957 | A     | 0     | 1      |
| 6.  | 7  | 1        | 160.639  | A     | 0     | 1      |
| 7.  | 8  | 1        | 131.2604 | A     | 0     | 1      |
| 8.  | 9  | 1        | 168.5186 | A     | 0     | 1      |
| 9.  | 10 | 2        | 137.0627 | B     | 0     | 1      |
| 10. | 12 | 2        | 153.4038 | B     | 0     | 1      |
| 11. | 13 | 2        | 163.4593 | B     | 0     | 1      |
| 12. | 14 | 2        | 146.0462 | B     | 0     | 1      |
| 13. | 15 | 2        | 158.1457 | B     | 0     | 1      |
| 14. | 18 | 2        | 147.1977 | B     | 0     | 1      |
| 15. | 19 | 2        | 164.9988 | B     | 0     | 1      |
| 16. | 20 | 2        | 145.3823 | B     | 0     | 1      |
| 17. | 1  | 1        | 218.5551 | B     | A     | 2      |
| 18. | 2  | 1        | 133.3201 | B     | A     | 2      |
| 19. | 3  | 1        | 126.0635 | B     | A     | 2      |
| 20. | 4  | 1        | 96.17461 | B     | A     | 2      |
| 21. | 5  | 1        | 188.9038 | B     | A     | 2      |
| 22. | 7  | 1        | 223.6922 | B     | A     | 2      |
| 23. | 8  | 1        | 104.0139 | B     | A     | 2      |
| 24. | 9  | 1        | 237.8962 | B     | A     | 2      |
| 25. | 10 | 2        | 139.7382 | A     | B     | 2      |
| 26. | 12 | 2        | 202.3942 | A     | B     | 2      |
| 27. | 13 | 2        | 136.7848 | A     | B     | 2      |
| 28. | 14 | 2        | 104.5191 | A     | B     | 2      |
| 29. | 15 | 2        | 165.8654 | A     | B     | 2      |
| 30. | 18 | 2        | 139.235  | A     | B     | 2      |
| 31. | 19 | 2        | 166.2391 | A     | B     | 2      |
| 32. | 20 | 2        | 158.5146 | A     | B     | 2      |

Now we can conduct a bioequivalence test between  $\text{treat} = A$  and  $\text{treat} = B$ .

```
. set seed 1
. pkequiv outcome treat period seq id
 Classic confidence interval for bioequivalence
```

|             | [equivalence limits] |        | [ test limits ] |          |
|-------------|----------------------|--------|-----------------|----------|
| difference: | -30.296              | 30.296 | -11.332         | 26.416   |
| ratio:      | 80%                  | 120%   | 92.519%         | 117.439% |

```

probability test limits are within equivalence limits = 0.6410
note: reference treatment = 1
```

The default output for `pkequiv` shows a confidence interval for the difference of the means (test limits), the ratio of the means, and the federal equivalence limits. The classic confidence interval can be constructed around the difference between the average measure of effect for the two drugs or around the ratio of the average measure of effect for the two drugs. `pkequiv` reports both the difference measure and the ratio measure. For these data, U.S. federal government regulations state that the confidence interval for the difference must be entirely contained within the range  $[-30.296, 30.296]$  and between 80% and 120% for the ratio. Here the test limits are within the equivalence limits. Although the test limits are inside the equivalence limits, there is only a 64% assurance that the observed confidence interval will be within the equivalence limits in the long run. This is an interesting case because, although this sample shows bioequivalence, the evaluation of the long-run performance indicates possible problems. These fictitious data were generated with high intersubject variability, which causes poor long-run performance.

If we conduct a bioequivalence test with the data published in [Chow and Liu \(2009, 71\)](#), which we introduced in [\[R\] pk](#) and fully described in [\[R\] pkshape](#), we observe that the probability that the test limits are within the equivalence limits is high.

```
. use http://www.stata-press.com/data/r11/chowliu2
. set seed 1
. pkequiv outcome treat period seq id
 Classic confidence interval for bioequivalence
```

|             | [equivalence limits] |        | [ test limits ] |          |
|-------------|----------------------|--------|-----------------|----------|
| difference: | -16.512              | 16.512 | -8.698          | 4.123    |
| ratio:      | 80%                  | 120%   | 89.464%         | 104.994% |

```

probability test limits are within equivalence limits = 0.9980
note: reference treatment = 1
```

For these data, the test limits are well within the equivalence limits, and the probability that the test limits are within the equivalence limits is 99.8%.

◀

(Continued on next page)

➤ Example 2

We compute a confidence interval that is symmetric about zero:

```
. pkequiv outcome treat period seq id, symmetric
Westlake's symmetric confidence interval for bioequivalence
```

|                   | [Equivalence limits] |        | [ Test mean ] |
|-------------------|----------------------|--------|---------------|
| Test formulation: | 75.145               | 89.974 | 80.272        |

```
note: reference treatment = 1
```

The reported equivalence limit is constructed symmetrically about the reference mean, which is equivalent to constructing a confidence interval symmetric about zero for the difference in the two drugs. In the output above, we see that the test formulation mean of 80.272 is within the equivalence limits, indicating that the test drug is bioequivalent to the reference drug.

pkequiv displays interval hypothesis tests of bioequivalence if you specify the `tost` or the `anderson` option, or both. For example,

```
. set seed 1
. pkequiv outcome treat period seq id, tost anderson
Classic confidence interval for bioequivalence
```

|             | [equivalence limits] |        | [ test limits ] |          |
|-------------|----------------------|--------|-----------------|----------|
| difference: | -16.512              | 16.512 | -8.698          | 4.123    |
| ratio:      | 80%                  | 120%   | 89.464%         | 104.994% |

```
probability test limits are within equivalence limits = 0.9980
Schuirmann's two one-sided tests
```

|                        |        |           |       |
|------------------------|--------|-----------|-------|
| upper test statistic = | -5.036 | p-value = | 0.000 |
| lower test statistic = | 3.810  | p-value = | 0.001 |

```
Anderson and Hauck's test
```

|                           |        |                     |        |
|---------------------------|--------|---------------------|--------|
| noncentrality parameter = | 4.423  |                     |        |
| test statistic =          | -0.613 | empirical p-value = | 0.0005 |

```
note: reference treatment = 1
```

Both of Schuirmann's one-sided tests are highly significant, suggesting that the two drugs are bioequivalent. A similar conclusion is drawn from the Anderson and Hauck test of bioequivalence.

Saved results

pkequiv saves the following in `r()`:

- Scalars
- `r(stddev)` pooled-sample standard deviation of period differences from both sequences
  - `r(uci)` upper confidence interval for a classic interval
  - `r(lci)` lower confidence interval for a classic interval
  - `r(delta)` delta value used in calculating a symmetric confidence interval
  - `r(u3)` upper confidence interval for Fieller's confidence interval
  - `r(l3)` lower confidence interval for Fieller's confidence interval

## Methods and formulas

pkequiv is implemented as an ado-file.

The lower confidence interval for the difference in the two treatments for the classic shortest confidence interval is

$$L_1 = (\bar{Y}_T - \bar{Y}_R) - t_{(\alpha, n_1 + n_2 - 2)} \hat{\sigma}_d \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}$$

The upper limit is

$$U_1 = (\bar{Y}_T - \bar{Y}_R) + t_{(\alpha, n_1 + n_2 - 2)} \hat{\sigma}_d \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}$$

The limits for the ratio measure are

$$L_2 = \left( \frac{L_1}{\bar{Y}_R} + 1 \right) 100\%$$

and

$$U_2 = \left( \frac{U_1}{\bar{Y}_R} + 1 \right) 100\%$$

where  $\bar{Y}_T$  is the mean of the test formulation of the drug,  $\bar{Y}_R$  is the mean of the reference formulation of the drug, and  $t_{(\alpha, n_1 + n_2 - 2)}$  is the  $t$  distribution with  $n_1 + n_2 - 2$  degrees of freedom.  $\hat{\sigma}_d$  is the pooled sample variance of the period differences from both sequences, defined as

$$\hat{\sigma}_d = \frac{1}{n_1 + n_2 - 2} \sum_{k=1}^2 \sum_{i=1}^{n_k} (d_{ik} - \bar{d}_{.k})^2$$

The upper and lower limits for the symmetric confidence interval are  $\bar{Y}_R + \Delta$  and  $\bar{Y}_R - \Delta$ , where

$$\Delta = k_1 \hat{\sigma}_d \sqrt{\frac{1}{n_1} + \frac{1}{n_2}} - (\bar{Y}_T - \bar{Y}_R)$$

and (simultaneously)

$$\Delta = -k_2 \hat{\sigma}_d \sqrt{\frac{1}{n_1} + \frac{1}{n_2}} + 2 (\bar{Y}_T - \bar{Y}_R)$$

and  $k_1$  and  $k_2$  are computed iteratively to satisfy the above equalities and the condition

$$\int_{k_1}^{k_2} f(t) dt = 1 - 2\alpha$$

where  $f(t)$  is the probability density function of the  $t$  distribution with  $n_1 + n_2 - 2$  degrees of freedom.

See [Chow and Liu \(2009, 88–92\)](#) for details about calculating the confidence interval based on Fieller's theorem.

The two test statistics for the two one-sided tests of equivalence are

$$T_L = \frac{(\bar{Y}_T - \bar{Y}_R) - \theta_L}{\hat{\sigma}_d \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

and

$$T_U = \frac{(\bar{Y}_T - \bar{Y}_R) - \theta_U}{\hat{\sigma}_d \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

where  $-\theta_L = \theta_U$  and are the regulated confidence limits.

The logic of the Anderson and Hauck test is tricky; see [Chow and Liu \(2009\)](#) for a complete explanation. However, the test statistic is

$$T_{AH} = \frac{(\bar{Y}_T - \bar{Y}_R) - \left(\frac{\theta_L + \theta_U}{2}\right)}{\hat{\sigma}_d \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

and the noncentrality parameter is estimated by

$$\hat{\delta} = \frac{\theta_U - \theta_L}{2\hat{\sigma}_d \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

The empirical  $p$ -value is calculated as

$$p = F_t(|T_{AH}| - \hat{\delta}) - F_t(-|T_{AH}| - \hat{\delta})$$

where  $F_t$  is the cumulative distribution function of the  $t$  distribution with  $n_1 + n_2 - 2$  degrees of freedom.

## References

- Anderson, S., and W. W. Hauck. 1983. A new procedure for testing equivalence in comparative bioavailability and other clinical trials. *Communications in Statistics, Theory and Methods* 12: 2663–2692.
- Chow, S. C., and J. P. Liu. 2009. *Design and Analysis of Bioavailability and Bioequivalence Studies*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC.
- Fieller, E. C. 1954. Some problems in interval estimation. *Journal of the Royal Statistical Society, Series B* 16: 175–185.
- Kutner, M. H., C. J. Nachtsheim, J. Neter, and W. Li. 2005. *Applied Linear Statistical Models*. 5th ed. New York: McGraw–Hill/Irwin.
- Locke, C. S. 1984. An exact confidence interval from untransformed data for the ratio of two formulation means. *Journal of Pharmacokinetics and Biopharmaceutics* 12: 649–655.
- Schuurmann, D. J. 1989. Confidence intervals for the ratio of two means from a cross-over study. In *Proceedings of the Biopharmaceutical Section*, 121–126. Washington, DC: American Statistical Association.
- Westlake, W. J. 1976. Symmetrical confidence intervals for bioequivalence trials. *Biometrics* 32: 741–744.

## Also see

[R] **pk** — Pharmacokinetic (biopharmaceutical) data

Title

pkexamine — Calculate pharmacokinetic measures

Syntax

```
pkexamine time concentration [if] [in] [, options]
```

| options                                     | description                                                                               |
|---------------------------------------------|-------------------------------------------------------------------------------------------|
| Main                                        |                                                                                           |
| <code>fit(#)</code>                         | use # points to estimate $AUC_{0,\infty}$ ; default is <code>fit(3)</code>                |
| <code>trapezoid</code>                      | use trapezoidal rule; default is cubic splines                                            |
| <code>graph</code>                          | graph the AUC                                                                             |
| <code>line</code>                           | graph the linear extension                                                                |
| <code>log</code>                            | graph the log extension                                                                   |
| <code>exp(#)</code>                         | plot the exponential fit for the $AUC_{0,\infty}$                                         |
| AUC plot                                    |                                                                                           |
| <code>cline_options</code>                  | affect rendition of plotted points connected by lines                                     |
| <code>marker_options</code>                 | change look of markers (color, size, etc.)                                                |
| <code>marker_label_options</code>           | add marker labels; change look or position                                                |
| Add plots                                   |                                                                                           |
| <code>addplot(plot)</code>                  | add other plots to the generated graph                                                    |
| Y axis, X axis, Titles, Legend, Overall     |                                                                                           |
| <code>twoway_options</code>                 | any options other than <code>by()</code> documented in [G] <a href="#">twoway_options</a> |
| by is allowed; see [D] <a href="#">by</a> . |                                                                                           |

Menu

Statistics > Epidemiology and related > Other > Pharmacokinetic measures

Description

`pkexamine` calculates pharmacokinetic measures from time-and-concentration subject-level data. `pkexamine` computes and displays the maximum measured concentration, the time at the maximum measured concentration, the time of the last measurement, the elimination time, the half-life, and the area under the concentration-time curve (AUC). Three estimates of the area under the concentration-time curve from 0 to infinity ( $AUC_{0,\infty}$ ) are also calculated.

`pkexamine` is one of the `pk` commands. Please read [R] [pk](#) before reading this entry.



## Options

### Main

`fit(#)` specifies the number of points, counting back from the last measurement, to use in fitting the extension to estimate the  $AUC_{0,\infty}$ . The default is `fit(3)`, or the last three points. This value should be viewed as a minimum; the appropriate number of points will depend on your data.

`trapezoid` specifies that the trapezoidal rule be used to calculate the AUC. The default is cubic splines, which give better results for most functions. When the curve is irregular, `trapezoid` may give better results.

`graph` tells `pkexamine` to graph the concentration-time curve.

`line` and `log` specify the estimates of the  $AUC_{0,\infty}$  to display when graphing the  $AUC_{0,\infty}$ . These options are ignored, unless they are specified with the `graph` option.

`exp(#)` specifies that the exponential fit for the  $AUC_{0,\infty}$  be plotted. You must specify the maximum time value to which you want to plot the curve, and this time value must be greater than the maximum time measurement in the data. If you specify 0, the curve will be plotted to the point at which the linear extension would cross the  $x$  axis. This option is not valid with the `line` or `log` option and is ignored, unless the `graph` option is also specified.

### AUC plot

`cline_options` affect the rendition of the plotted points connected by lines; see [\[G\] \*cline\\_options\*](#).

`marker_options` specify the look of markers. This look includes the marker symbol, the marker size, and its color and outline; see [\[G\] \*marker\\_options\*](#).

`marker_label_options` specify if and how the markers are to be labeled; see [\[G\] \*marker\\_label\\_options\*](#).

### Add plots

`addplot(plot)` provides a way to add other plots to the generated graph. See [\[G\] \*addplot\\_option\*](#).

### Y axis, X axis, Titles, Legend, Overall

`twoway_options` are any of the options documented in [\[G\] \*twoway\\_options\*](#), excluding `by()`. These include options for titling the graph (see [\[G\] \*title\\_options\*](#)) and for saving the graph to disk (see [\[G\] \*saving\\_option\*](#)).

## Remarks

`pkexamine` computes summary statistics for a given patient in a pharmacokinetic trial. If by `idvar` is specified, statistics will be displayed for each subject in the data.

### ► Example 1

[Chow and Liu \(2009, 13\)](#) present data on a study examining primidone concentrations versus time for a subject over a 32-hour period after dosing.

```
. use http://www.stata-press.com/data/r11/auc
. list, abbrev(14)
```

|     | id | time | concentration |
|-----|----|------|---------------|
| 1.  | 1  | 0    | 0             |
| 2.  | 1  | .5   | 0             |
| 3.  | 1  | 1    | 2.8           |
| 4.  | 1  | 1.5  | 4.4           |
| 5.  | 1  | 2    | 4.4           |
| 6.  | 1  | 3    | 4.7           |
| 7.  | 1  | 4    | 4.1           |
| 8.  | 1  | 6    | 4             |
| 9.  | 1  | 8    | 3.6           |
| 10. | 1  | 12   | 3             |
| 11. | 1  | 16   | 2.5           |
| 12. | 1  | 24   | 2             |
| 13. | 1  | 32   | 1.6           |

We use pkexamine to produce the summary statistics:

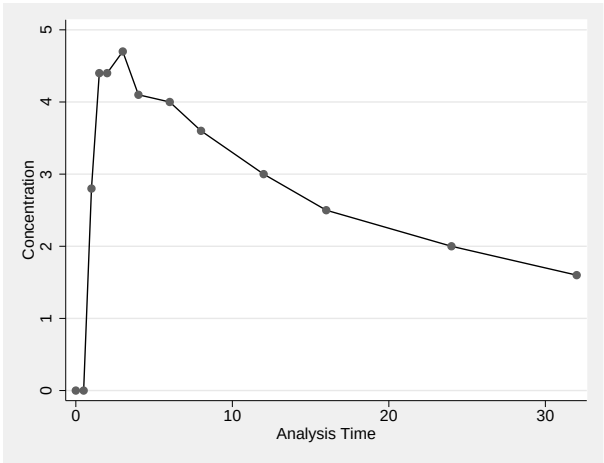
```
. pkexamine time conc, graph

Maximum concentration = 4.7
Time of maximum concentration = 3
Time of last observation (Tmax) = 32
Elimination rate = 0.0279
Half life = 24.8503
```

Area under the curve

| AUC [0, Tmax] | AUC [0, inf.)<br>Linear of log conc. | AUC [0, inf.)<br>Linear fit | AUC [0, inf.)<br>Exponential fit |
|---------------|--------------------------------------|-----------------------------|----------------------------------|
| 85.24         | 142.603                              | 107.759                     | 142.603                          |

Fit based on last 3 points.



The maximum concentration of 4.7 occurs at time 3, and the time of the last observation (Tmax) is 32. In addition to the AUC, which is calculated from 0 to the maximum value of time, pkexamine also reports the area under the curve, computed by extending the curve with each of three methods: a linear fit to the log of the concentration, a linear regression line, and a decreasing exponential regression line. See [Methods and formulas](#) for details on these three methods.

By default, all extensions to the AUC are based on the last three points. Looking at the graph for these data, it seems more appropriate to use the last seven points to estimate the  $AUC_{0,\infty}$ :

```
. pkexamine time conc, fit(7)
```

|                                   |         |
|-----------------------------------|---------|
| Maximum concentration =           | 4.7     |
| Time of maximum concentration =   | 3       |
| Time of last observation (Tmax) = | 32      |
| Elimination rate =                | 0.0349  |
| Half life =                       | 19.8354 |

Area under the curve

| AUC [0, Tmax] | AUC [0, inf.)<br>Linear of log conc. | AUC [0, inf.)<br>Linear fit | AUC [0, inf.)<br>Exponential fit |
|---------------|--------------------------------------|-----------------------------|----------------------------------|
| 85.24         | 131.027                              | 96.805                      | 129.181                          |

Fit based on last 7 points.

This approach decreased the estimate of the  $AUC_{0,\infty}$  for all extensions. To see a graph of the  $AUC_{0,\infty}$  using a linear extension, specify the `graph` and `line` options.

```
. pkexamine time conc, fit(7) graph line
```

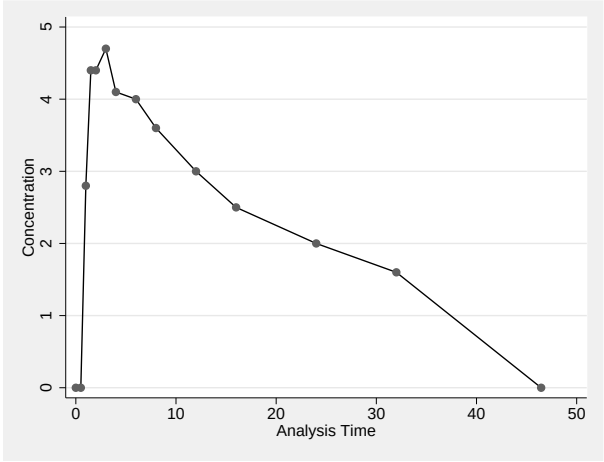
|                                   |         |
|-----------------------------------|---------|
| Maximum concentration =           | 4.7     |
| Time of maximum concentration =   | 3       |
| Time of last observation (Tmax) = | 32      |
| Elimination rate =                | 0.0349  |
| Half life =                       | 19.8354 |

Area under the curve

| AUC [0, Tmax] | AUC [0, inf.)<br>Linear of log conc. | AUC [0, inf.)<br>Linear fit | AUC [0, inf.)<br>Exponential fit |
|---------------|--------------------------------------|-----------------------------|----------------------------------|
| 85.24         | 131.027                              | 96.805                      | 129.181                          |

Fit based on last 7 points.

(Continued on next page)



## Saved results

pkexamine saves the following in `r()`:

|                          |                                                                 |
|--------------------------|-----------------------------------------------------------------|
| Scalars                  |                                                                 |
| <code>r(auc)</code>      | area under the concentration curve                              |
| <code>r(half)</code>     | half-life of the drug                                           |
| <code>r(ke)</code>       | elimination rate                                                |
| <code>r(tmax)</code>     | time at last concentration measurement                          |
| <code>r(cmax)</code>     | maximum concentration                                           |
| <code>r(tomc)</code>     | time of maximum concentration                                   |
| <code>r(auc_line)</code> | $AUC_{0,\infty}$ estimated with a linear fit                    |
| <code>r(auc_exp)</code>  | $AUC_{0,\infty}$ estimated with an exponential fit              |
| <code>r(auc_ln)</code>   | $AUC_{0,\infty}$ estimated with a linear fit of the natural log |

## Methods and formulas

pkexamine is implemented as an ado-file.

The  $AUC_{0,t_{\max}}$  is defined as

$$AUC_{0,t_{\max}} = \int_0^{t_{\max}} C_t dt$$

where  $C_t$  is the concentration at time  $t$ . By default, the integral is calculated numerically using cubic splines. However, if the trapezoidal rule is used, the  $AUC_{0,t_{\max}}$  is given as

$$AUC_{0,t_{\max}} = \sum_{i=2}^k \frac{C_{i-1} + C_i}{2} (t_i - t_{i-1})$$

The  $AUC_{0,\infty}$  is the  $AUC_{0,t_{\max}} + AUC_{t_{\max},\infty}$ , or

$$AUC_{0,\infty} = \int_0^{t_{\max}} C_t dt + \int_{t_{\max}}^{\infty} C_t dt$$

When using the linear extension to the  $AUC_{0,t_{\max}}$ , the integration is cut off when the line crosses the  $x$  axis. The log extension is a linear extension on the log concentration scale. The area for the exponential extension is

$$AUC_{0,\infty} = \int_{t_{\max}}^{\infty} e^{-(\beta_0 + t\beta_1)} dt = - \frac{e^{-(\beta_0 + t_{\max}\beta_1)}}{\beta_1}$$

Finally, the elimination rate,  $K_{eq}$ , is the negative of the parameter estimate for a linear regression of log time on concentration and is given in the standard manner:

$$K_{eq} = - \frac{\sum_{i=1}^k (C_i - \bar{C}) (\ln t_i - \bar{\ln t})}{\sum_{i=1}^k (C_i - \bar{C})^2}$$

and

$$t_{1/2} = \frac{\ln 2}{K_{eq}}$$

## Reference

Chow, S. C., and J. P. Liu. 2009. *Design and Analysis of Bioavailability and Bioequivalence Studies*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC.

## Also see

[R] **pk** — Pharmacokinetic (biopharmaceutical) data

Syntax

```
pkshape id sequence period1 period2 [period list] [, options]
```

| options                   | description                                               |
|---------------------------|-----------------------------------------------------------|
| <u>order</u> (string)     | apply treatments in specified order                       |
| <u>outcome</u> (newvar)   | name for outcome variable; default is outcome(outcome)    |
| <u>treatment</u> (newvar) | name for treatment variable; default is treatment(treat)  |
| <u>carryover</u> (newvar) | name for carryover variable; default is carryover(carry)  |
| <u>sequence</u> (newvar)  | name for sequence variable; default is sequence(sequence) |
| <u>period</u> (newvar)    | name for period variable; default is period(period)       |

Menu

Statistics > Epidemiology and related > Other > Reshape pharmacokinetic latin-square data

Description

pkshape reshapes the data for use with anova, pkcross, and pkequiv. Latin-square and crossover data are often organized in a manner that cannot be analyzed easily with Stata. pkshape reorganizes the data in memory for use in Stata.

pkshape is one of the pk commands. Please read [R] pk before reading this entry.

Options

- order(string) specifies the order in which treatments were applied. If the sequence() specifier is a string variable that specifies the order, this option is not necessary. Otherwise, order() specifies how to generate the treatment and carryover variables. Any string variable can be used to specify the order. For crossover designs, any washout periods can be indicated with the number 0.
- outcome(newvar) specifies the name for the outcome variable in the reorganized data. By default, outcome(outcome) is used.
- treatment(newvar) specifies the name for the treatment variable in the reorganized data. By default, treatment(treat) is used.
- carryover(newvar) specifies the name for the carryover variable in the reorganized data. By default, carryover(carry) is used.
- sequence(newvar) specifies the name for the sequence variable in the reorganized data. By default, sequence(sequence) is used.
- period(newvar) specifies the name for the period variable in the reorganized data. By default, period(period) is used.

## Remarks

Often data from a Latin-square experiment are naturally organized in a manner that Stata cannot manage easily. `pkshape` reorganizes Latin-square data so that they can be used with `anova` (see [R] `anova`) or any `pk` command. This includes the classic  $2 \times 2$  crossover design commonly used in pharmaceutical research, as well as many other Latin-square designs.

### ► Example 1

Consider the example data published in [Chow and Liu \(2009, 71\)](#). There are 24 patients, 12 in each sequence. Sequence 1 consists of the reference formulation followed by the test formulation; sequence 2 is the test formulation followed by the reference formulation. The measurements reported are the  $AUC_{0-t_{\max}}$  for each patient and for each period.

```
. use http://www.stata-press.com/data/r11/chowliu
. list, sep(4)
```

|     | id | seq | period1 | period2 |
|-----|----|-----|---------|---------|
| 1.  | 1  | 1   | 74.675  | 73.675  |
| 2.  | 4  | 1   | 96.4    | 93.25   |
| 3.  | 5  | 1   | 101.95  | 102.125 |
| 4.  | 6  | 1   | 79.05   | 69.45   |
| 5.  | 11 | 1   | 79.05   | 69.025  |
| 6.  | 12 | 1   | 85.95   | 68.7    |
| 7.  | 15 | 1   | 69.725  | 59.425  |
| 8.  | 16 | 1   | 86.275  | 76.125  |
| 9.  | 19 | 1   | 112.675 | 114.875 |
| 10. | 20 | 1   | 99.525  | 116.25  |
| 11. | 23 | 1   | 89.425  | 64.175  |
| 12. | 24 | 1   | 55.175  | 74.575  |
| 13. | 2  | 2   | 74.825  | 37.35   |
| 14. | 3  | 2   | 86.875  | 51.925  |
| 15. | 7  | 2   | 81.675  | 72.175  |
| 16. | 8  | 2   | 92.7    | 77.5    |
| 17. | 9  | 2   | 50.45   | 71.875  |
| 18. | 10 | 2   | 66.125  | 94.025  |
| 19. | 13 | 2   | 122.45  | 124.975 |
| 20. | 14 | 2   | 99.075  | 85.225  |
| 21. | 17 | 2   | 86.35   | 95.925  |
| 22. | 18 | 2   | 49.925  | 67.1    |
| 23. | 21 | 2   | 42.7    | 59.425  |
| 24. | 22 | 2   | 91.725  | 114.05  |

Because the outcome for one person is in two different variables, the treatment that was applied to an individual is a function of the period and the sequence. To analyze this treatment using `anova`, all the outcomes must be in one variable, and each covariate must be in its own variable. To reorganize these data, use `pkshape`:

```
. pkshape id seq period1 period2, order(ab ba)
. sort seq id treat
```

```
. list, sep(8)
```

|     | id | sequence | outcome | treat | carry | period |
|-----|----|----------|---------|-------|-------|--------|
| 1.  | 1  | 1        | 74.675  | 1     | 0     | 1      |
| 2.  | 1  | 1        | 73.675  | 2     | 1     | 2      |
| 3.  | 4  | 1        | 96.4    | 1     | 0     | 1      |
| 4.  | 4  | 1        | 93.25   | 2     | 1     | 2      |
| 5.  | 5  | 1        | 101.95  | 1     | 0     | 1      |
| 6.  | 5  | 1        | 102.125 | 2     | 1     | 2      |
| 7.  | 6  | 1        | 79.05   | 1     | 0     | 1      |
| 8.  | 6  | 1        | 69.45   | 2     | 1     | 2      |
| 9.  | 11 | 1        | 79.05   | 1     | 0     | 1      |
| 10. | 11 | 1        | 69.025  | 2     | 1     | 2      |
| 11. | 12 | 1        | 85.95   | 1     | 0     | 1      |
| 12. | 12 | 1        | 68.7    | 2     | 1     | 2      |
| 13. | 15 | 1        | 69.725  | 1     | 0     | 1      |
| 14. | 15 | 1        | 59.425  | 2     | 1     | 2      |
| 15. | 16 | 1        | 86.275  | 1     | 0     | 1      |
| 16. | 16 | 1        | 76.125  | 2     | 1     | 2      |
| 17. | 19 | 1        | 112.675 | 1     | 0     | 1      |
| 18. | 19 | 1        | 114.875 | 2     | 1     | 2      |
| 19. | 20 | 1        | 99.525  | 1     | 0     | 1      |
| 20. | 20 | 1        | 116.25  | 2     | 1     | 2      |
| 21. | 23 | 1        | 89.425  | 1     | 0     | 1      |
| 22. | 23 | 1        | 64.175  | 2     | 1     | 2      |
| 23. | 24 | 1        | 55.175  | 1     | 0     | 1      |
| 24. | 24 | 1        | 74.575  | 2     | 1     | 2      |
| 25. | 2  | 2        | 37.35   | 1     | 2     | 2      |
| 26. | 2  | 2        | 74.825  | 2     | 0     | 1      |
| 27. | 3  | 2        | 51.925  | 1     | 2     | 2      |
| 28. | 3  | 2        | 86.875  | 2     | 0     | 1      |
| 29. | 7  | 2        | 72.175  | 1     | 2     | 2      |
| 30. | 7  | 2        | 81.675  | 2     | 0     | 1      |
| 31. | 8  | 2        | 77.5    | 1     | 2     | 2      |
| 32. | 8  | 2        | 92.7    | 2     | 0     | 1      |
| 33. | 9  | 2        | 71.875  | 1     | 2     | 2      |
| 34. | 9  | 2        | 50.45   | 2     | 0     | 1      |
| 35. | 10 | 2        | 94.025  | 1     | 2     | 2      |
| 36. | 10 | 2        | 66.125  | 2     | 0     | 1      |
| 37. | 13 | 2        | 124.975 | 1     | 2     | 2      |
| 38. | 13 | 2        | 122.45  | 2     | 0     | 1      |
| 39. | 14 | 2        | 85.225  | 1     | 2     | 2      |
| 40. | 14 | 2        | 99.075  | 2     | 0     | 1      |
| 41. | 17 | 2        | 95.925  | 1     | 2     | 2      |
| 42. | 17 | 2        | 86.35   | 2     | 0     | 1      |
| 43. | 18 | 2        | 67.1    | 1     | 2     | 2      |
| 44. | 18 | 2        | 49.925  | 2     | 0     | 1      |
| 45. | 21 | 2        | 59.425  | 1     | 2     | 2      |
| 46. | 21 | 2        | 42.7    | 2     | 0     | 1      |
| 47. | 22 | 2        | 114.05  | 1     | 2     | 2      |
| 48. | 22 | 2        | 91.725  | 2     | 0     | 1      |

Now the data are organized into separate variables that indicate each factor level for each of the covariates, so the data may be used with `anova` or `pkcross`; see [\[R\] anova](#) and [\[R\] pkcross](#).



➤ Example 2

Consider the study of background music on bank teller productivity published in [Kutner et al. \(2005\)](#). The data are

| Week | Monday | Tuesday | Wednesday | Thursday | Friday |
|------|--------|---------|-----------|----------|--------|
| 1    | 18(D)  | 17(C)   | 14(A)     | 21(B)    | 17(E)  |
| 2    | 13(C)  | 34(B)   | 21(E)     | 16(A)    | 15(D)  |
| 3    | 7(A)   | 29(D)   | 32(B)     | 27(E)    | 13(C)  |
| 4    | 17(E)  | 13(A)   | 24(C)     | 31(D)    | 25(B)  |
| 5    | 21(B)  | 26(E)   | 26(D)     | 31(C)    | 7(A)   |

The numbers are the productivity scores, and the letters represent the treatment. We entered the data into Stata:

```
. use http://www.stata-press.com/data/r11/music, clear
. list
```

|    | id | seq   | day1 | day2 | day3 | day4 | day5 |
|----|----|-------|------|------|------|------|------|
| 1. | 1  | dcabe | 18   | 17   | 14   | 21   | 17   |
| 2. | 2  | cbead | 13   | 34   | 21   | 16   | 15   |
| 3. | 3  | adbec | 7    | 29   | 32   | 27   | 13   |
| 4. | 4  | eacdb | 17   | 13   | 24   | 31   | 25   |
| 5. | 5  | bedca | 21   | 26   | 26   | 31   | 7    |

We reshape these data with `pkshape`:

```
. pkshape id seq day1 day2 day3 day4 day5
. list, sep(0)
```

|     | id | sequence | outcome | treat | carry | period |
|-----|----|----------|---------|-------|-------|--------|
| 1.  | 3  | 1        | 7       | 1     | 0     | 1      |
| 2.  | 5  | 2        | 21      | 3     | 0     | 1      |
| 3.  | 2  | 3        | 13      | 5     | 0     | 1      |
| 4.  | 1  | 4        | 18      | 2     | 0     | 1      |
| 5.  | 4  | 5        | 17      | 4     | 0     | 1      |
| 6.  | 3  | 1        | 29      | 2     | 1     | 2      |
| 7.  | 5  | 2        | 26      | 4     | 3     | 2      |
| 8.  | 2  | 3        | 34      | 3     | 5     | 2      |
| 9.  | 1  | 4        | 17      | 5     | 2     | 2      |
| 10. | 4  | 5        | 13      | 1     | 4     | 2      |
| 11. | 3  | 1        | 32      | 3     | 2     | 3      |
| 12. | 5  | 2        | 26      | 2     | 4     | 3      |
| 13. | 2  | 3        | 21      | 4     | 3     | 3      |
| 14. | 1  | 4        | 14      | 1     | 5     | 3      |
| 15. | 4  | 5        | 24      | 5     | 1     | 3      |
| 16. | 3  | 1        | 27      | 4     | 3     | 4      |
| 17. | 5  | 2        | 31      | 5     | 2     | 4      |
| 18. | 2  | 3        | 16      | 1     | 4     | 4      |
| 19. | 1  | 4        | 21      | 3     | 1     | 4      |
| 20. | 4  | 5        | 31      | 2     | 5     | 4      |
| 21. | 3  | 1        | 13      | 5     | 4     | 5      |
| 22. | 5  | 2        | 7       | 1     | 5     | 5      |
| 23. | 2  | 3        | 15      | 2     | 1     | 5      |
| 24. | 1  | 4        | 17      | 4     | 3     | 5      |
| 25. | 4  | 5        | 25      | 3     | 2     | 5      |

Here the `sequence` variable is a string variable that specifies how the treatments were applied, so the `order` option is not used. When the `sequence` variable is a string and the `order` is specified, the arguments from the `order` option are used. We could now produce an ANOVA table:

```
. anova outcome seq period treat
```

|          |                 |         |                 |        |          |
|----------|-----------------|---------|-----------------|--------|----------|
|          | Number of obs = | 25      | R-squared =     | 0.8666 |          |
|          | Root MSE =      | 3.96232 | Adj R-squared = | 0.7331 |          |
| Source   | Partial SS      | df      | MS              | F      | Prob > F |
| Model    | 1223.6          | 12      | 101.966667      | 6.49   | 0.0014   |
| sequence | 82              | 4       | 20.5            | 1.31   | 0.3226   |
| period   | 477.2           | 4       | 119.3           | 7.60   | 0.0027   |
| treat    | 664.4           | 4       | 166.1           | 10.58  | 0.0007   |
| Residual | 188.4           | 12      | 15.7            |        |          |
| Total    | 1412            | 24      | 58.8333333      |        |          |

➤ Example 3

Consider the Latin-square crossover example published in [Kutner et al. \(2005\)](#). The example is about apple sales given different methods for displaying apples.

| Pattern | Store | Week 1 | Week 2 | Week 3 |
|---------|-------|--------|--------|--------|
| 1       | 1     | 9(B)   | 12(C)  | 15(A)  |
|         | 2     | 4(B)   | 12(C)  | 9(A)   |
| 2       | 1     | 12(A)  | 14(B)  | 3(C)   |
|         | 2     | 13(A)  | 14(B)  | 3(C)   |
| 3       | 1     | 7(C)   | 18(A)  | 6(B)   |
|         | 2     | 5(C)   | 20(A)  | 4(B)   |

We entered the data into Stata:

```
. use http://www.stata-press.com/data/r11/applesales, clear
. list, sep(2)
```

|    | id | seq | p1 | p2 | p3 | square |
|----|----|-----|----|----|----|--------|
| 1. | 1  | 1   | 9  | 12 | 15 | 1      |
| 2. | 2  | 1   | 4  | 12 | 9  | 2      |
| 3. | 3  | 2   | 12 | 14 | 3  | 1      |
| 4. | 4  | 2   | 13 | 14 | 3  | 2      |
| 5. | 5  | 3   | 7  | 18 | 6  | 1      |
| 6. | 6  | 3   | 5  | 20 | 4  | 2      |

Now the data can be reorganized using descriptive names for the outcome variables.

```
. pkshape id seq p1 p2 p3, order(bca abc cab) seq(pattern) period(order)
> treat(displays)
```

```
. anova outcome pattern order display id|pattern
```

|            | Number of obs = 18 |    | R-squared = 0.9562     |       |          |
|------------|--------------------|----|------------------------|-------|----------|
|            | Root MSE = 1.59426 |    | Adj R-squared = 0.9069 |       |          |
| Source     | Partial SS         | df | MS                     | F     | Prob > F |
| Model      | 443.666667         | 9  | 49.2962963             | 19.40 | 0.0002   |
| pattern    | .333333333         | 2  | .166666667             | 0.07  | 0.9370   |
| order      | 233.333333         | 2  | 116.666667             | 45.90 | 0.0000   |
| displays   | 189                | 2  | 94.5                   | 37.18 | 0.0001   |
| id pattern | 21                 | 3  | 7                      | 2.75  | 0.1120   |
| Residual   | 20.3333333         | 8  | 2.54166667             |       |          |
| Total      | 464                | 17 | 27.2941176             |       |          |

These are the same results reported by [Kutner et al. \(2005\)](#).

◀

#### ► Example 4

We continue with [example 1](#) from [R] [pkcollapse](#); the data are

```
. use http://www.stata-press.com/data/r11/pkdata2, clear
. list, sep(4) abbrev(10)
```

|     | id | seq | auc_concA | auc_concB |
|-----|----|-----|-----------|-----------|
| 1.  | 1  | 1   | 150.9643  | 218.5551  |
| 2.  | 2  | 1   | 146.7606  | 133.3201  |
| 3.  | 3  | 1   | 160.6548  | 126.0635  |
| 4.  | 4  | 1   | 157.8622  | 96.17461  |
| 5.  | 5  | 1   | 133.6957  | 188.9038  |
| 6.  | 7  | 1   | 160.639   | 223.6922  |
| 7.  | 8  | 1   | 131.2604  | 104.0139  |
| 8.  | 9  | 1   | 168.5186  | 237.8962  |
| 9.  | 10 | 2   | 137.0627  | 139.7382  |
| 10. | 12 | 2   | 153.4038  | 202.3942  |
| 11. | 13 | 2   | 163.4593  | 136.7848  |
| 12. | 14 | 2   | 146.0462  | 104.5191  |
| 13. | 15 | 2   | 158.1457  | 165.8654  |
| 14. | 18 | 2   | 147.1977  | 139.235   |
| 15. | 19 | 2   | 164.9988  | 166.2391  |
| 16. | 20 | 2   | 145.3823  | 158.5146  |

```
. pkshape id seq auc_concA auc_concB, order(ab ba)
. sort period id
```

```
. list, sep(4)
```

|     | id | sequence | outcome  | treat | carry | period |
|-----|----|----------|----------|-------|-------|--------|
| 1.  | 1  | 1        | 150.9643 | 1     | 0     | 1      |
| 2.  | 2  | 1        | 146.7606 | 1     | 0     | 1      |
| 3.  | 3  | 1        | 160.6548 | 1     | 0     | 1      |
| 4.  | 4  | 1        | 157.8622 | 1     | 0     | 1      |
| 5.  | 5  | 1        | 133.6957 | 1     | 0     | 1      |
| 6.  | 7  | 1        | 160.639  | 1     | 0     | 1      |
| 7.  | 8  | 1        | 131.2604 | 1     | 0     | 1      |
| 8.  | 9  | 1        | 168.5186 | 1     | 0     | 1      |
| 9.  | 10 | 2        | 137.0627 | 2     | 0     | 1      |
| 10. | 12 | 2        | 153.4038 | 2     | 0     | 1      |
| 11. | 13 | 2        | 163.4593 | 2     | 0     | 1      |
| 12. | 14 | 2        | 146.0462 | 2     | 0     | 1      |
| 13. | 15 | 2        | 158.1457 | 2     | 0     | 1      |
| 14. | 18 | 2        | 147.1977 | 2     | 0     | 1      |
| 15. | 19 | 2        | 164.9988 | 2     | 0     | 1      |
| 16. | 20 | 2        | 145.3823 | 2     | 0     | 1      |
| 17. | 1  | 1        | 218.5551 | 2     | 1     | 2      |
| 18. | 2  | 1        | 133.3201 | 2     | 1     | 2      |
| 19. | 3  | 1        | 126.0635 | 2     | 1     | 2      |
| 20. | 4  | 1        | 96.17461 | 2     | 1     | 2      |
| 21. | 5  | 1        | 188.9038 | 2     | 1     | 2      |
| 22. | 7  | 1        | 223.6922 | 2     | 1     | 2      |
| 23. | 8  | 1        | 104.0139 | 2     | 1     | 2      |
| 24. | 9  | 1        | 237.8962 | 2     | 1     | 2      |
| 25. | 10 | 2        | 139.7382 | 1     | 2     | 2      |
| 26. | 12 | 2        | 202.3942 | 1     | 2     | 2      |
| 27. | 13 | 2        | 136.7848 | 1     | 2     | 2      |
| 28. | 14 | 2        | 104.5191 | 1     | 2     | 2      |
| 29. | 15 | 2        | 165.8654 | 1     | 2     | 2      |
| 30. | 18 | 2        | 139.235  | 1     | 2     | 2      |
| 31. | 19 | 2        | 166.2391 | 1     | 2     | 2      |
| 32. | 20 | 2        | 158.5146 | 1     | 2     | 2      |



We call the resulting dataset `pkdata3`. We conduct equivalence testing on the data in [\[R\] pkequiv](#), and we fit an ANOVA model to these data in the [third example](#) of [\[R\] pkcross](#).

## Methods and formulas

`pkshape` is implemented as an ado-file.

## References

Chow, S. C., and J. P. Liu. 2009. *Design and Analysis of Bioavailability and Bioequivalence Studies*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC.

Kutner, M. H., C. J. Nachtsheim, J. Neter, and W. Li. 2005. *Applied Linear Statistical Models*. 5th ed. New York: McGraw-Hill/Irwin.

## Also see

[R] [pk](#) — Pharmacokinetic (biopharmaceutical) data

Syntax

```
pksumm id time concentration [if] [in] [, options]
```

| options                                                           | description                                                                         |
|-------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| Main                                                              |                                                                                     |
| <code>trapezoid</code>                                            | use trapezoidal rule to calculate AUC; default is cubic splines                     |
| <code>fit(#)</code>                                               | use # points to estimate AUC; default is <code>fit(3)</code>                        |
| <code>notimechk</code>                                            | do not check whether follow-up time for all subjects is the same                    |
| <code>nodots</code>                                               | suppress the dots during calculation                                                |
| <code>graph</code>                                                | graph the distribution of <i>statistic</i>                                          |
| <code>stat(statistic)</code>                                      | graph the specified statistic; default is <code>stat(auc)</code>                    |
| Histogram, Density plots, Y axis, X axis, Titles, Legend, Overall |                                                                                     |
| <code>histogram_options</code>                                    | any option other than <code>by()</code> documented in <a href="#">[R] histogram</a> |

| statistic            | description                                                                               |
|----------------------|-------------------------------------------------------------------------------------------|
| <code>auc</code>     | area under the concentration-time curve ( $AUC_{0,\infty}$ ); the default                 |
| <code>aucline</code> | area under the concentration-time curve from 0 to $\infty$ using a linear extension       |
| <code>aucexp</code>  | area under the concentration-time curve from 0 to $\infty$ using an exponential extension |
| <code>auclog</code>  | area under the log-concentration-time curve extended with a linear fit                    |
| <code>half</code>    | half-life of the drug                                                                     |
| <code>ke</code>      | elimination rate                                                                          |
| <code>cmax</code>    | maximum concentration                                                                     |
| <code>tmax</code>    | time at last concentration                                                                |
| <code>tomc</code>    | time of maximum concentration                                                             |

Menu

Statistics > Epidemiology and related > Other > Summarize pharmacokinetic data

Description

`pksumm` obtains summary measures based on the first four moments from the empirical distribution of each pharmacokinetic measurement and tests the null hypothesis that the distribution of that measurement is normally distributed.

`pksumm` is one of the `pk` commands. Please read [\[R\] pk](#) before reading this entry.

## Options

### Main

`trapezoid` specifies that the trapezoidal rule be used to calculate the AUC. The default is cubic splines, which give better results for most situations. When the curve is irregular, the trapezoidal rule may give better results.

`fit(#)` specifies the number of points, counting back from the last time measurement, to use in fitting the extension to estimate the  $AUC_{0,\infty}$ . The default is `fit(3)`, the last three points. This default should be viewed as a minimum; the appropriate number of points will depend on the data.

`notimechk` suppresses the check that the follow-up time for all subjects is the same. By default, `pksumm` expects the maximum follow-up time to be equal for all subjects.

`nodots` suppresses the progress dots during calculation. By default, a period is displayed for every call to calculate the pharmacokinetic measures.

`graph` requests a graph of the distribution of the statistic specified with `stat()`.

`stat(statistic)` specifies the statistic that `pksumm` should graph. The default is `stat(auc)`. If the `graph` option is not specified, this option is ignored.

### Histogram, Density plots, Y axis, X axis, Titles, Legend, Overall

`histogram_options` are any of the options documented in [\[R\] histogram](#), excluding `by()`. For `pksumm`, `fraction` is the default, not `density`.

## Remarks

`pksumm` produces summary statistics for the distribution of nine common pharmacokinetic measurements. If there are more than eight subjects, `pksumm` also computes a test for normality on each measurement. The nine measurements summarized by `pksumm` are listed above and are described in *Methods and formulas* of [\[R\] pkexamine](#) and [\[R\] pk](#).

### ► Example 1

We demonstrate the use of `pksumm` on a variation of the data described in [\[R\] pk](#). We have drug concentration data on 15 subjects, each measured at 13 time points over a 32-hour period. A few of the records are

(Continued on next page)

```
. use http://www.stata-press.com/data/r11/pksumm
. list, sep(0)
```

|                  | id | time | conc     |
|------------------|----|------|----------|
| 1.               | 1  | 0    | 0        |
| 2.               | 1  | .5   | 3.073403 |
| 3.               | 1  | 1    | 5.188444 |
| 4.               | 1  | 1.5  | 5.898577 |
| 5.               | 1  | 2    | 5.096378 |
| 6.               | 1  | 3    | 6.094085 |
| (output omitted) |    |      |          |
| 183.             | 15 | 0    | 0        |
| 184.             | 15 | .5   | 3.86493  |
| 185.             | 15 | 1    | 6.432444 |
| 186.             | 15 | 1.5  | 6.969195 |
| 187.             | 15 | 2    | 6.307024 |
| 188.             | 15 | 3    | 6.509584 |
| 189.             | 15 | 4    | 6.555091 |
| 190.             | 15 | 6    | 7.318319 |
| 191.             | 15 | 8    | 5.329813 |
| 192.             | 15 | 12   | 5.411624 |
| 193.             | 15 | 16   | 3.891397 |
| 194.             | 15 | 24   | 5.167516 |
| 195.             | 15 | 32   | 2.649686 |

We can use pksumm to view the summary statistics for all the pharmacokinetic parameters.

```
. pksumm id time conc
.....
Summary statistics for the pharmacokinetic measures
```

|         | Number of observations = 15 |        |           |          |          |         |
|---------|-----------------------------|--------|-----------|----------|----------|---------|
| Measure | Mean                        | Median | Variance  | Skewness | Kurtosis | p-value |
| auc     | 150.74                      | 150.96 | 123.07    | -0.26    | 2.10     | 0.69    |
| aucline | 408.30                      | 214.17 | 188856.87 | 2.57     | 8.93     | 0.00    |
| aucexp  | 691.68                      | 297.08 | 762679.94 | 2.56     | 8.87     | 0.00    |
| auclog  | 688.98                      | 297.67 | 797237.24 | 2.59     | 9.02     | 0.00    |
| half    | 94.84                       | 29.39  | 18722.13  | 2.26     | 7.37     | 0.00    |
| ke      | 0.02                        | 0.02   | 0.00      | 0.89     | 3.70     | 0.09    |
| cmax    | 7.36                        | 7.42   | 0.42      | -0.60    | 2.56     | 0.44    |
| tomc    | 3.47                        | 3.00   | 7.62      | 2.17     | 7.18     | 0.00    |
| tmax    | 32.00                       | 32.00  | 0.00      | .        | .        | .       |

For the 15 subjects, the mean  $AUC_{0,t_{\max}}$  is 150.74, and  $\sigma^2 = 123.07$ . The skewness of  $-0.26$  indicates that the distribution is slightly skewed left. The  $p$ -value of 0.69 for the  $\chi^2$  test of normality indicates that we cannot reject the null hypothesis that the distribution is normal.

If we were to consider any of the three variants of the  $AUC_{0,\infty}$ , we would see that there is huge variability and that the distribution is heavily skewed. A skewness different from 0 and a kurtosis different from 3 are expected because the distribution of the  $AUC_{0,\infty}$  is not normal.

We now graph the distribution of  $AUC_{0,t_{\max}}$  by specifying the `graph` option.

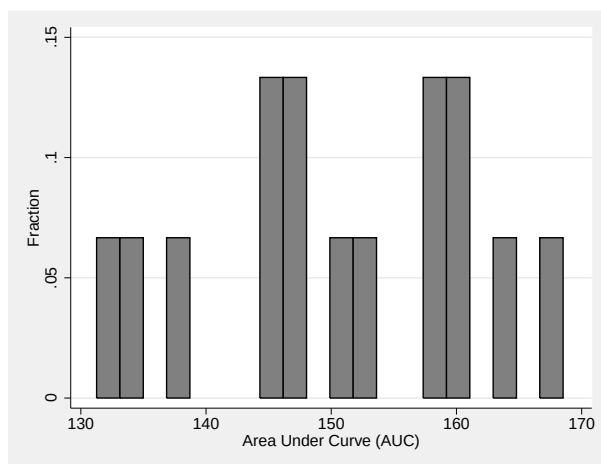


```
. pksumm id time conc, graph bin(20)
```

```
.....
```

Summary statistics for the pharmacokinetic measures

| Measure | Number of observations = 15 |        |           |          |          |         |
|---------|-----------------------------|--------|-----------|----------|----------|---------|
|         | Mean                        | Median | Variance  | Skewness | Kurtosis | p-value |
| auc     | 150.74                      | 150.96 | 123.07    | -0.26    | 2.10     | 0.69    |
| aucline | 408.30                      | 214.17 | 188856.87 | 2.57     | 8.93     | 0.00    |
| aucexp  | 691.68                      | 297.08 | 762679.94 | 2.56     | 8.87     | 0.00    |
| auclog  | 688.98                      | 297.67 | 797237.24 | 2.59     | 9.02     | 0.00    |
| half    | 94.84                       | 29.39  | 18722.13  | 2.26     | 7.37     | 0.00    |
| ke      | 0.02                        | 0.02   | 0.00      | 0.89     | 3.70     | 0.09    |
| cmax    | 7.36                        | 7.42   | 0.42      | -0.60    | 2.56     | 0.44    |
| tomc    | 3.47                        | 3.00   | 7.62      | 2.17     | 7.18     | 0.00    |
| tmax    | 32.00                       | 32.00  | 0.00      | .        | .        | .       |



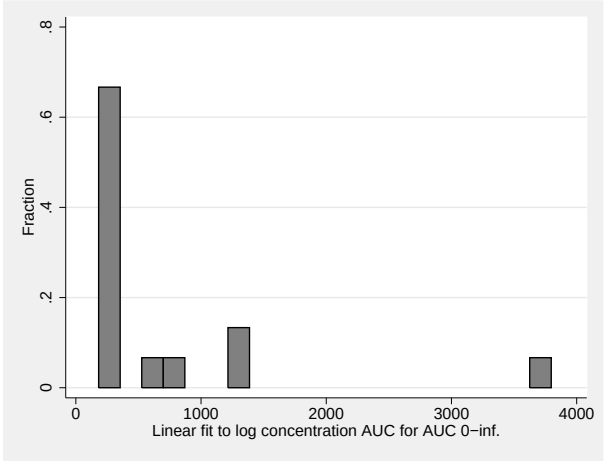
`graph`, by default, plots  $AUC_{0,t_{\max}}$ . To plot a graph of one of the other pharmacokinetic measurements, we need to specify the `stat()` option. For example, we can ask Stata to produce a plot of the  $AUC_{0,\infty}$  using the `log` extension:

```
. pksumm id time conc, stat(auclog) graph bin(20)
```

```
.....
```

Summary statistics for the pharmacokinetic measures

| Measure | Number of observations = 15 |        |           |          |          |         |
|---------|-----------------------------|--------|-----------|----------|----------|---------|
|         | Mean                        | Median | Variance  | Skewness | Kurtosis | p-value |
| auc     | 150.74                      | 150.96 | 123.07    | -0.26    | 2.10     | 0.69    |
| aucline | 408.30                      | 214.17 | 188856.87 | 2.57     | 8.93     | 0.00    |
| aucexp  | 691.68                      | 297.08 | 762679.94 | 2.56     | 8.87     | 0.00    |
| auclog  | 688.98                      | 297.67 | 797237.24 | 2.59     | 9.02     | 0.00    |
| half    | 94.84                       | 29.39  | 18722.13  | 2.26     | 7.37     | 0.00    |
| ke      | 0.02                        | 0.02   | 0.00      | 0.89     | 3.70     | 0.09    |
| cmax    | 7.36                        | 7.42   | 0.42      | -0.60    | 2.56     | 0.44    |
| tomc    | 3.47                        | 3.00   | 7.62      | 2.17     | 7.18     | 0.00    |
| tmax    | 32.00                       | 32.00  | 0.00      | .        | .        | .       |



## Methods and formulas

`pksumm` is implemented as an ado-file.

The  $\chi^2$  test for normality is conducted with `sktest`; see [\[R\] sktest](#) for more information on the test of normality.

The statistics reported by `pksumm` are identical to those reported by `summarize` and `sktest`; see [\[R\] summarize](#) and [\[R\] sktest](#).

## Also see

[\[R\] pk](#) — Pharmacokinetic (biopharmaceutical) data

Syntax

```
poisson depvar [indepvars] [if] [in] [weight] [, options]
```

| <i>options</i>                                    | description                                                                                                                                                                 |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                             |                                                                                                                                                                             |
| <code>noconstant</code>                           | suppress constant term                                                                                                                                                      |
| <code>exposure(<i>varname<sub>e</sub></i>)</code> | include $\ln(\textit{varname}_e)$ in model with coefficient constrained to 1                                                                                                |
| <code>offset(<i>varname<sub>o</sub></i>)</code>   | include $\textit{varname}_o$ in model with coefficient constrained to 1                                                                                                     |
| <code>constraints(<i>constraints</i>)</code>      | apply specified linear constraints                                                                                                                                          |
| <code>collinear</code>                            | keep collinear variables                                                                                                                                                    |
| SE/Robust                                         |                                                                                                                                                                             |
| <code>vce(<i>vcetype</i>)</code>                  | <i>vcetype</i> may be <code>oim</code> , <code>robust</code> , <code>cluster <i>clustvar</i></code> , <code>opg</code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                                         |                                                                                                                                                                             |
| <code>level(#)</code>                             | set confidence level; default is <code>level(95)</code>                                                                                                                     |
| <code>irr</code>                                  | report incidence-rate ratios                                                                                                                                                |
| <code>nocnsreport</code>                          | do not display constraints                                                                                                                                                  |
| <code>display_options</code>                      | control spacing and display of omitted variables and base and empty cells                                                                                                   |
| Maximization                                      |                                                                                                                                                                             |
| <code>maximize_options</code>                     | control the maximization process; seldom used                                                                                                                               |
| <sup>†</sup> <code>coeflegend</code>              | display coefficients' legend instead of coefficient table                                                                                                                   |

<sup>†</sup>`coeflegend` does not appear in the dialog box.

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar*, *indepvars*, *varname<sub>e</sub>*, and *varname<sub>o</sub>* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

`bootstrap`, `by`, `fracpoly`, `jackknife`, `mfp`, `mi estimate`, `nestreg`, `rolling`, `statsby`, `stepwise`, and `svy` are allowed; see [U] 11.1.10 Prefix commands.

`vce(bootstrap)` and `vce(jackknife)` are not allowed with the `mi estimate` prefix.

Weights are not allowed with the `bootstrap` prefix.

`vce()` and `weights` are not allowed with the `svy` prefix.

`pweights`, `iweights`, and `pweights` are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Count outcomes > Poisson regression

## Description

`poisson` fits a Poisson regression of *depvar* on *indepvars*, where *depvar* is a nonnegative count variable.

If you have panel data, see [XT] [xtpoisson](#).

## Options

### Model

`noconstant`, `exposure(varnamee)`, `offset(varnameo)`, `constraints(constraints)`, `collinear`; see [R] [estimation options](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

### Reporting

`level(#)`; see [R] [estimation options](#).

`irr` reports estimated coefficients transformed to incidence-rate ratios, that is,  $e^{\beta_i}$  rather than  $\beta_i$ .

Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated or stored. `irr` may be specified at estimation or when replaying previously estimated results.

`nocnsreport`; see [R] [estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [R] [maximize](#). These options are seldom used.

Setting the optimization type to `technique(bhhh)` resets the default `vcetype` to `vce(opg)`.

The following option is available with `poisson` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

The basic idea of Poisson regression was outlined by [Coleman \(1964, 378–379\)](#). See [Cameron and Trivedi \(1998; 2009, chap. 17\)](#) and [Johnson, Kemp, and Kotz \(2005, chap. 4\)](#) for information about the Poisson distribution. See [Cameron and Trivedi \(1998\)](#), [Long \(1997, chap. 8\)](#), [Long and Freese \(2006, chap. 8\)](#), [McNeil \(1996, chap. 6\)](#), and [Selvin \(2004, chap. 9\)](#) for an introduction to Poisson regression. Also see [Selvin \(2004, chap. 5\)](#) for a discussion of the analysis of spatial distributions, which includes a discussion of the Poisson distribution. An early example of Poisson regression was [Cochran \(1940\)](#).

Poisson regression fits models of the number of occurrences (counts) of an event. The Poisson distribution has been applied to diverse events, such as the number of soldiers kicked to death by horses in the Prussian army ([von Bortkewitsch 1898](#)); the pattern of hits by buzz bombs launched against London during World War II ([Clarke 1946](#)); telephone connections to a wrong number ([Thorndike 1926](#)); and disease incidence, typically with respect to time, but occasionally with respect to space. The basic assumptions are as follows:

1. There is a quantity called the *incidence rate* that is the rate at which events occur. Examples are 5 per second, 20 per 1,000 person-years, 17 per square meter, and 38 per cubic centimeter.
2. The incidence rate can be multiplied by *exposure* to obtain the expected number of observed events. For example, a rate of 5 per second multiplied by 30 seconds means that 150 events are expected; a rate of 20 per 1,000 person-years multiplied by 2,000 person-years means that 40 events are expected; and so on.
3. Over very small exposures  $\epsilon$ , the probability of finding more than one event is small compared with  $\epsilon$ .
4. Nonoverlapping exposures are mutually independent.

With these assumptions, to find the probability of  $k$  events in an exposure of size  $E$ , you divide  $E$  into  $n$  subintervals  $E_1, E_2, \dots, E_n$ , and approximate the answer as the binomial probability of observing  $k$  successes in  $n$  trials. If you let  $n \rightarrow \infty$ , you obtain the Poisson distribution.

In the Poisson regression model, the incidence rate for the  $j$ th observation is assumed to be given by

$$r_j = e^{\beta_0 + \beta_1 x_{1,j} + \dots + \beta_k x_{k,j}}$$

If  $E_j$  is the exposure, the expected number of events,  $C_j$ , will be

$$\begin{aligned} C_j &= E_j e^{\beta_0 + \beta_1 x_{1,j} + \dots + \beta_k x_{k,j}} \\ &= e^{\ln(E_j) + \beta_0 + \beta_1 x_{1,j} + \dots + \beta_k x_{k,j}} \end{aligned}$$

This model is fit by `poisson`. Without the `exposure()` or `offset()` options,  $E_j$  is assumed to be 1 (equivalent to assuming that exposure is unknown), and controlling for exposure, if necessary, is your responsibility.

Comparing rates is most easily done by calculating *incidence-rate ratios* (IRRs). For instance, what is the relative incidence rate of chromosome interchanges in cells as the intensity of radiation increases; the relative incidence rate of telephone connections to a wrong number as load increases; or the relative incidence rate of deaths due to cancer for females relative to males? That is, you want to hold all the  $x$ 's in the model constant except one, say, the  $i$ th. The IRR for a one-unit change in  $x_i$  is

$$\frac{e^{\ln(E) + \beta_1 x_1 + \dots + \beta_i (x_i + 1) + \dots + \beta_k x_k}}{e^{\ln(E) + \beta_1 x_1 + \dots + \beta_i x_i + \dots + \beta_k x_k}} = e^{\beta_i}$$

More generally, the IRR for a  $\Delta x_i$  change in  $x_i$  is  $e^{\beta_i \Delta x_i}$ . The `lincom` command can be used after `poisson` to display incidence-rate ratios for any group relative to another; see [\[R\] lincom](#).

## ► Example 1

[Chatterjee and Hadi \(2006, 162\)](#) give the number of injury incidents and the proportion of flights for each airline out of the total number of flights from New York for nine major U.S. airlines in one year:

```
. use http://www.stata-press.com/data/r11/airline
. list
```

|    | airline | injuries | n      | XYZowned |
|----|---------|----------|--------|----------|
| 1. | 1       | 11       | 0.0950 | 1        |
| 2. | 2       | 7        | 0.1920 | 0        |
| 3. | 3       | 7        | 0.0750 | 0        |
| 4. | 4       | 19       | 0.2078 | 0        |
| 5. | 5       | 9        | 0.1382 | 0        |
| 6. | 6       | 4        | 0.0540 | 1        |
| 7. | 7       | 3        | 0.1292 | 0        |
| 8. | 8       | 1        | 0.0503 | 0        |
| 9. | 9       | 3        | 0.0629 | 1        |

To their data, we have added a fictional variable, XYZowned. We will imagine that an accusation is made that the airlines owned by XYZ Company have a higher injury rate.

```
. poisson injuries XYZowned, exposure(n) irr
Iteration 0: log likelihood = -23.027197
Iteration 1: log likelihood = -23.027177
Iteration 2: log likelihood = -23.027177
```

|                    |               |   |        |
|--------------------|---------------|---|--------|
| Poisson regression | Number of obs | = | 9      |
|                    | LR chi2(1)    | = | 1.77   |
|                    | Prob > chi2   | = | 0.1836 |
|                    | Pseudo R2     | = | 0.0370 |

Log likelihood = -23.027177

| injuries      | IRR                    | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|---------------|------------------------|-----------|------|-------|----------------------|----------|
| XYZowned<br>n | 1.463467<br>(exposure) | .406872   | 1.37 | 0.171 | .8486578             | 2.523675 |

We specified irr to see the IRRs rather than the underlying coefficients. We estimate that XYZ Airlines’ injury rate is 1.46 times larger than that for other airlines, but the 95% confidence interval is 0.85 to 2.52; we cannot even reject the hypothesis that XYZ Airlines has a lower injury rate.

❑ Technical note

In example 1, we assumed that each airline’s exposure was proportional to its fraction of flights out of New York. What if “large” airlines, however, also used larger planes, and so had even more passengers than would be expected, given this measure of exposure? A better measure would be each airline’s fraction of passengers on flights out of New York, a number that we do not have. Even so, we suppose that n represents this number to some extent, so a better estimate of the effect might be

```
. gen lnN=ln(n)
. poisson injuries XYZowned lnN
Iteration 0: log likelihood = -22.333875
Iteration 1: log likelihood = -22.332276
Iteration 2: log likelihood = -22.332276
Poisson regression
```

|               |   |        |
|---------------|---|--------|
| Number of obs | = | 9      |
| LR chi2(2)    | = | 19.15  |
| Prob > chi2   | = | 0.0001 |
| Pseudo R2     | = | 0.3001 |

Log likelihood = -22.332276

| injuries | Coef.    | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|----------|----------|-----------|------|-------|----------------------|----------|
| XYZowned | .6840667 | .3895877  | 1.76 | 0.079 | -.0795111            | 1.447645 |
| lnN      | 1.424169 | .3725155  | 3.82 | 0.000 | .6940517             | 2.154285 |
| _cons    | 4.863891 | .7090501  | 6.86 | 0.000 | 3.474178             | 6.253603 |

Here rather than specifying the `exposure()` option, we explicitly included the variable that would normalize for exposure in the model. We did not specify the `irr` option, so we see coefficients rather than IRRs. We started with the model

$$\text{rate} = e^{\beta_0 + \beta_1 \text{XYZowned}}$$

The observed counts are therefore

$$\text{count} = n e^{\beta_0 + \beta_1 \text{XYZowned}} = e^{\ln(n) + \beta_0 + \beta_1 \text{XYZowned}}$$

which amounts to constraining the coefficient on `ln(n)` to 1. This is what was estimated when we specified the `exposure(n)` option. In the above model, we included the normalizing exposure ourselves and, rather than constraining the coefficient to be 1, estimated the coefficient.

The estimated coefficient is 1.42, a respectable distance away from 1, and is consistent with our speculation that larger airlines also use larger airplanes. With this small amount of data, however, we also have a wide confidence interval that includes 1.

Our estimated *coefficient* on `XYZowned` is now 0.684, and the implied IRR is  $e^{0.684} \approx 1.98$  (which we could also see by typing `poisson, irr`). The 95% confidence interval for the coefficient still includes 0 (the interval for the IRR includes 1), so although the point estimate is now larger, we still cannot be certain of our results.

Our expert opinion would be that, although there is not enough evidence to support the charge, there is enough evidence to justify collecting more data.

□

## ► Example 2

In a famous age-specific study of coronary disease deaths among male British doctors, [Doll and Hill \(1966\)](#) reported the following data (reprinted in [Rothman, Greenland, and Lash \[2008, 264\]](#)):

| Age   | Smokers |              | Nonsmokers |              |
|-------|---------|--------------|------------|--------------|
|       | Deaths  | Person-years | Deaths     | Person-years |
| 35–44 | 32      | 52,407       | 2          | 18,790       |
| 45–54 | 104     | 43,248       | 12         | 10,673       |
| 55–64 | 206     | 28,612       | 28         | 5,710        |
| 65–74 | 186     | 12,663       | 28         | 2,585        |
| 75–84 | 102     | 5,317        | 31         | 1,462        |

The first step is to enter these data into Stata, which we have done:

```
. use http://www.stata-press.com/data/r11/dollhill13, clear
. list
```

|     | agecat | smokes | deaths | pyears |
|-----|--------|--------|--------|--------|
| 1.  | 1      | 1      | 32     | 52,407 |
| 2.  | 2      | 1      | 104    | 43,248 |
| 3.  | 3      | 1      | 206    | 28,612 |
| 4.  | 4      | 1      | 186    | 12,663 |
| 5.  | 5      | 1      | 102    | 5,317  |
| 6.  | 1      | 0      | 2      | 18,790 |
| 7.  | 2      | 0      | 12     | 10,673 |
| 8.  | 3      | 0      | 28     | 5,710  |
| 9.  | 4      | 0      | 28     | 2,585  |
| 10. | 5      | 0      | 31     | 1,462  |

agecat 1 corresponds to 35–44, agecat 2 to 45–54, and so on. The most “natural” analysis of these data would begin by introducing indicator variables for each age category and one indicator for smoking:

```
. poisson deaths smokes i.agecat, exposure(pyears) irr
Iteration 0: log likelihood = -33.823284
Iteration 1: log likelihood = -33.600471
Iteration 2: log likelihood = -33.600153
Iteration 3: log likelihood = -33.600153
```

|                             |               |   |        |
|-----------------------------|---------------|---|--------|
| Poisson regression          | Number of obs | = | 10     |
|                             | LR chi2(5)    | = | 922.93 |
|                             | Prob > chi2   | = | 0.0000 |
| Log likelihood = -33.600153 | Pseudo R2     | = | 0.9321 |

| deaths | IRR        | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|--------|------------|-----------|-------|-------|----------------------|----------|
| smokes | 1.425519   | .1530638  | 3.30  | 0.001 | 1.154984             | 1.759421 |
| agecat | (base)     |           |       |       |                      |          |
| 1      |            |           |       |       |                      |          |
| 2      | 4.410584   | .8605197  | 7.61  | 0.000 | 3.009011             | 6.464997 |
| 3      | 13.8392    | 2.542638  | 14.30 | 0.000 | 9.654328             | 19.83809 |
| 4      | 28.51678   | 5.269878  | 18.13 | 0.000 | 19.85177             | 40.96395 |
| 5      | 40.45121   | 7.775511  | 19.25 | 0.000 | 27.75326             | 58.95885 |
| pyears | (exposure) |           |       |       |                      |          |

In the above, we specified `irr` to obtain IRRs. We estimate that smokers have 1.43 times the mortality rate of nonsmokers. See, however, [example 1](#) in [\[R\] poisson postestimation](#).



## Saved results

`poisson` saves the following in `e()`:

### Scalars

|                            |                                                |
|----------------------------|------------------------------------------------|
| <code>e(N)</code>          | number of observations                         |
| <code>e(k)</code>          | number of parameters                           |
| <code>e(k_eq)</code>       | number of equations                            |
| <code>e(k_eq_model)</code> | number of equations in model Wald test         |
| <code>e(k_dv)</code>       | number of dependent variables                  |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints |
| <code>e(df_m)</code>       | model degrees of freedom                       |
| <code>e(r2_p)</code>       | pseudo- <i>R</i> -squared                      |
| <code>e(ll)</code>         | log likelihood                                 |
| <code>e(ll_0)</code>       | log likelihood, constant-only model            |
| <code>e(N_clust)</code>    | number of clusters                             |
| <code>e(chi2)</code>       | $\chi^2$                                       |
| <code>e(p)</code>          | significance                                   |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                      |
| <code>e(ic)</code>         | number of iterations                           |
| <code>e(rc)</code>         | return code                                    |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                    |

### Macros

|                                 |                                                                          |
|---------------------------------|--------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>poisson</code>                                                     |
| <code>e(cmdline)</code>         | command as typed                                                         |
| <code>e(depvar)</code>          | name of dependent variable                                               |
| <code>e(wtype)</code>           | weight type                                                              |
| <code>e(wexp)</code>            | weight expression                                                        |
| <code>e(title)</code>           | title in estimation output                                               |
| <code>e(clustvar)</code>        | name of cluster variable                                                 |
| <code>e(offset)</code>          | offset                                                                   |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                  |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                           |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                            |
| <code>e(opt)</code>             | type of optimization                                                     |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>       | type of ml method                                                        |
| <code>e(user)</code>            | name of likelihood-evaluator program                                     |
| <code>e(technique)</code>       | maximization technique                                                   |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular              |
| <code>e(crittype)</code>        | optimization criterion                                                   |
| <code>e(properties)</code>      | b V                                                                      |
| <code>e(estat_cmd)</code>       | program used to implement <code>estat</code>                             |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                           |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>           |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>           |

|                 |                                              |
|-----------------|----------------------------------------------|
| Matrices        |                                              |
| e(b)            | coefficient vector                           |
| e(Cns)          | constraints matrix                           |
| e(ilog)         | iteration log (up to 20 iterations)          |
| e(gradient)     | gradient vector                              |
| e(V)            | variance–covariance matrix of the estimators |
| e(V_modelbased) | model-based variance                         |
| Functions       |                                              |
| e(sample)       | marks estimation sample                      |

Siméon-Denis Poisson (1781–1840) was a French mathematician and physicist who contributed to several fields: his name is perpetuated in Poisson brackets, Poisson’s constant, Poisson’s differential equation, Poisson’s integral, and Poisson’s ratio. Among many other results, he produced a version of the law of large numbers. His rather misleadingly titled *Recherches sur la probabilité des jugements* embraces a complete treatise on probability, as the subtitle indicates, including what is now known as the Poisson distribution. That, however, was discovered earlier by the Huguenot–British mathematician Abraham de Moivre (1667–1754).

## Methods and formulas

poisson is implemented as an ado-file.

The log likelihood (with weights  $w_j$  and offsets) is given by

$$\begin{aligned}\Pr(Y = y) &= \frac{e^{-\lambda} \lambda^y}{y!} \\ \xi_j &= \mathbf{x}_j \boldsymbol{\beta} + \text{offset}_j \\ f(y_j) &= \frac{e^{-\exp(\xi_j)} e^{\xi_j y_j}}{y_j!} \\ \ln L &= \sum_{j=1}^n w_j \{ -e^{\xi_j} + \xi_j y_j - \ln(y_j!) \}\end{aligned}$$

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

`poisson` also supports estimation with survey data. For details on VCEs with survey data, see [\[SVY\] variance estimation](#).

## References

Bru, B. 2001. Siméon-Denis Poisson. In *Statisticians of the Centuries*, ed. C. C. Heyde and E. Seneta, 123–126. New York: Springer.

Cameron, A. C., and P. K. Trivedi. 1998. *Regression Analysis of Count Data*. Cambridge: Cambridge University Press.

———. 2009. *Microeconometrics Using Stata*. College Station, TX: Stata Press.

- Chatterjee, S., and A. S. Hadi. 2006. *Regression Analysis by Example*. 4th ed. New York: Wiley.
- Clarke, R. D. 1946. An application of the Poisson distribution. *Journal of the Institute of Actuaries* 72: 481.
- Cochran, W. G. 1940. The analysis of variance when experimental errors follow the Poisson or binomial laws. *Annals of Mathematical Statistics* 11: 335–347.
- . 1982. *Contributions to Statistics*. New York: Wiley.
- Coleman, J. S. 1964. *Introduction to Mathematical Sociology*. New York: Free Press.
- Doll, R., and A. B. Hill. 1966. Mortality of British doctors in relation to smoking: Observations on coronary thrombosis. *Journal of the National Cancer Institute, Monographs* 19: 205–268.
- Hilbe, J. M. 1998. [sg91: Robust variance estimators for MLE Poisson and negative binomial regression](#). *Stata Technical Bulletin* 45: 26–28. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 177–180. College Station, TX: Stata Press.
- . 1999. [sg102: Zero-truncated Poisson and negative binomial regression](#). *Stata Technical Bulletin* 47: 37–40. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 233–236. College Station, TX: Stata Press.
- Hilbe, J. M., and D. H. Judson. 1998. [sg94: Right, left, and uncensored Poisson regression](#). *Stata Technical Bulletin* 46: 18–20. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 186–189. College Station, TX: Stata Press.
- Johnson, N. L., A. W. Kemp, and S. Kotz. 2005. *Univariate Discrete Distributions*. 3rd ed. New York: Wiley.
- Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.
- Long, J. S., and J. Freese. 2001. Predicted probabilities for count models. *Stata Journal* 1: 51–57.
- . 2006. *Regression Models for Categorical Dependent Variables Using Stata*. 2nd ed. College Station, TX: Stata Press.
- McNeil, D. 1996. *Epidemiological Research Methods*. Chichester, UK: Wiley.
- Miranda, A., and S. Rabe-Hesketh. 2006. Maximum likelihood estimation of endogenous switching and sample selection models for binary, ordinal, and count variables. *Stata Journal* 6: 285–308.
- Newman, S. C. 2001. *Biostatistical Methods in Epidemiology*. New York: Wiley.
- Poisson, S. D. 1837. *Recherches sur la probabilité des jugements en matière criminelle et en matière civile: précédées des règles générales du calcul des probabilités*. Paris: Bachelier.
- Rodríguez, G. 1993. [sbe10: An improvement to poisson](#). *Stata Technical Bulletin* 11: 11–14. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 94–98. College Station, TX: Stata Press.
- Rogers, W. H. 1991. [sbe1: Poisson regression with rates](#). *Stata Technical Bulletin* 1: 11–12. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 62–64. College Station, TX: Stata Press.
- Rothman, K. J., S. Greenland, and T. L. Lash. 2008. *Modern Epidemiology*. 3rd ed. Philadelphia: Lippincott Williams & Wilkins.
- Rutherford, E., J. Chadwick, and C. D. Ellis. 1930. *Radiations from Radioactive Substances*. Cambridge: Cambridge University Press.
- Schonlau, M. 2005. [Boosted regression \(boosting\): An introductory tutorial and a Stata plugin](#). *Stata Journal* 5: 330–354.
- Selvin, S. 2004. *Statistical Analysis of Epidemiologic Data*. 3rd ed. New York: Oxford University Press.
- Thorndike, F. 1926. Applications of Poisson's probability summation. *Bell System Technical Journal* 5: 604–624.
- Tobías, A., and M. J. Campbell. 1998. [sg90: Akaike's information criterion and Schwarz's criterion](#). *Stata Technical Bulletin* 45: 23–25. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 174–177. College Station, TX: Stata Press.
- von Bortkewitsch, L. 1898. *Das Gesetz der Kleinen Zahlen*. Leipzig: Teubner.

(Continued on next page)

## Also see

[R] [poisson postestimation](#) — Postestimation tools for poisson

[R] [glm](#) — Generalized linear models

[R] [nbreg](#) — Negative binomial regression

[R] [zip](#) — Zero-inflated Poisson regression

[R] [ztp](#) — Zero-truncated Poisson regression

[SVY] [svy estimation](#) — Estimation commands for survey data

[XT] [xtpoisson](#) — Fixed-effects, random-effects, and population-averaged Poisson models

[U] [20 Estimation and postestimation commands](#)

Description

The following postestimation command is of special interest after `poisson`:

| command                                                                                                                                | description          |
|----------------------------------------------------------------------------------------------------------------------------------------|----------------------|
| <code>estat gof</code>                                                                                                                 | goodness-of-fit test |
| <code>estat gof</code> is not appropriate after the <code>svy</code> prefix. For information about <code>estat gof</code> , see below. |                      |

The following standard postestimation commands are also available:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>            | link test for model specification                                                                   |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Special-interest postestimation command

`estat gof` performs a goodness-of-fit test of the model. The default is the deviance statistic; specifying option `pearson` will give the Pearson statistic. If the test is significant, the Poisson regression model is inappropriate. Then you could try a negative binomial model; see [\[R\] nbreg](#).

## Syntax for predict

```
predict [type] newvar [if] [in] [, statistic nooffset]
```

| statistic | description                                                                |
|-----------|----------------------------------------------------------------------------|
| Main      |                                                                            |
| n         | number of events; the default                                              |
| ir        | incidence rate                                                             |
| xb        | linear prediction                                                          |
| stdp      | standard error of the linear prediction                                    |
| score     | first derivative of the log likelihood with respect to $\mathbf{x}_j\beta$ |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

- n**, the default, calculates the predicted number of events, which is  $\exp(\mathbf{x}_j\beta)$  if neither `offset()` nor `exposure()` was specified when the model was fit;  $\exp(\mathbf{x}_j\beta + \text{offset}_j)$  if `offset()` was specified; or  $\exp(\mathbf{x}_j\beta) \times \text{exposure}_j$  if `exposure()` was specified.
- ir** calculates the incidence rate,  $\exp(\mathbf{x}_j\beta)$ , which is the predicted number of events when exposure is 1. Specifying `ir` is equivalent to specifying `n` when neither `offset()` nor `exposure()` was specified when the model was fit.
- xb** calculates the linear prediction, which is  $\mathbf{x}_j\beta$  if neither `offset()` nor `exposure()` was specified;  $\mathbf{x}_j\beta + \text{offset}_j$  if `offset()` was specified; or  $\mathbf{x}_j\beta + \ln(\text{exposure}_j)$  if `exposure()` was specified; see `nooffset` below.
- stdp** calculates the standard error of the linear prediction.
- score** calculates the equation-level score,  $\partial \ln L / \partial (\mathbf{x}_j\beta)$ .
- nooffset** is relevant only if you specified `offset()` or `exposure()` when you fit the model. It modifies the calculations made by `predict` so that they ignore the offset or exposure variable; the linear prediction is treated as  $\mathbf{x}_j\beta$  rather than as  $\mathbf{x}_j\beta + \text{offset}_j$  or  $\mathbf{x}_j\beta + \ln(\text{exposure}_j)$ . Specifying `predict ... , nooffset` is equivalent to specifying `predict ... , ir`.

## Syntax for estat gof

```
estat gof [, pearson]
```

# Menu

Statistics > Postestimation > Reports and statistics

## Option for estat gof

pearson requests that estat gof calculate the Pearson statistic rather than the deviance statistic.

# Remarks

### ► Example 1

Continuing with [example 2](#) of [\[R\] poisson](#), we use estat gof to determine whether the model fits the data well.

```
. use http://www.stata-press.com/data/r11/dollhill3
. poisson deaths smokes i.agecat, exp(pyears) irr
 (output omitted)
. estat gof
 Goodness-of-fit chi2 = 12.13244
 Prob > chi2(4) = 0.0164
```

The goodness-of-fit  $\chi^2$  tells us that, given the model, we can reject the hypothesis that these data are Poisson distributed at the 1.64% significance level.

So let us now back up and be more careful. We can most easily obtain the incidence-rate ratios within age categories by using irr; see [\[ST\] epitab](#):

```
. irr deaths smokes pyears, by(agecat) nohet
```

| agecat       | IRR      | [95% Conf. Interval] |          | M-H Weight       |
|--------------|----------|----------------------|----------|------------------|
| 1            | 5.736638 | 1.463557             | 49.40468 | 1.472169 (exact) |
| 2            | 2.138812 | 1.173714             | 4.272545 | 9.624747 (exact) |
| 3            | 1.46824  | .9863624             | 2.264107 | 23.34176 (exact) |
| 4            | 1.35606  | .9081925             | 2.096412 | 23.25315 (exact) |
| 5            | .9047304 | .6000757             | 1.399687 | 24.31435 (exact) |
| Crude        | 1.719823 | 1.391992             | 2.14353  | (exact)          |
| M-H combined | 1.424682 | 1.154703             | 1.757784 |                  |

We find that the mortality incidence ratios are greatly different within age category, being highest for the youngest categories and actually dropping below 1 for the oldest. (In the last case, we might argue that those who smoke and who have not died by age 75 are self-selected to be particularly robust.)

Seeing this, we will now parameterize the smoking effects separately for each age category, although we will begin by constraining the smoking effects on age categories 3 and 4 to be equivalent:

(Continued on next page)

```
. constraint 1 smokes#3.agecat = smokes#4.agecat
. poisson deaths c.smokes#agecat i.agecat, exposure(pyears) irr constraints(1)
Iteration 0: log likelihood = -31.95424
Iteration 1: log likelihood = -27.796801
Iteration 2: log likelihood = -27.574177
Iteration 3: log likelihood = -27.572645
Iteration 4: log likelihood = -27.572645

Poisson regression Number of obs = 10
 Wald chi2(8) = 632.14
Log likelihood = -27.572645 Prob > chi2 = 0.0000
(1) [deaths]3.agecat#c.smokes - [deaths]4.agecat#c.smokes = 0
```

| deaths   | IRR        | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|----------|------------|-----------|-------|-------|----------------------|----------|
| agecat#  |            |           |       |       |                      |          |
| c.smokes |            |           |       |       |                      |          |
| 1        | 5.736637   | 4.181256  | 2.40  | 0.017 | 1.374811             | 23.93711 |
| 2        | 2.138812   | .6520701  | 2.49  | 0.013 | 1.176691             | 3.887609 |
| 3        | 1.412229   | .2017485  | 2.42  | 0.016 | 1.067343             | 1.868557 |
| 4        | 1.412229   | .2017485  | 2.42  | 0.016 | 1.067343             | 1.868557 |
| 5        | .9047304   | .1855513  | -0.49 | 0.625 | .6052658             | 1.35236  |
| agecat   |            |           |       |       |                      |          |
| 2        | 10.5631    | 8.067701  | 3.09  | 0.002 | 2.364153             | 47.19623 |
| 3        | 47.671     | 34.37409  | 5.36  | 0.000 | 11.60056             | 195.8978 |
| 4        | 98.22765   | 70.85012  | 6.36  | 0.000 | 23.89324             | 403.8244 |
| 5        | 199.2099   | 145.3356  | 7.26  | 0.000 | 47.67693             | 832.3648 |
| pyears   | (exposure) |           |       |       |                      |          |

```
. estat gof
 Goodness-of-fit chi2 = .0774185
 Prob > chi2(1) = 0.7808
```

The goodness-of-fit  $\chi^2$  is now small; we are no longer running roughshod over the data. Let us now consider simplifying the model. The point estimate of the incidence-rate ratio for smoking in age category 1 is much larger than that for smoking in age category 2, but the confidence interval for `smokes#1.agecat` is similarly wide. Is the difference real?

```
. test smokes#1.agecat = smokes#2.agecat
(1) [deaths]1b.agecat#c.smokes - [deaths]2.agecat#c.smokes = 0
 chi2(1) = 1.56
 Prob > chi2 = 0.2117
```

The point estimates may be far apart, but there is insufficient data, and we may be observing random differences. With that success, might we also combine the smokers in age categories 3 and 4 with those in 1 and 2?

```
. test smokes#2.agecat = smokes#3.agecat, accum
(1) [deaths]1b.agecat#c.smokes - [deaths]2.agecat#c.smokes = 0
(2) [deaths]2.agecat#c.smokes - [deaths]3.agecat#c.smokes = 0
 chi2(2) = 4.73
 Prob > chi2 = 0.0938
```

Combining age categories 1–4 may be overdoing it—the 9.38% significance level is enough to stop us, although others may disagree.



Thus we now fit our final model:

```
. constraint 2 smokes#1.agecat = smokes#2.agecat
. poisson deaths c.smokes#agecat i.agecat, exposure(pyears) irr constraints(1/2)
Iteration 0: log likelihood = -31.550722
Iteration 1: log likelihood = -28.525057
Iteration 2: log likelihood = -28.514535
Iteration 3: log likelihood = -28.514535

Poisson regression Number of obs = 10
 Wald chi2(7) = 642.25
Log likelihood = -28.514535 Prob > chi2 = 0.0000
(1) [deaths]3.agecat#c.smokes - [deaths]4.agecat#c.smokes = 0
(2) [deaths]1b.agecat#c.smokes - [deaths]2.agecat#c.smokes = 0
```

| deaths   |            | IRR      | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|----------|------------|----------|-----------|-------|-------|----------------------|----------|
| agecat#  |            |          |           |       |       |                      |          |
| c.smokes |            |          |           |       |       |                      |          |
|          | 1          | 2.636259 | .7408403  | 3.45  | 0.001 | 1.519791             | 4.572907 |
|          | 2          | 2.636259 | .7408403  | 3.45  | 0.001 | 1.519791             | 4.572907 |
|          | 3          | 1.412229 | .2017485  | 2.42  | 0.016 | 1.067343             | 1.868557 |
|          | 4          | 1.412229 | .2017485  | 2.42  | 0.016 | 1.067343             | 1.868557 |
|          | 5          | .9047304 | .1855513  | -0.49 | 0.625 | .6052658             | 1.35236  |
| agecat   |            |          |           |       |       |                      |          |
|          | 2          | 4.294559 | .8385329  | 7.46  | 0.000 | 2.928987             | 6.296797 |
|          | 3          | 23.42263 | 7.787716  | 9.49  | 0.000 | 12.20738             | 44.94164 |
|          | 4          | 48.26309 | 16.06939  | 11.64 | 0.000 | 25.13068             | 92.68856 |
|          | 5          | 97.87965 | 34.30881  | 13.08 | 0.000 | 49.24123             | 194.561  |
| pyears   | (exposure) |          |           |       |       |                      |          |

The above strikes us as a fair representation of the data.



Methods and formulas

All postestimation commands listed above are implemented as ado-files.

In the following, we use the same notation as in [R] poisson.

The equation-level scores are given by

$$\text{score}(\mathbf{x}\beta)_j = y_j - e^{\xi_j}$$

The deviance (D) and Pearson (P) goodness-of-fit statistics are given by

$$\begin{aligned} \ln L_{\max} &= \sum_{j=1}^n w_j [-y_j \{ \ln(y_j) - 1 \} - \ln(y_j!)] \\ \chi_D^2 &= -2 \{ \ln L - \ln L_{\max} \} \\ \chi_P^2 &= \sum_{j=1}^n \frac{w_j (y_j - e^{\xi_j})^2}{e^{\xi_j}} \end{aligned}$$

## Also see

[R] [poisson](#) — Poisson regression

[U] [20 Estimation and postestimation commands](#)

Title

**predict** — Obtain predictions, residuals, etc., after estimation

Syntax

After single-equation (SE) models

```
predict [type] newvar [if] [in] [, single_options]
```

After multiple-equation (ME) models

```
predict [type] newvar [if] [in] [, multiple_options]
predict [type] { stub*|newvar1 ... newvarq } [if] [in] , scores
```

| <i>single_options</i>  | description                                                                 |
|------------------------|-----------------------------------------------------------------------------|
| Main                   |                                                                             |
| <b>xb</b>              | calculate linear prediction                                                 |
| <b>stdp</b>            | calculate standard error of the prediction                                  |
| <b><u>score</u></b>    | calculate first derivative of the log likelihood with respect to $x_j\beta$ |
| Options                |                                                                             |
| <b><u>nooffset</u></b> | ignore any <code>offset()</code> or <code>exposure()</code> variable        |
| <i>other_options</i>   | command-specific options                                                    |

| <i>multiple_options</i>                                 | description                                                          |
|---------------------------------------------------------|----------------------------------------------------------------------|
| Main                                                    |                                                                      |
| <b><u>equation</u></b> ( <i>eqno</i> [ , <i>eqno</i> ]) | specify equations                                                    |
| <b>xb</b>                                               | calculate linear prediction                                          |
| <b>stdp</b>                                             | calculate standard error of the prediction                           |
| <b>stddp</b>                                            | calculate the difference in linear predictions                       |
| Options                                                 |                                                                      |
| <b><u>nooffset</u></b>                                  | ignore any <code>offset()</code> or <code>exposure()</code> variable |
| <i>other_options</i>                                    | command-specific options                                             |

Menu

Statistics > Postestimation > Predictions, residuals, etc.

Description

`predict` calculates predictions, residuals, influence statistics, and the like after estimation. Exactly what `predict` can do is determined by the previous estimation command; command-specific options are documented with each estimation command. Regardless of command-specific options, the actions of `predict` share certain similarities across estimation commands:

1. `predict newvar` creates *newvar* containing “predicted values”—numbers related to the  $E(y_j|x_j)$ . For instance, after linear regression, `predict newvar` creates  $\mathbf{x}_j\mathbf{b}$  and, after probit, creates the probability  $\Phi(\mathbf{x}_j\mathbf{b})$ .
2. `predict newvar, xb` creates *newvar* containing  $\mathbf{x}_j\mathbf{b}$ . This may be the same result as option 1 (e.g., linear regression) or different (e.g., probit), but regardless, option `xb` is allowed.
3. `predict newvar, stdp` creates *newvar* containing the standard error of the linear prediction  $\mathbf{x}_j\mathbf{b}$ .
4. `predict newvar, other_options` may create *newvar* containing other useful quantities; see `help` or the reference manual entry for the particular estimation command to find out about other available options.
5. `nooffset` added to any of the above commands requests that the calculation ignore any offset or exposure variable specified by including the `offset(varnameo)` or `exposure(varnamee)` option when you fit the model.

`predict` can be used to make in-sample or out-of-sample predictions:

6. `predict` calculates the requested statistic for all possible observations, whether they were used in fitting the model or not. `predict` does this for standard options 1–3 and generally does this for estimator-specific options 4.
7. `predict newvar if e(sample), ...` restricts the prediction to the estimation subsample.
8. Some statistics make sense only with respect to the estimation subsample. In such cases, the calculation is automatically restricted to the estimation subsample, and the documentation for the specific option states this. Even so, you can still specify `if e(sample)` if you are uncertain.
9. `predict` can make out-of-sample predictions even using other datasets. In particular, you can

```
. use ds1
. (fit a model)
. use two /* another dataset */
. predict yhat, ... /* fill in the predictions */
```

## Options

### Main

`xb` calculates the linear prediction from the fitted model. All models can be thought of as estimating a set of parameters  $b_1, b_2, \dots, b_k$ , and the linear prediction is  $\hat{y}_j = b_1x_{1j} + b_2x_{2j} + \dots + b_kx_{kj}$ , often written in matrix notation as  $\hat{\mathbf{y}}_j = \mathbf{x}_j\mathbf{b}$ . For linear regression, the values  $\hat{y}_j$  are called the predicted values or, for out-of-sample predictions, the forecast. For logit and probit, for example,  $\hat{y}_j$  is called the logit or probit index.

$x_{1j}, x_{2j}, \dots, x_{kj}$  are obtained from the data currently in memory and do not necessarily correspond to the data on the independent variables used to fit the model (obtaining  $b_1, b_2, \dots, b_k$ ).

`stdp` calculates the standard error of the linear prediction. Here the prediction means the same thing as the “index”, namely,  $\mathbf{x}_j\mathbf{b}$ . The statistic produced by `stdp` can be thought of as the standard error of the predicted expected value, or mean index, for the observation’s covariate pattern. The standard error of the prediction is also commonly referred to as the standard error of the fitted value. The calculation can be made in or out of sample.

`stdp` is allowed only after you have previously fit a multiple-equation model. The standard error of the difference in linear predictions ( $\mathbf{x}_{1j}\mathbf{b} - \mathbf{x}_{2j}\mathbf{b}$ ) between equations 1 and 2 is calculated. This option requires that `equation(eqno1,eqno2)` be specified.

`score` calculates the equation-level score,  $\partial \ln L / \partial (\mathbf{x}_j \beta)$ . Here  $\ln L$  refers to the log-likelihood function.

`scores` is the ME model equivalent of the `score` option, resulting in multiple equation-level score variables. An equation-level score variable is created for each equation in the model; ancillary parameters—such as  $\ln \sigma$  and  $\text{atanh} \rho$ —make up separate equations.

`equation(eqno1,eqno2)`—synonym `outcome()`—is relevant only when you have previously fit a multiple-equation model. It specifies the equation to which you are referring.

`equation()` is typically filled in with one `eqno`—it would be filled in that way with options `xb` and `stdp`, for instance. `equation(#1)` would mean the calculation is to be made for the first equation, `equation(#2)` would mean the second, and so on. You could also refer to the equations by their names. `equation(income)` would refer to the equation named `income` and `equation(hours)` to the equation named `hours`.

If you do not specify `equation()`, results are the same as if you specified `equation(#1)`.

Other statistics, such as `stdp`, refer to between-equation concepts. In those cases, you might specify `equation(#1,#2)` or `equation(income,hours)`. When two equations must be specified, `equation()` is required.

#### Options

`nooffset` may be combined with most statistics and specifies that the calculation should be made, ignoring any offset or exposure variable specified when the model was fit.

This option is available, even if it is not documented for `predict` after a specific command. If neither the `offset(varnameo)` option nor the `exposure(varnamee)` option was specified when the model was fit, specifying `nooffset` does nothing.

*other\_options* refers to command-specific options that are documented with each command.

## Remarks

Remarks are presented under the following headings:

- Estimation-sample predictions*
- Out-of-sample predictions*
- Residuals*
- Single-equation (SE) models*
- SE model scores*
- Multiple-equation (ME) models*
- ME model scores*

Most of the examples are presented using linear regression, but the general syntax is applicable to all estimators.

You can think of any estimation command as estimating a set of coefficients  $b_1, b_2, \dots, b_k$  corresponding to the variables  $x_1, x_2, \dots, x_k$ , along with a (possibly empty) set of ancillary statistics  $\gamma_1, \gamma_2, \dots, \gamma_m$ . All estimation commands save the  $b_i$ s and  $\gamma_i$ s. `predict` accesses that saved information and combines it with the data currently in memory to make various calculations. For instance, `predict` can calculate the linear prediction,  $\hat{y}_j = b_1 x_{1j} + b_2 x_{2j} + \dots + b_k x_{kj}$ . The data on which `predict` makes the calculation can be the same data used to fit the model or a different dataset—it does not matter. `predict` uses the saved parameter estimates from the model, obtains

the corresponding values of  $x$  for each observation in the data, and then combines them to produce the desired result.

### Estimation-sample predictions

► Example 1

We have a 74-observation dataset on automobiles, including the mileage rating (`mpg`), the car’s weight (`weight`), and whether the car is foreign (`foreign`). We fit the model

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. regress mpg weight if foreign
```

| Source   | SS         | df | MS         |
|----------|------------|----|------------|
| Model    | 427.990298 | 1  | 427.990298 |
| Residual | 489.873338 | 20 | 24.4936669 |
| Total    | 917.863636 | 21 | 43.7077922 |

|                 |          |
|-----------------|----------|
| Number of obs = | 22       |
| F( 1, 20) =     | 17.47    |
| Prob > F        | = 0.0005 |
| R-squared       | = 0.4663 |
| Adj R-squared = | 0.4396   |
| Root MSE        | = 4.9491 |

| mpg    | Coef.    | Std. Err. | t     | P> t  | [95% Conf. Interval] |
|--------|----------|-----------|-------|-------|----------------------|
| weight | -.010426 | .0024942  | -4.18 | 0.000 | -.0156287 -.0052232  |
| _cons  | 48.9183  | 5.871851  | 8.33  | 0.000 | 36.66983 61.16676    |

If we were to type `predict pmpg` now, we would obtain the linear predictions for all 74 observations. To obtain the predictions just for the sample on which we fit the model, we could type

```
. predict pmpg if e(sample)
(option xb assumed; fitted values)
(52 missing values generated)
```

Here `e(sample)` is true only for foreign cars because we typed `if foreign` when we fit the model and because there are no missing values among the relevant variables. If there had been missing values, `e(sample)` would also account for those.

By the way, the `if e(sample)` restriction can be used with any Stata command, so we could obtain summary statistics on the estimation sample by typing

```
. summarize if e(sample)
(output omitted)
```



### Out-of-sample predictions

By out-of-sample predictions, we mean predictions extending beyond the estimation sample. In the example above, typing `predict pmpg` would generate linear predictions using all 74 observations.

`predict` will work on other datasets, too. You can use a new dataset and type `predict` to obtain results for that sample.

## ► Example 2

Using the same auto dataset, assume that we wish to fit the model

$$\text{mpg} = \beta_1 \text{weight} + \beta_2 \ln(\text{weight}) + \beta_3 \text{foreign} + \beta_4$$

We first create the  $\ln(\text{weight})$  variable, and then type the `regress` command:

```
. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)
. generate lnweight = ln(weight)
. regress mpg weight lnweight foreign
```

| Source   | SS         | df | MS         | Number of obs = 74 |   |        |
|----------|------------|----|------------|--------------------|---|--------|
| Model    | 1690.27997 | 3  | 563.426657 | F( 3, 70)          | = | 52.36  |
| Residual | 753.179489 | 70 | 10.759707  | Prob > F           | = | 0.0000 |
|          |            |    |            | R-squared          | = | 0.6918 |
|          |            |    |            | Adj R-squared      | = | 0.6785 |
| Total    | 2443.45946 | 73 | 33.4720474 | Root MSE           | = | 3.2802 |

| mpg      | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| weight   | .003304   | .0038995  | 0.85  | 0.400 | -.0044734            | .0110813  |
| lnweight | -29.59133 | 11.52018  | -2.57 | 0.012 | -52.5676             | -6.615061 |
| foreign  | -2.125299 | 1.052324  | -2.02 | 0.047 | -4.224093            | -.0265044 |
| _cons    | 248.0548  | 80.37079  | 3.09  | 0.003 | 87.76035             | 408.3493  |

If we typed `predict pmpg` now, we would obtain predictions for all 74 cars in the current data. Instead, we are going to use a new dataset.

The dataset `newautos.dta` contains the make, weight, and place of manufacture of two cars, the Pontiac Sunbird and the Volvo 260. Let's use the dataset and create the predictions:

```
. use http://www.stata-press.com/data/r11/newautos, clear
(New Automobile Models)
. list
```

|    | make          | weight | foreign  |
|----|---------------|--------|----------|
| 1. | Pont. Sunbird | 2690   | Domestic |
| 2. | Volvo 260     | 3170   | Foreign  |

```
. predict mpg
(option xb assumed; fitted values)
variable lnweight not found
r(111);
```

Things did not work. We typed `predict mpg`, and Stata responded with the message “variable `lnweight` not found”. `predict` can calculate predicted values on a different dataset only if that dataset contains the variables that went into the model. Here our dataset does not contain a variable called `lnweight`. `lnweight` is just the log of weight, so we can create it and try again:

```
. generate lnweight = ln(weight)
. predict mpg
(option xb assumed; fitted values)
```

```
. list
```

|    | make          | weight | foreign  | lnweight | mpg      |
|----|---------------|--------|----------|----------|----------|
| 1. | Pont. Sunbird | 2690   | Domestic | 7.897296 | 23.25097 |
| 2. | Volvo 260     | 3170   | Foreign  | 8.061487 | 17.85295 |

We obtained our predicted values. The Pontiac Sunbird has a predicted mileage rating of 23.3 mpg, whereas the Volvo 260 has a predicted rating of 17.9 mpg.



Residuals

➤ Example 3

With many estimators, `predict` can calculate more than predicted values. With most regression-type estimators, we can, for instance, obtain residuals. Using our regression example, we return to our original data and obtain residuals by typing

```
. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)
. generate lnweight = ln(weight)
. regress mpg weight lnweight foreign
(output omitted)
. predict double resid, residuals
. summarize resid
```

| Variable | Obs | Mean      | Std. Dev. | Min       | Max      |
|----------|-----|-----------|-----------|-----------|----------|
| resid    | 74  | -1.31e-14 | 3.212091  | -5.453078 | 13.83719 |

We could do this without refitting the model. Stata always remembers the last set of estimates, even as we use new datasets.

It was not necessary to type the `double` in `predict double resid, residuals`, but we wanted to remind you that you can specify the type of a variable in front of the variable’s name; see [U] 11.4.2 Lists of new variables. We made the new variable `resid` a `double` rather than the default `float`.

If you want your residuals to have a mean as close to zero as possible, remember to request the extra precision of `double`. If we had not specified `double`, the mean of `resid` would have been roughly  $10^{-9}$  rather than  $10^{-14}$ . Although  $10^{-14}$  sounds more precise than  $10^{-9}$ , the difference really does not matter.



For linear regression, `predict` can also calculate standardized residuals and studentized residuals with the options `rstandard` and `rstudent`; for examples, see [R] regress postestimation.



## Single-equation (SE) models

If you have not read the discussion above on using `predict` after linear regression, please do so. And `predict`'s default calculation almost always produces a statistic in the same metric as the dependent variable of the fitted model—e.g., predicted counts for Poisson regression. In any case, `xb` can always be specified to obtain the linear prediction.

`predict` can calculate the standard error of the prediction, which is obtained by using the covariance matrix of the estimators.

### ► Example 4

After most binary outcome models (e.g., `logistic`, `logit`, `probit`, `cloglog`, `scobit`), `predict` calculates the probability of a positive outcome if we do not tell it otherwise. We can specify the `xb` option if we want the linear prediction (also known as the logit or probit index). The odd abbreviation `xb` is meant to suggest  $\mathbf{x}\beta$ . In logit and probit models, for example, the predicted probability is  $p = F(\mathbf{x}\beta)$ , where  $F()$  is the logistic or normal cumulative distribution function, respectively.

```
. logistic foreign mpg weight
 (output omitted)
. predict phat
(option pr assumed; Pr(foreign))
. predict idxhat, xb
. summarize foreign phat idxhat
```

| Variable | Obs | Mean      | Std. Dev. | Min       | Max      |
|----------|-----|-----------|-----------|-----------|----------|
| foreign  | 74  | .2972973  | .4601885  | 0         | 1        |
| phat     | 74  | .2972973  | .3052979  | .000729   | .8980594 |
| idxhat   | 74  | -1.678202 | 2.321509  | -7.223107 | 2.175845 |

Because this is a logit model, we could obtain the predicted probabilities ourselves from the predicted index

```
. generate phat2 = exp(idxhat)/(1+exp(idxhat))
```

but using `predict` without options is easier.



### ► Example 5

For all models, `predict` attempts to produce a predicted value in the same metric as the dependent variable of the model. We have seen that for dichotomous outcome models, the default statistic produced by `predict` is the probability of a success. Similarly, for Poisson regression, the default statistic produced by `predict` is the predicted count for the dependent variable. You can always specify the `xb` option to obtain the linear combination of the coefficients with an observation's  $x$  values (the inner product of the coefficients and  $x$  values). For `poisson` (without an explicit exposure), this is the natural log of the count.

```
. use http://www.stata-press.com/data/r11/airline, clear
. poisson injuries XYZowned
 (output omitted)
. predict injhat
(option n assumed; predicted number of events)
```

```
. predict idx, xb
. generate exp_idx = exp(idx)
. summarize injuries injhat exp_idx idx
```

| Variable | Obs | Mean     | Std. Dev. | Min      | Max      |
|----------|-----|----------|-----------|----------|----------|
| injuries | 9   | 7.111111 | 5.487359  | 1        | 19       |
| injhat   | 9   | 7.111111 | .8333333  | 6        | 7.666667 |
| exp_idx  | 9   | 7.111111 | .8333333  | 6        | 7.666667 |
| idx      | 9   | 1.955174 | .1225612  | 1.791759 | 2.036882 |

We note that our “hand-computed” prediction of the count (`exp_idx`) matches what was produced by the default operation of `predict`.

If our model has an exposure-time variable, we can use `predict` to obtain the linear prediction with or without the exposure. Let’s verify what we are getting by obtaining the linear prediction with and without exposure, transforming these predictions to count predictions and comparing them with the default count prediction from `predict`. We must remember to multiply by the exposure time when using `predict ... , nooffset`.

```
. use http://www.stata-press.com/data/r11/airline, clear
. poisson injuries XYZowned, exposure(n)
 (output omitted)
. predict double injhat
(option n assumed; predicted number of events)
. predict double idx, xb
. generate double exp_idx = exp(idx)
. predict double idxn, xb nooffset
. generate double exp_idxn = exp(idxn)*n
. summarize injuries injhat exp_idx exp_idxn idx idxn
```

| Variable | Obs | Mean     | Std. Dev. | Min      | Max      |
|----------|-----|----------|-----------|----------|----------|
| injuries | 9   | 7.111111 | 5.487359  | 1        | 19       |
| injhat   | 9   | 7.111111 | 3.10936   | 2.919621 | 12.06158 |
| exp_idx  | 9   | 7.111111 | 3.10936   | 2.919621 | 12.06158 |
| exp_idxn | 9   | 7.111111 | 3.10936   | 2.919621 | 12.06158 |
| idx      | 9   | 1.869722 | .4671044  | 1.071454 | 2.490025 |
| idxn     | 9   | 4.18814  | .1904042  | 4.061204 | 4.442013 |

Looking at the identical means and standard deviations for `injhat`, `exp_idx`, and `exp_idxn`, we see that we can reproduce the default computations of `predict` for poisson estimations. We have also demonstrated the relationship between the count predictions and the linear predictions with and without exposure.



SE model scores

► Example 6

With most maximum likelihood estimators, `predict` can calculate equation-level scores. The first derivative of the log likelihood with respect to  $\mathbf{x}_j\beta$  is the equation-level score.

```
. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)
. logistic foreign mpg weight
 (output omitted)
```

```
. predict double sc, score
```

```
. summarize sc
```

| Variable | Obs | Mean      | Std. Dev. | Min       | Max      |
|----------|-----|-----------|-----------|-----------|----------|
| sc       | 74  | -1.05e-13 | .3533133  | -.8760856 | .8821309 |

See [\[P\] `\_robust`](#) and [\[SVY\] variance estimation](#) for details regarding the role equation-level scores play in linearization-based variance estimators.



## □ Technical note

`predict` after some estimation commands, such as `regress` and `cnsreg`, allows the `score` option as a synonym for the `residuals` option.



## Multiple-equation (ME) models

If you have not read the above discussion on using `predict` after SE models, please do so. With the exception of the ability to select specific equations to predict from, the use of `predict` after ME models follows almost the same form that it does for SE models.

## ▷ Example 7

The details of prediction statistics that are specific to particular ME models are documented with the estimation command. If you are using ME commands that do not have separate discussions on obtaining predictions, read the `predict` section in [\[R\] `mlogit`](#), even if your interest is not in multinomial logistic regression. As a general introduction to the ME models, we will demonstrate `predict` after `sureg`:

```
. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)
```

```
. sureg (price foreign displ) (weight foreign length)
```

Seemingly unrelated regression

| Equation | Obs | Parms | RMSE     | "R-sq" | chi2   | P      |
|----------|-----|-------|----------|--------|--------|--------|
| price    | 74  | 2     | 2202.447 | 0.4348 | 45.21  | 0.0000 |
| weight   | 74  | 2     | 245.5238 | 0.8988 | 658.85 | 0.0000 |

|              | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|--------------|-----------|-----------|-------|-------|----------------------|-----------|
| price        |           |           |       |       |                      |           |
| foreign      | 3137.894  | 697.3805  | 4.50  | 0.000 | 1771.054             | 4504.735  |
| displacement | 23.06938  | 3.443212  | 6.70  | 0.000 | 16.32081             | 29.81795  |
| _cons        | 680.8438  | 859.8142  | 0.79  | 0.428 | -1004.361            | 2366.049  |
| weight       |           |           |       |       |                      |           |
| foreign      | -154.883  | 75.3204   | -2.06 | 0.040 | -302.5082            | -7.257674 |
| length       | 30.67594  | 1.531981  | 20.02 | 0.000 | 27.67331             | 33.67856  |
| _cons        | -2699.498 | 302.3912  | -8.93 | 0.000 | -3292.173            | -2106.822 |

sureg estimated two equations, one called price and the other weight; see [R] sureg.

```
. predict pred_p, equation(price)
(option xb assumed; fitted values)
. predict pred_w, equation(weight)
(option xb assumed; fitted values)
. summarize price pred_p weight pred_w
```

| Variable | Obs | Mean     | Std. Dev. | Min      | Max      |
|----------|-----|----------|-----------|----------|----------|
| price    | 74  | 6165.257 | 2949.496  | 3291     | 15906    |
| pred_p   | 74  | 6165.257 | 1678.805  | 2664.81  | 10485.33 |
| weight   | 74  | 3019.459 | 777.1936  | 1760     | 4840     |
| pred_w   | 74  | 3019.459 | 726.0468  | 1501.602 | 4447.996 |

You may specify the equation by name, as we did above, or by number: equation(#1) means the same thing as equation(price) in this case.

## ME model scores

### Example 8

For ME models, predict allows you to specify a stub when generating equation-level score variables. predict generates new variables using this stub by appending an equation index. Depending upon the command, the index will start with 0 or 1. Here is an example where predict starts indexing the score variables with 0.

```
. ologit rep78 mpg weight
(output omitted)
. predict double sc*, scores
. summarize sc*
```

| Variable | Obs | Mean      | Std. Dev. | Min       | Max      |
|----------|-----|-----------|-----------|-----------|----------|
| sc0      | 69  | 5.81e-18  | .5337363  | -.9854088 | .921433  |
| sc1      | 69  | -1.69e-17 | .186919   | -.2738537 | .9854088 |
| sc2      | 69  | 3.30e-17  | .4061637  | -.5188487 | 1.130178 |
| sc3      | 69  | -1.41e-17 | .5315368  | -1.067351 | .8194842 |
| sc4      | 69  | -4.83e-18 | .360525   | -.921433  | .6140182 |

Although it involves much more typing, we could also specify the new variable names individually.

```
. predict double (sc_xb sc_1 sc_2 sc_3 sc_4), scores
. summarize sc_*
```

| Variable | Obs | Mean      | Std. Dev. | Min       | Max      |
|----------|-----|-----------|-----------|-----------|----------|
| sc_xb    | 69  | 5.81e-18  | .5337363  | -.9854088 | .921433  |
| sc_1     | 69  | -1.69e-17 | .186919   | -.2738537 | .9854088 |
| sc_2     | 69  | 3.30e-17  | .4061637  | -.5188487 | 1.130178 |
| sc_3     | 69  | -1.41e-17 | .5315368  | -1.067351 | .8194842 |
| sc_4     | 69  | -4.83e-18 | .360525   | -.921433  | .6140182 |

## Methods and formulas

`predict` is implemented as an ado-file.

Denote the previously estimated coefficient vector as  $\mathbf{b}$  and its estimated variance matrix as  $\mathbf{V}$ . `predict` works by recalling various aspects of the model, such as  $\mathbf{b}$ , and combining that information with the data currently in memory. Let's write  $\mathbf{x}_j$  for the  $j$ th observation currently in memory.

The *predicted value* (`xb` option) is defined as  $\hat{y}_j = \mathbf{x}_j \mathbf{b} + \text{offset}_j$

The *standard error of the prediction* (the `stdp` option) is defined as  $s_{p_j} = \sqrt{\mathbf{x}_j \mathbf{V} \mathbf{x}_j'}$

The *standard error of the difference in linear predictions* between equations 1 and 2 is defined as

$$s_{dp_j} = \{(\mathbf{x}_{1j}, -\mathbf{x}_{2j}, \mathbf{0}, \dots, \mathbf{0}) \mathbf{V} (\mathbf{x}_{1j}, -\mathbf{x}_{2j}, \mathbf{0}, \dots, \mathbf{0})'\}^{\frac{1}{2}}$$

See the individual estimation commands for information about calculating command-specific `predict` statistics.

## Also see

[R] [predictnl](#) — Obtain nonlinear predictions, standard errors, etc., after estimation

[P] [\\_predict](#) — Obtain predictions, residuals, etc., after estimation programming command

[U] [20 Estimation and postestimation commands](#)

**predictnl** — Obtain nonlinear predictions, standard errors, etc., after estimation

## Syntax

`predictnl` [*type*] *newvar* = *pnl\_exp* [*if*] [*in*] [, *options*]

| <i>options</i>                       | description                                                                                                  |
|--------------------------------------|--------------------------------------------------------------------------------------------------------------|
| Main                                 |                                                                                                              |
| <code>se(<i>newvar</i>)</code>       | create <i>newvar</i> containing standard errors                                                              |
| <code>variance(<i>newvar</i>)</code> | create <i>newvar</i> containing variances                                                                    |
| <code>wald(<i>newvar</i>)</code>     | create <i>newvar</i> containing the Wald test statistic                                                      |
| <code>p(<i>newvar</i>)</code>        | create <i>newvar</i> containing the significance level ( <i>p</i> -value) of the Wald test                   |
| <code>ci(<i>newvars</i>)</code>      | create <i>newvars</i> containing lower and upper confidence intervals                                        |
| <code>level(#)</code>                | set confidence level; default is <code>level(95)</code>                                                      |
| <code>g(<i>stub</i>)</code>          | create <i>stub1</i> , <i>stub2</i> , ..., <i>stubk</i> variables containing observation-specific derivatives |
| Advanced                             |                                                                                                              |
| <code>iterate(#)</code>              | maximum iterations for finding optimal step size; default is 100                                             |
| <code>force</code>                   | calculate standard errors, etc., even when possibly inappropriate                                            |

## Menu

Statistics > Postestimation > Nonlinear predictions

## Description

`predictnl` calculates (possibly) nonlinear predictions after any Stata estimation command and optionally calculates the variances, standard errors, Wald test statistics, significance levels, and confidence limits for these predictions. Unlike its companion nonlinear postestimation commands `testnl` and `nlcom`, `predictnl` generates functions of the data (i.e., predictions), not scalars. The quantities generated by `predictnl` are thus vectorized over the observations in the data.

Consider some general prediction,  $g(\boldsymbol{\theta}, \mathbf{x}_i)$ , for  $i = 1, \dots, n$ , where  $\boldsymbol{\theta}$  are the model parameters and  $\mathbf{x}_i$  are some data for the  $i$ th observation;  $\mathbf{x}_i$  is assumed fixed. Typically,  $g(\boldsymbol{\theta}, \mathbf{x}_i)$  is estimated by  $g(\hat{\boldsymbol{\theta}}, \mathbf{x}_i)$ , where  $\hat{\boldsymbol{\theta}}$  are the estimated model parameters, which are stored in `e(b)` following any Stata estimation command.

In its most common use, `predictnl` generates two variables: one containing the estimated prediction,  $g(\hat{\boldsymbol{\theta}}, \mathbf{x}_i)$ , the other containing the estimated standard error of  $g(\hat{\boldsymbol{\theta}}, \mathbf{x}_i)$ . The calculation of standard errors (and other obtainable quantities that are based on the standard errors, such as test statistics) is based on the delta method, an approximation appropriate in large samples; see [Methods and formulas](#).

`predictnl` can be used with `svy` estimation results (assuming that `predict` is also allowed), see [\[SVY\] svy postestimation](#).

The specification of  $g(\hat{\theta}, \mathbf{x}_i)$  is handled by specifying *pnl\_exp*, and the values of  $g(\hat{\theta}, \mathbf{x}_i)$  are stored in the new variable *newvar* of storage type *type*. *pnl\_exp* is any valid Stata expression and may also contain calls to two special functions unique to **predictnl**:

1. **predict**([*predict\_options*]): When you are evaluating *pnl\_exp*, **predict()** is a convenience function that replicates the calculation performed by the command

```
predict ..., predict_options
```

As such, the **predict()** function may be used either as a shorthand for the formula used to make this prediction or when the formula is not readily available. When used without arguments, **predict()** replicates the default prediction for that particular estimation command.

2. **xb**([*eqno*]): The **xb()** function replicates the calculation of the linear predictor  $\mathbf{x}_i\mathbf{b}$  for equation *eqno*. If **xb()** is specified without *eqno*, the linear predictor for the first equation (or the only equation in single-equation estimation) is obtained.

For example, **xb**(#1) (or equivalently, **xb()** with no arguments) translates to the linear predictor for the first equation, **xb**(#2) for the second, and so on. You could also refer to the equations by their names, such as **xb**(*income*).

When specifying *pnl\_exp*, both of these functions may be used repeatedly, in combination, and in combination with other Stata functions and expressions. See *Remarks* for examples that use both of these functions.

## Options

### Main

**se**(*newvar*) adds *newvar* of storage type *type*, where for each *i* in the prediction sample, *newvar*[*i*] contains the estimated standard error of  $g(\hat{\theta}, \mathbf{x}_i)$ .

**variance**(*newvar*) adds *newvar* of storage type *type*, where for each *i* in the prediction sample, *newvar*[*i*] contains the estimated variance of  $g(\hat{\theta}, \mathbf{x}_i)$ .

**wald**(*newvar*) adds *newvar* of storage type *type*, where for each *i* in the prediction sample, *newvar*[*i*] contains the Wald test statistic for the test of the hypothesis  $H_0: g(\theta, \mathbf{x}_i) = 0$ .

**p**(*newvar*) adds *newvar* of storage type *type*, where *newvar*[*i*] contains the significance level (*p*-value) of the Wald test of  $H_0: g(\theta, \mathbf{x}_i) = 0$  versus the two-sided alternative.

**ci**(*newvars*) requires the specification of two *newvars*, such that the *i*th observation of each will contain the left and right endpoints (respectively) of a confidence interval for  $g(\theta, \mathbf{x}_i)$ . The level of the confidence intervals is determined by **level**(#).

**level**(#) specifies the confidence level, as a percentage, for confidence intervals. The default is **level**(95) or as set by **set level**; see [U] 20.7 Specifying the width of confidence intervals.

**g**(*stub*) specifies that new variables, *stub1*, *stub2*, ..., *stubk* be created, where *k* is the dimension of  $\theta$ . *stub1* will contain the observation-specific derivatives of  $g(\theta, \mathbf{x}_i)$  with respect to the first element,  $\theta_1$ , of  $\theta$ ; *stub2* will contain the derivatives of  $g(\theta, \mathbf{x}_i)$  with respect to  $\theta_2$ , etc.; If the derivative of  $g(\theta, \mathbf{x}_i)$  with respect to a particular coefficient in  $\theta$  equals zero for all observations in the prediction sample, the *stub* variable for that coefficient is not created. The ordering of the parameters in  $\theta$  is precisely that of the stored vector of parameter estimates  $\mathbf{e}(\mathbf{b})$ .

### Advanced

`iterate(#)` specifies the maximum number of iterations used to find the optimal step size in the calculation of numerical derivatives of  $g(\theta, \mathbf{x}_i)$  with respect to  $\theta$ . By default, the maximum number of iterations is 100, but convergence is usually achieved after only a few iterations. You should rarely have to use this option.

`force` forces the calculation of standard errors and other inference-related quantities in situations where `predictnl` would otherwise refuse to do so. The calculation of standard errors takes place by evaluating (at  $\hat{\theta}$ ) the numerical derivative of  $g(\theta, \mathbf{x}_i)$  with respect to  $\theta$ . If `predictnl` detects that  $g()$  is possibly a function of random quantities other than  $\hat{\theta}$ , it will refuse to calculate standard errors or any other quantity derived from them. The `force` option forces the calculation to take place anyway. If you use the `force` option, there is no guarantee that any inference quantities (e.g., standard errors) will be correct or that the values obtained can be interpreted.

## Remarks

Remarks are presented under the following headings:

[Introduction](#)  
[Nonlinear transformations and standard errors](#)  
[Using `xb\(\)` and `predict\(\)`](#)  
[Multiple-equation \(ME\) estimators](#)  
[Test statistics and significance levels](#)  
[Manipulability](#)  
[Confidence intervals](#)

## Introduction

`predictnl` and `nlcom` are Stata's delta method commands—they take a nonlinear transformation of the estimated parameter vector from some fitted model and apply the delta method to calculate the variance, standard error, Wald test statistic, etc., of this transformation. `nlcom` is designed for scalar functions of the parameters, and `predictnl` is designed for functions of the parameters and of the data, that is, for predictions.

## Nonlinear transformations and standard errors

We begin by fitting a probit model to the low-birthweight data of [Hosmer Jr. and Lemeshow \(2000, 25\)](#). The data are described in detail in [\[R\] logistic](#).



```
. use http://www.stata-press.com/data/r11/lbw
(Hosmer & Lemeshow data)

. probit low lwt smoke ptl ht

Iteration 0: log likelihood = -117.336
Iteration 1: log likelihood = -106.75886
Iteration 2: log likelihood = -106.67852
Iteration 3: log likelihood = -106.67851

Probit regression Number of obs = 189
 LR chi2(4) = 21.31
 Prob > chi2 = 0.0003
Log likelihood = -106.67851 Pseudo R2 = 0.0908
```

| low   | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-------|-----------|-----------|-------|-------|----------------------|-----------|
| lwt   | -.0095164 | .0036875  | -2.58 | 0.010 | -.0167438            | -.0022891 |
| smoke | .3487004  | .2041772  | 1.71  | 0.088 | -.0514794            | .7488803  |
| ptl   | .365667   | .1921201  | 1.90  | 0.057 | -.0108815            | .7422154  |
| ht    | 1.082355  | .410673   | 2.64  | 0.008 | .2774503             | 1.887259  |
| _cons | .4238985  | .4823224  | 0.88  | 0.379 | -.5214361            | 1.369233  |

After we fit such a model, we first would want to generate the predicted probabilities of a low birthweight, given the covariate values in the estimation sample. This is easily done using `predict` after `probit`, but it doesn't answer the question, "What are the standard errors of those predictions?"

For the time being, we will consider ourselves ignorant of any automated way to obtain the predicted probabilities after `probit`. The formula for the prediction is

$$\Pr(y \neq 0 | \mathbf{x}_i) = \Phi(\mathbf{x}_i \boldsymbol{\beta})$$

where  $\Phi$  is the standard cumulative normal. Thus for this example,  $g(\boldsymbol{\theta}, \mathbf{x}_i) = \Phi(\mathbf{x}_i \boldsymbol{\beta})$ . Armed with the formula, we can use `predictnl` to generate the predictions and their standard errors:

```
. predictnl phat = normal(_b[_cons] + _b[ht]*ht + _b[ptl]*ptl +
> _b[smoke]*smoke + _b[lwt]*lwt), se(phat_se)
. list phat phat_se lwt smoke ptl ht in -10/1
```

|      | phat     | phat_se  | lwt | smoke | ptl | ht |
|------|----------|----------|-----|-------|-----|----|
| 180. | .2363556 | .042707  | 120 | 0     | 0   | 0  |
| 181. | .6577712 | .1580714 | 154 | 0     | 1   | 1  |
| 182. | .2793261 | .0519958 | 106 | 0     | 0   | 0  |
| 183. | .1502118 | .0676338 | 190 | 1     | 0   | 0  |
| 184. | .5702871 | .0819911 | 101 | 1     | 1   | 0  |
| 185. | .4477045 | .079889  | 95  | 1     | 0   | 0  |
| 186. | .2988379 | .0576306 | 100 | 0     | 0   | 0  |
| 187. | .4514706 | .080815  | 94  | 1     | 0   | 0  |
| 188. | .5615571 | .1551051 | 142 | 0     | 0   | 1  |
| 189. | .7316517 | .1361469 | 130 | 1     | 0   | 1  |

Thus subject 180 in our data has an estimated probability of low birthweight of 23.6% with standard error 4.3%.

Used without options, `predictnl` is not much different from `generate`. By specifying the `se(phat_se)` option, we were able to obtain a variable containing the standard errors of the predictions; therein lies the utility of `predictnl`.

Using `xb()` and `predict()`

As was the case above, a prediction is often not a function of a few isolated parameters and their corresponding variables but instead is some (possibly elaborate) function of the entire linear predictor. For models with many predictors, the brute-force expression for the linear predictor can be cumbersome to type. An alternative is to use the inline function `xb()`. `xb()` is a shortcut for having to type `_b[_cons] + _b[ht]*ht + _b[ptl]*ptl + ...`,

```
. drop phat phat_se
. predictnl phat = normal(xb()), se(phat_se)
. list phat phat_se lwt smoke ptl ht in -10/1
```

|      | phat     | phat_se  | lwt | smoke | ptl | ht |
|------|----------|----------|-----|-------|-----|----|
| 180. | .2363556 | .042707  | 120 | 0     | 0   | 0  |
| 181. | .6577712 | .1580714 | 154 | 0     | 1   | 1  |
| 182. | .2793261 | .0519958 | 106 | 0     | 0   | 0  |
| 183. | .1502118 | .0676338 | 190 | 1     | 0   | 0  |
| 184. | .5702871 | .0819911 | 101 | 1     | 1   | 0  |
| 185. | .4477045 | .079889  | 95  | 1     | 0   | 0  |
| 186. | .2988379 | .0576306 | 100 | 0     | 0   | 0  |
| 187. | .4514706 | .080815  | 94  | 1     | 0   | 0  |
| 188. | .5615571 | .1551051 | 142 | 0     | 0   | 1  |
| 189. | .7316517 | .1361469 | 130 | 1     | 0   | 1  |

which yields the same results. This approach is easier, produces more readable code, and is less prone to error, such as forgetting to include a term in the sum.

Here we used `xb()` without arguments because we have only one equation in our model. In multiple-equation (ME) settings, `xb()` (or equivalently `xb(#1)`) yields the linear predictor from the first equation, `xb(#2)` from the second, etc. You can also refer to equations by their names, e.g., `xb(income)`.

❑ Technical note

Most estimation commands in Stata allow the postestimation calculation of linear predictors and their standard errors via `predict`. For example, to obtain these for the first (or only) equation in the model, you could type

```
predict xbvar, xb
predict stdpvar, stdp
```

Equivalently, you could type

```
predictnl xbvar = xb(), se(stdpvar)
```

but we recommend the first method, as it is faster. As we demonstrated above, however, `predictnl` is more general.



Returning to our probit example, we can further simplify the calculation by using the inline function `predict()`. `predict(pred_options)` works by substituting, within our `predictnl` expression, the calculation performed by

```
predict ..., pred_options
```

In our example, we are interested in the predicted probabilities after a probit regression, normally obtained via

```
predict ..., p
```

We can obtain these predictions (and standard errors) by using

```
. drop phat phat_se
. predictnl phat = predict(p), se(phat_se)
. list phat phat_se lwt smoke ptl ht in -10/1
```

|      | phat     | phat_se  | lwt | smoke | ptl | ht |
|------|----------|----------|-----|-------|-----|----|
| 180. | .2363556 | .042707  | 120 | 0     | 0   | 0  |
| 181. | .6577712 | .1580714 | 154 | 0     | 1   | 1  |
| 182. | .2793261 | .0519958 | 106 | 0     | 0   | 0  |
| 183. | .1502118 | .0676338 | 190 | 1     | 0   | 0  |
| 184. | .5702871 | .0819911 | 101 | 1     | 1   | 0  |
| 185. | .4477045 | .079889  | 95  | 1     | 0   | 0  |
| 186. | .2988379 | .0576306 | 100 | 0     | 0   | 0  |
| 187. | .4514706 | .080815  | 94  | 1     | 0   | 0  |
| 188. | .5615571 | .1551051 | 142 | 0     | 0   | 1  |
| 189. | .7316517 | .1361469 | 130 | 1     | 0   | 1  |

which again replicates what we have already done by other means. However, this version did not require knowledge of the formula for the predicted probabilities after a probit regression—`predict(p)` took care of that for us.

Because the predicted probability is the default prediction after `probit`, we could have just used `predict()` without arguments, namely,

```
. predictnl phat = predict(), se(phat_se)
```

Also, the expression *pnl\_exp* can be inordinately complicated, with multiple calls to `predict()` and `xb()`. For example,

```
. predictnl phat = normal(invnormal(predict()) + predict(xb)/xb() - 1),
> se(phat_se)
```

is perfectly valid and will give the same result as before, albeit a bit inefficiently.

## □ Technical note

When using `predict()` and `xb()`, the *formula* for the calculation is substituted within *pnl\_exp*, not the values that result from the application of that formula. To see this, note the subtle difference between

```
. predict xbeta, xb
. predictnl phat = normal(xbeta), se(phat_se)
```

and

```
. predictnl phat = normal(xb()), se(phat_se)
```

Both sequences will yield the same `phat`, yet for the first sequence, `phat_se` will equal zero for all observations. The reason is that, once evaluated, `xbeta` will contain the values of the linear predictor, yet these values are treated as fixed and nonstochastic as far as `predictnl` is concerned. By contrast, because `xb()` is shorthand for the formula used to calculate the linear predictor, it contains not values, but references to the estimated regression coefficients and corresponding variables. Thus the second method produces the desired result. □

Multiple-equation (ME) estimators

In [R] `mlogit`, data on insurance choice (Tarlov et al. 1989; Wells et al. 1989) were examined, and a multinomial logit was used to assess the effects of age, gender, race, and site of study (one of three sites) on the type of insurance:

```
. use http://www.stata-press.com/data/r11/sysdsn1, clear
(Health insurance data)
. mlogit insure age male nonwhite i.site, nolog

Multinomial logistic regression Number of obs = 615
 LR chi2(10) = 42.99
 Prob > chi2 = 0.0000
Log likelihood = -534.36165 Pseudo R2 = 0.0387
```

| insure    | Coef.          | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-----------|----------------|-----------|-------|-------|----------------------|-----------|
| Indemnity | (base outcome) |           |       |       |                      |           |
| Prepaid   |                |           |       |       |                      |           |
| age       | -.011745       | .0061946  | -1.90 | 0.058 | -.0238862            | .0003962  |
| male      | .5616934       | .2027465  | 2.77  | 0.006 | .1643175             | .9590693  |
| nonwhite  | .9747768       | .2363213  | 4.12  | 0.000 | .5115955             | 1.437958  |
| site      |                |           |       |       |                      |           |
| 2         | .1130359       | .2101903  | 0.54  | 0.591 | -.2989296            | .5250013  |
| 3         | -.5879879      | .2279351  | -2.58 | 0.010 | -1.034733            | -.1412433 |
| _cons     | .2697127       | .3284422  | 0.82  | 0.412 | -.3740222            | .9134476  |
| Uninsure  |                |           |       |       |                      |           |
| age       | -.0077961      | .0114418  | -0.68 | 0.496 | -.0302217            | .0146294  |
| male      | .4518496       | .3674867  | 1.23  | 0.219 | -.268411             | 1.17211   |
| nonwhite  | .2170589       | .4256361  | 0.51  | 0.610 | -.6171725            | 1.05129   |
| site      |                |           |       |       |                      |           |
| 2         | -1.211563      | .4705127  | -2.57 | 0.010 | -2.133751            | -.2893747 |
| 3         | -.2078123      | .3662926  | -0.57 | 0.570 | -.9257327            | .510108   |
| _cons     | -1.286943      | .5923219  | -2.17 | 0.030 | -2.447872            | -.1260134 |

Of particular interest is the estimation of the relative risk, which, for a given selection, is the ratio of the probability of making that selection to the probability of selecting the base category (`Indemnity` here), given a set of covariate values. In a multinomial logit model, the relative risk (when comparing to the base category) simplifies to the exponentiated linear predictor for that selection.

Using this example, we can estimate the observation-specific relative risks of selecting a prepaid plan over the base category (with standard errors) by either referring to the `Prepaid` equation by name or number,

```
. predictnl RRppaid = exp(xb(Prepaid)), se(SERRppaid)
```

or

```
. predictnl RRppaid = exp(xb(#1)), se(SERRppaid)
```

because Prepaid is the first equation in the model.

Those of us for whom the simplified formula for the relative risk does not immediately come to mind may prefer to calculate the relative risk directly from its definition, that is, as a ratio of two predicted probabilities. After `mlogit`, the predicted probability for a category may be obtained using `predict`, but we must specify the category as the outcome:

```
. predictnl RRppaid = predict(outcome(Prepaid))/predict(outcome(Indemnity)),
> se(SERRppaid)
(1 missing value generated)
. list RRppaid SERRppaid age male nonwhite site in 1/10
```

|     | RRppaid  | SERRpp-d | age       | male | nonwhite | site |
|-----|----------|----------|-----------|------|----------|------|
| 1.  | .6168578 | .1503759 | 73.722107 | 0    | 0        | 2    |
| 2.  | 1.056658 | .1790703 | 27.89595  | 0    | 0        | 2    |
| 3.  | .8426442 | .1511281 | 37.541397 | 0    | 0        | 1    |
| 4.  | 1.460581 | .3671465 | 23.641327 | 0    | 1        | 3    |
| 5.  | .9115747 | .1324168 | 40.470901 | 0    | 0        | 2    |
| 6.  | 1.034701 | .1696923 | 29.683777 | 0    | 0        | 2    |
| 7.  | .9223664 | .1344981 | 39.468857 | 0    | 0        | 2    |
| 8.  | 1.678312 | .4216626 | 26.702255 | 1    | 0        | 1    |
| 9.  | .9188519 | .2256017 | 63.101974 | 0    | 1        | 3    |
| 10. | .5766296 | .1334877 | 69.839828 | 0    | 0        | 1    |

The “(1 missing value generated)” message is not an error; further examination of the data would reveal that **age** is missing in one observation and that the offending observation (among others) is not in the estimation sample. Just as with `predict`, `predictnl` can generate predictions in or out of the estimation sample.

Thus we estimate (among other things) that a white, female, 73-year-old from site 2 is less likely to choose a prepaid plan over an indemnity plan—her relative risk is about 62% with standard error 15%.

## Test statistics and significance levels

Often a standard error calculation is just a means to an end, and what is really desired is a test of the hypothesis,

$$H_0 : g(\theta, \mathbf{x}_i) = 0$$

versus the two-sided alternative.

We can use `predictnl` to obtain the Wald test statistics or significance levels (or both) for the above tests, whether or not we want standard errors. To obtain the Wald test statistics, we use the `wald()` option; for significance levels, we use `p()`.

Returning to our `mlogit` example, suppose that we wanted for each observation a test of whether the relative risk of choosing a prepaid plan over an indemnity plan is different from one. One way to do this would be to define  $g()$  to be the relative risk minus one and then test whether  $g()$  is different from zero.

```
. predictnl RRm1 = exp(xb(Prepaid)) - 1, wald(W_RRm1) p(sig_RRm1)
```

```
(1 missing value generated)
note: significance levels are with respect to the chi-squared(1) distribution.
. list RRM1 W_RRM1 sig_RRM1 age male nonwhite in 1/10
```

|     | RRm1      | W_RRM1   | sig_RRM1 | age       | male | nonwhite |
|-----|-----------|----------|----------|-----------|------|----------|
| 1.  | -.3831422 | 6.491778 | .0108375 | 73.722107 | 0    | 0        |
| 2.  | .0566578  | .100109  | .7516989 | 27.89595  | 0    | 0        |
| 3.  | -.1573559 | 1.084116 | .2977787 | 37.541397 | 0    | 0        |
| 4.  | .4605812  | 1.573743 | .2096643 | 23.641327 | 0    | 1        |
| 5.  | -.0884253 | .4459299 | .5042742 | 40.470901 | 0    | 0        |
| 6.  | .0347015  | .0418188 | .8379655 | 29.683777 | 0    | 0        |
| 7.  | -.0776336 | .3331707 | .563798  | 39.468857 | 0    | 0        |
| 8.  | .6783119  | 2.587788 | .1076906 | 26.702255 | 1    | 0        |
| 9.  | -.0811482 | .1293816 | .719074  | 63.101974 | 0    | 1        |
| 10. | -.4233705 | 10.05909 | .001516  | 69.839828 | 0    | 0        |

The newly created variable `W_RRM1` contains the Wald test statistic for each observation, and `sig_RRM1` contains the level of significance. Thus our 73-year-old white female represented by the first observation would have a relative risk of choosing prepaid over indemnity that is significantly different from 1, at least at the 5% level. For this test, it was not necessary to generate a variable containing the standard error of the relative risk minus 1, but we could have done so had we wanted. We could have also omitted specifying `wald(W_RRM1)` if all we cared about were, say, the significance levels of the tests.

In this regard, `predictnl` acts as an observation-specific version of `testnl`, with the test results vectorized over the observations in the data. The significance levels are pointwise—they are not adjusted to reflect any simultaneous testing over the observations in the data.

## Manipulability

There are many ways to specify  $g(\boldsymbol{\theta}, \mathbf{x}_i)$  to yield tests such that, for multiple specifications of  $g()$ , the theoretical conditions for which

$$H_0: g(\boldsymbol{\theta}, \mathbf{x}_i) = 0$$

is true will be equivalent. However, this does not mean that the tests themselves will be equivalent. This is known as the manipulability of the Wald test for nonlinear hypotheses; also see [R] [boxcox](#).

As an example, consider the previous section where we defined  $g()$  to be the relative risk between choosing a prepaid plan over an indemnity plan, minus 1. We could also have defined  $g()$  to be the risk difference—the probability of choosing a prepaid plan minus the probability of choosing an indemnity plan. Either specification of  $g()$  yields a mathematically equivalent specification of  $H_0: g() = 0$ ; that is, the risk difference will equal zero when the relative risk equals one. However, the tests themselves do not give the same results:

```
. predictnl RD = predict(outcome(Prepaid)) - predict(outcome(Indemnity)),
> wald(W_RD) p(sig_RD)
(1 missing value generated)
note: significance levels are with respect to the chi-squared(1) distribution.
. list RD W_RD sig_RD RRm1 W_RRm1 sig_RRm1 in 1/10
```

|     | RD        | W_RD     | sig_RD   | RRm1      | W_RRm1   | sig_RRm1 |
|-----|-----------|----------|----------|-----------|----------|----------|
| 1.  | -.2303744 | 4.230243 | .0397097 | -.3831422 | 6.491778 | .0108375 |
| 2.  | .0266902  | .1058542 | .7449144 | .0566578  | .100109  | .7516989 |
| 3.  | -.0768078 | .9187646 | .3377995 | -.1573559 | 1.084116 | .2977787 |
| 4.  | .1710702  | 2.366535 | .1239619 | .4605812  | 1.573743 | .2096643 |
| 5.  | -.0448509 | .4072922 | .5233471 | -.0884253 | .4459299 | .5042742 |
| 6.  | .0165251  | .0432816 | .835196  | .0347015  | .0418188 | .8379655 |
| 7.  | -.0391535 | .3077611 | .5790573 | -.0776336 | .3331707 | .563798  |
| 8.  | .22382    | 4.539085 | .0331293 | .6783119  | 2.587788 | .1076906 |
| 9.  | -.0388409 | .1190183 | .7301016 | -.0811482 | .1293816 | .719074  |
| 10. | -.2437626 | 6.151558 | .0131296 | -.4233705 | 10.05909 | .001516  |

In certain cases (such as subject 8), the difference can be severe enough to potentially change the conclusion. The reason for this inconsistency is that the nonlinear Wald test is actually a standard Wald test of a first-order Taylor approximation of  $g()$ , and this approximation can differ according to how  $g()$  is specified.

As such, keep in mind the manipulability of nonlinear Wald tests when drawing scientific conclusions.

## Confidence intervals

We can also use `predictnl` to obtain confidence intervals for the observation-specific  $g(\theta, \mathbf{x}_i)$  by using the `ci()` option to specify two new variables to contain the left and right endpoints of the confidence interval, respectively. For example, we could generate confidence intervals for the risk differences calculated previously:

```
. drop RD
. predictnl RD = predict(outcome(Prepaid)) - predict(outcome(Indemnity)),
> ci(RD_lcl RD_rc1)
(1 missing value generated)
note: Confidence intervals calculated using Z critical values.
. list RD RD_lcl RD_rc1 age male nonwhite in 1/10
```

|     | RD        | RD_lcl    | RD_rc1    | age       | male | nonwhite |
|-----|-----------|-----------|-----------|-----------|------|----------|
| 1.  | -.2303744 | -.4499073 | -.0108415 | 73.722107 | 0    | 0        |
| 2.  | .0266902  | -.1340948 | .1874752  | 27.89595  | 0    | 0        |
| 3.  | -.0768078 | -.2338625 | .080247   | 37.541397 | 0    | 0        |
| 4.  | .1710702  | -.0468844 | .3890248  | 23.641327 | 0    | 1        |
| 5.  | -.0448509 | -.1825929 | .092891   | 40.470901 | 0    | 0        |
| 6.  | .0165251  | -.1391577 | .1722078  | 29.683777 | 0    | 0        |
| 7.  | -.0391535 | -.177482  | .099175   | 39.468857 | 0    | 0        |
| 8.  | .22382    | .0179169  | .4297231  | 26.702255 | 1    | 0        |
| 9.  | -.0388409 | -.2595044 | .1818226  | 63.101974 | 0    | 1        |
| 10. | -.2437626 | -.4363919 | -.0511332 | 69.839828 | 0    | 0        |

The confidence level, here, 95%, is either set using the `level()` option or obtained from the current default level, `c(level)`; see [\[U\] 20.7 Specifying the width of confidence intervals](#).

From the above output, we can see that, for subjects 1, 8, and 10, a 95% confidence interval for the risk difference does not contain zero, meaning that, for these subjects, there is some evidence of a significant difference in risks.

The confidence intervals calculated by `predictnl` are pointwise; there is no adjustment (such as a Bonferroni correction) made so that these confidence intervals may be considered jointly at the specified level.

## Methods and formulas

`predictnl` is implemented as an ado-file.

For the  $i$ th observation, consider the transformation  $g(\boldsymbol{\theta}, \mathbf{x}_i)$ , estimated by  $g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i)$ , for the  $1 \times k$  parameter vector  $\boldsymbol{\theta}$  and data  $\mathbf{x}_i$  ( $\mathbf{x}_i$  is assumed fixed). The variance of  $g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i)$  is estimated by

$$\widehat{\text{Var}} \left\{ g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i) \right\} = \mathbf{G} \mathbf{V} \mathbf{G}'$$

where  $\mathbf{G}$  is the vector of derivatives

$$\mathbf{G} = \left\{ \left. \frac{\partial g(\boldsymbol{\theta}, \mathbf{x}_i)}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}} \right\}_{(1 \times k)}$$

and  $\mathbf{V}$  is the estimated variance–covariance matrix of  $\widehat{\boldsymbol{\theta}}$ . Standard errors,  $\widehat{\text{se}}\{g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i)\}$ , are obtained as the square roots of the variances.

The Wald test statistic for testing

$$H_0: g(\boldsymbol{\theta}, \mathbf{x}_i) = 0$$

versus the two-sided alternative is given by

$$W_i = \frac{\left\{ g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i) \right\}^2}{\widehat{\text{Var}} \left\{ g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i) \right\}}$$

When the variance–covariance matrix of  $\widehat{\boldsymbol{\theta}}$  is an asymptotic covariance matrix,  $W_i$  is approximately distributed as  $\chi^2$  with 1 degree of freedom. For linear regression,  $W_i$  is taken to be approximately distributed as  $F_{1,r}$ , where  $r$  is the residual degrees of freedom from the original model fit. The levels of significance of the observation-by-observation tests of  $H_0$  versus the two-sided alternative are given by

$$p_i = \Pr(T > W_i)$$

where  $T$  is either a  $\chi^2$ - or  $F$ -distributed random variable, as described above.

A  $(1 - \alpha) \times 100\%$  confidence interval for  $g(\boldsymbol{\theta}, \mathbf{x}_i)$  is given by

$$g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i) \pm z_{\alpha/2} \left[ \widehat{\text{se}} \left\{ g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i) \right\} \right]$$

when  $W_i$  is  $\chi^2$ -distributed, and

$$g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i) \pm t_{\alpha/2,r} \left[ \widehat{\text{se}} \left\{ g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i) \right\} \right]$$



when  $W_i$  is  $F$ -distributed.  $z_p$  is the  $1 - p$  quantile of the standard normal distribution, and  $t_{p,r}$  is the  $1 - p$  quantile of the  $t$  distribution with  $r$  degrees of freedom.

## References

- Gould, W. W. 1996. [crc43: Wald test of nonlinear hypotheses after model estimation](#). *Stata Technical Bulletin* 29: 2–4. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 15–18. College Station, TX: Stata Press.
- Hosmer Jr., D. W., and S. Lemeshow. 2000. *Applied Logistic Regression*. 2nd ed. New York: Wiley.
- Phillips, P. C. B., and J. Y. Park. 1988. On the formulation of Wald tests of nonlinear restrictions. *Econometrica* 56: 1065–1083.
- Tarlov, A. R., J. E. Ware Jr., S. Greenfield, E. C. Nelson, E. Perrin, and M. Zubkoff. 1989. The medical outcomes study. An application of methods for monitoring the results of medical care. *Journal of the American Medical Association* 262: 925–930.
- Wells, K. B., R. D. Hays, M. A. Burnam, W. H. Rogers, S. Greenfield, and J. E. Ware Jr. 1989. Detection of depressive disorder for patients receiving prepaid or fee-for-service care. Results from the Medical Outcomes Survey. *Journal of the American Medical Association* 262: 3298–3302.

## Also see

- [R] [lincom](#) — Linear combinations of estimators
- [R] [nlcom](#) — Nonlinear combinations of estimators
- [R] [predict](#) — Obtain predictions, residuals, etc., after estimation
- [R] [test](#) — Test linear hypotheses after estimation
- [R] [testnl](#) — Test nonlinear hypotheses after estimation
- [U] [20 Estimation and postestimation commands](#)

Syntax

```
probit depvar [indepvars] [if] [in] [weight] [, options]
```

| <i>options</i>                            | description                                                                                                                |
|-------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| Model                                     |                                                                                                                            |
| <u>noconstant</u>                         | suppress constant term                                                                                                     |
| <u>offset</u> ( <i>varname</i> )          | include <i>varname</i> in model with coefficient constrained to 1                                                          |
| <u>asis</u>                               | retain perfect predictor variables                                                                                         |
| <u>constraints</u> ( <i>constraints</i> ) | apply specified linear constraints                                                                                         |
| <u>collinear</u>                          | keep collinear variables                                                                                                   |
| SE/Robust                                 |                                                                                                                            |
| <u>vce</u> ( <i>vcetype</i> )             | <i>vcetype</i> may be <u>oim</u> , <u>robust</u> , <u>cluster</u> <i>clustvar</i> , <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting                                 |                                                                                                                            |
| <u>level</u> (#)                          | set confidence level; default is <u>level</u> (95)                                                                         |
| <u>nocnsreport</u>                        | do not display constraints                                                                                                 |
| <u>display_options</u>                    | control spacing and display of omitted variables and base and empty cells                                                  |
| Maximization                              |                                                                                                                            |
| <u>maximize_options</u>                   | control the maximization process; seldom used                                                                              |
| <sup>†</sup> <u>nocoef</u>                | do not display the coefficient table; seldom used                                                                          |
| <sup>†</sup> <u>coeflegend</u>            | display coefficients' legend instead of coefficient table                                                                  |

<sup>†</sup> nocoef and coeflegend do not appear in the dialog box.

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

bootstrap, by, fracpoly, jackknife, mfp, mi estimate, nestreg, rolling, statsby, stepwise, and svy are allowed; see [U] 11.1.10 Prefix commands.

vce(bootstrap) and vce(jackknife) are not allowed with the mi estimate prefix.

Weights are not allowed with the bootstrap prefix.

aweights are not allowed with the jackknife prefix.

vce(), nocoef, and weights are not allowed with the svy prefix.

fweights, iweights, and pweights are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Binary outcomes > Probit regression

## Description

`probit` fits a maximum-likelihood probit model.

If estimating on grouped data, see the `bprobit` command described in [R] [glogit](#).

Several auxiliary commands may be run after `probit`, `logit`, or `logistic`; see [R] [logistic postestimation](#) for a description of these commands.

See [R] [logistic](#) for a list of related estimation commands.

## Options

### Model

`noconstant`, `offset(varname)`, `constraints(constraints)`, `collinear`; see [R] [estimation options](#).

`asis` specifies that all specified variables and observations be retained in the maximization process. This option is typically not specified and may introduce numerical instability. Normally `probit` drops variables that perfectly predict success or failure in the dependent variable along with their associated observations. In those cases, the effective coefficient on the dropped variables is infinity (negative infinity) for variables that completely determine a success (failure). Dropping the variable and perfectly predicted observations has no effect on the likelihood or estimates of the remaining coefficients and increases the numerical stability of the optimization process. Specifying this option forces retention of perfect predictor variables and their associated observations.

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

### Reporting

`level(#)`; see [R] [estimation options](#).

`nocnsreport`; see [R] [estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [R] [maximize](#). These options are seldom used.

The following options are available with `probit` but are not shown in the dialog box:

`nocoef` specifies that the coefficient table not be displayed. This option is sometimes used by programmers but is of no use interactively.

`coeflegend`; see [R] [estimation options](#).

## Remarks

Remarks are presented under the following headings:

Robust standard errors  
Model identification

probit fits maximum likelihood models with dichotomous dependent (left-hand-side) variables coded as 0/1 (more precisely, coded as 0 and not 0).

► Example 1

We have data on the make, weight, and mileage rating of 22 foreign and 52 domestic automobiles. We wish to fit a probit model explaining whether a car is foreign based on its weight and mileage. Here is an overview of our data:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. keep make mpg weight foreign

. describe

Contains data from http://www.stata-press.com/data/r11/auto.dta
 obs: 74 1978 Automobile Data
 vars: 4 13 Apr 2009 17:45
 size: 1,998 (99.9% of memory free) (_dta has notes)
```

| variable name | storage type | display format | value label | variable label |
|---------------|--------------|----------------|-------------|----------------|
| make          | str18        | %-18s          |             | Make and Model |
| mpg           | int          | %8.0g          |             | Mileage (mpg)  |
| weight        | int          | %8.0gc         |             | Weight (lbs.)  |
| foreign       | byte         | %8.0g          | origin      | Car type       |

```
Sorted by: foreign
 Note: dataset has changed since last saved
```

```
. inspect foreign
foreign: Car type
```

| foreign: Car type |          | Number of Observations |          |             |
|-------------------|----------|------------------------|----------|-------------|
|                   |          | Total                  | Integers | Nonintegers |
| #                 | Negative | -                      | -        | -           |
| #                 | Zero     | 52                     | 52       | -           |
| #                 | Positive | 22                     | 22       | -           |
| #                 |          |                        |          |             |
| #                 | Total    | 74                     | 74       | -           |
| #                 | Missing  | -                      |          |             |
|                   |          | 74                     |          |             |

foreign is labeled and all values are documented in the label.

The **foreign** variable takes on two unique values, 0 and 1. The value 0 denotes a domestic car, and 1 denotes a foreign car.

The model that we wish to fit is

$$\Pr(\text{foreign} = 1) = \Phi(\beta_0 + \beta_1 \text{weight} + \beta_2 \text{mpg})$$

where  $\Phi$  is the cumulative normal distribution.

To fit this model, we type

```
. probit foreign weight mpg
Iteration 0: log likelihood = -45.03321
Iteration 1: log likelihood = -27.914626
(output omitted)
Iteration 5: log likelihood = -26.844189
Probit regression
Log likelihood = -26.844189
```

Number of obs = 74
LR chi2(2) = 36.38
Prob > chi2 = 0.0000
Pseudo R2 = 0.4039

| foreign | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|---------|-----------|-----------|-------|-------|----------------------|-----------|
| weight  | -.0023355 | .0005661  | -4.13 | 0.000 | -.003445             | -.0012261 |
| mpg     | -.1039503 | .0515689  | -2.02 | 0.044 | -.2050235            | -.0028772 |
| _cons   | 8.275464  | 2.554142  | 3.24  | 0.001 | 3.269437             | 13.28149  |

We find that heavier cars are less likely to be foreign and that cars yielding better gas mileage are also less likely to be foreign, at least holding the weight of the car constant.

See [R] [maximize](#) for an explanation of the output.

❑ Technical note

Stata interprets a value of 0 as a negative outcome (failure) and treats all other values (except missing) as positive outcomes (successes). Thus if your dependent variable takes on the values 0 and 1, then 0 is interpreted as failure and 1 as success. If your dependent variable takes on the values 0, 1, and 2, then 0 is still interpreted as failure, but both 1 and 2 are treated as successes.

If you prefer a more formal mathematical statement, when you type `probit y x`, Stata fits the model

$$\Pr(y_j \neq 0 \mid \mathbf{x}_j) = \Phi(\mathbf{x}_j\boldsymbol{\beta})$$

where  $\Phi$  is the standard cumulative normal.

Robust standard errors

If you specify the `vce(robust)` option, probit reports robust standard errors; see [U] [20.16 Obtaining robust variance estimates](#).

➤ Example 2

For the model from example 1, the robust calculation increases the standard error of the coefficient on `mpg` by almost 15%:

```
. probit foreign weight mpg, vce(robust) nolog
Probit regression
Log pseudolikelihood = -26.844189
```

Number of obs = 74
Wald chi2(2) = 30.26
Prob > chi2 = 0.0000
Pseudo R2 = 0.4039

| foreign | Coef.     | Robust Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|---------|-----------|------------------|-------|-------|----------------------|-----------|
| weight  | -.0023355 | .0004934         | -4.73 | 0.000 | -.0033025            | -.0013686 |
| mpg     | -.1039503 | .0593548         | -1.75 | 0.080 | -.2202836            | .0123829  |
| _cons   | 8.275464  | 2.539176         | 3.26  | 0.001 | 3.29877              | 13.25216  |

Without `vce(robust)`, the standard error for the coefficient on `mpg` was reported to be 0.052 with a resulting confidence interval of  $[-0.21, -0.00]$ .



➤ Example 3

The `vce(cluster clustvar)` option can relax the independence assumption required by the probit estimator to independence between clusters. To demonstrate, we will switch to a different dataset.

We are studying unionization of women in the United States and have a dataset with 26,200 observations on 4,434 women between 1970 and 1988. We will use the variables `age` (the women were 14–26 in 1968, and our data span the age range of 16–46), `grade` (years of schooling completed, ranging from 0 to 18), `not_smsa` (28% of the person-time was spent living outside an SMSA—standard metropolitan statistical area), `south` (41% of the person-time was in the South), and `year`. Each of these variables is included in the regression as a covariate along with the interaction between `south` and `year`. This interaction, along with the `south` and `year` variables, is specified in the `probit` command using factor-variables notation, `south##c.year`. We also have variable `union`, indicating union membership. Overall, 22% of the person-time is marked as time under union membership, and 44% of these women have belonged to a union.

We fit the following model, ignoring that the women are observed an average of 5.9 times each in these data:

```
. use http://www.stata-press.com/data/r11/union, clear
(NLS Women 14-24 in 1968)

. probit union age grade not_smsa south##c.year

Iteration 0: log likelihood = -13864.23
Iteration 1: log likelihood = -13545.541
Iteration 2: log likelihood = -13544.385
Iteration 3: log likelihood = -13544.385

Probit regression Number of obs = 26200
 LR chi2(6) = 639.69
 Prob > chi2 = 0.0000
Log likelihood = -13544.385 Pseudo R2 = 0.0231
```

| union        | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|--------------|-----------|-----------|-------|-------|----------------------|-----------|
| age          | .0118481  | .0029072  | 4.08  | 0.000 | .0061502             | .017546   |
| grade        | .0267365  | .0036689  | 7.29  | 0.000 | .0195457             | .0339273  |
| not_smsa     | -.1293525 | .0202595  | -6.38 | 0.000 | -.1690604            | -.0896445 |
| 1.south      | -.8281077 | .2472219  | -3.35 | 0.001 | -1.312654            | -.3435618 |
| year         | -.0080931 | .0033469  | -2.42 | 0.016 | -.0146529            | -.0015333 |
| south#c.year |           |           |       |       |                      |           |
| 1            | .0057369  | .0030917  | 1.86  | 0.064 | -.0003226            | .0117965  |
| _cons        | -.6542487 | .2007777  | -3.26 | 0.001 | -1.047766            | -.2607316 |

The reported standard errors in this model are probably meaningless. Women are observed repeatedly, and so the observations are not independent. Looking at the coefficients, we find a large southern effect against unionization and a time trend for the south that is almost significantly different from the overall downward trend. The `vce(cluster clustvar)` option provides a way to fit this model and obtains correct standard errors:

```
. probit union age grade not_smsa south##c.year, vce(cluster id)
Iteration 0: log pseudolikelihood = -13864.23
Iteration 1: log pseudolikelihood = -13545.541
Iteration 2: log pseudolikelihood = -13544.385
Iteration 3: log pseudolikelihood = -13544.385

Probit regression Number of obs = 26200
 Wald chi2(6) = 166.53
 Prob > chi2 = 0.0000
Log pseudolikelihood = -13544.385 Pseudo R2 = 0.0231
 (Std. Err. adjusted for 4434 clusters in idcode)
```

| union        | Coef.     | Robust<br>Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|--------------|-----------|---------------------|-------|-------|----------------------|-----------|
| age          | .0118481  | .0056625            | 2.09  | 0.036 | .0007499             | .0229463  |
| grade        | .0267365  | .0078124            | 3.42  | 0.001 | .0114244             | .0420486  |
| not_smsa     | -.1293525 | .0403885            | -3.20 | 0.001 | -.2085125            | -.0501925 |
| 1.south      | -.8281077 | .3201584            | -2.59 | 0.010 | -1.455607            | -.2006089 |
| year         | -.0080931 | .0060829            | -1.33 | 0.183 | -.0200153            | .0038292  |
| south#c.year |           |                     |       |       |                      |           |
| 1            | .0057369  | .0040133            | 1.43  | 0.153 | -.002129             | .0136029  |
| _cons        | -.6542487 | .3485976            | -1.88 | 0.061 | -1.337487            | .02899    |

These standard errors are larger than those reported by the inappropriate conventional calculation. By comparison, another model we could fit is an equal-correlation population-averaged probit model:

```
. xtprobit union age grade not_smsa south##c.year, pa
Iteration 1: tolerance = .12544249
Iteration 2: tolerance = .0034686
Iteration 3: tolerance = .00017448
Iteration 4: tolerance = 8.382e-06
Iteration 5: tolerance = 3.997e-07

GEE population-averaged model
Group variable: idcode Number of obs = 26200
Link: probit Number of groups = 4434
Family: binomial Obs per group: min = 1
Correlation: exchangeable avg = 5.9
 max = 12
 Wald chi2(6) = 242.57
Scale parameter: 1 Prob > chi2 = 0.0000
```

| union        | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|--------------|-----------|-----------|-------|-------|----------------------|-----------|
| age          | .0089699  | .0053208  | 1.69  | 0.092 | -.0014586            | .0193985  |
| grade        | .0333174  | .0062352  | 5.34  | 0.000 | .0210966             | .0455382  |
| not_smsa     | -.0715717 | .027543   | -2.60 | 0.009 | -.1255551            | -.0175884 |
| 1.south      | -1.017368 | .207931   | -4.89 | 0.000 | -1.424905            | -.6098308 |
| year         | -.0062708 | .0055314  | -1.13 | 0.257 | -.0171122            | .0045706  |
| south#c.year |           |           |       |       |                      |           |
| 1            | .0086294  | .00258    | 3.34  | 0.001 | .0035727             | .013686   |
| _cons        | -.8670997 | .294771   | -2.94 | 0.003 | -1.44484             | -.2893592 |

The coefficient estimates are similar, but these standard errors are smaller than those produced by `probit`, `vce(cluster clustvar)`, as we would expect. If the equal-correlation assumption is valid, the population-averaged probit estimator above should be more efficient.

Is the assumption valid? That is a difficult question to answer. The default population-averaged estimates correspond to an assumption of exchangeable correlation within person. It would not be unreasonable to assume an AR(1) correlation within person or to assume that the observations are correlated but that we do not wish to impose any structure. See [\[XT\] xtprobit](#) and [\[XT\] xtgee](#) for full details.



`probit, vce(cluster clustvar)` is robust to assumptions about within-cluster correlation. That is, it inefficiently sums within cluster for the standard error calculation rather than attempting to exploit what might be assumed about the within-cluster correlation.

### Model identification

The `probit` command has one more feature that is probably the most useful. It will automatically check the model for identification and, if the model is underidentified, drop whatever variables and observations are necessary for estimation to proceed.

#### ➤ Example 4

Have you ever fit a probit model where one or more of your independent variables perfectly predicted one or the other outcome?

For instance, consider the following data:

| Outcome <i>y</i> | Independent variable <i>x</i> |
|------------------|-------------------------------|
| 0                | 1                             |
| 0                | 1                             |
| 0                | 0                             |
| 1                | 0                             |

Say that we wish to predict the outcome on the basis of the independent variable. The outcome is always zero when the independent variable is one. In our data,  $\Pr(y = 0 \mid x = 1) = 1$ , which means that the probit coefficient on  $x$  must be minus infinity with a corresponding infinite standard error. At this point, you may suspect that we have a problem.

Unfortunately, not all such problems are so easily detected, especially if you have many independent variables in your model. If you have ever had such difficulties, then you have experienced one of the more unpleasant aspects of computer optimization. The computer has no idea that it is trying to solve for an infinite coefficient as it begins its iterative process. All it knows is that, at each step, making the coefficient a little bigger, or a little smaller, works wonders. It continues on its merry way until either 1) the whole thing comes crashing to the ground when a numerical overflow error occurs or 2) it reaches some predetermined cutoff that stops the process. Meanwhile, you have been waiting. And the estimates that you finally receive, if any, may be nothing more than numerical roundoff.

Stata watches for these sorts of problems, alerts you, fixes them, and then properly fits the model.

Let's return to our automobile data. Among the variables we have in the data is one called `repair` that takes on three values. A value of 1 indicates that the car has a poor repair record, 2 indicates an average record, and 3 indicates a better-than-average record. Here is a tabulation of our data:



```
. use http://www.stata-press.com/data/r11/repair
(1978 Automobile Data)
. tabulate foreign repair
```

| Car type | repair |    |    | Total |
|----------|--------|----|----|-------|
|          | 1      | 2  | 3  |       |
| Domestic | 10     | 27 | 9  | 46    |
| Foreign  | 0      | 3  | 9  | 12    |
| Total    | 10     | 30 | 18 | 58    |

All the cars with poor repair records (`repair = 1`) are domestic. If we were to attempt to predict `foreign` on the basis of the repair records, the predicted probability for the `repair = 1` category would have to be zero. This in turn means that the probit coefficient must be minus infinity, and that would set most computer programs buzzing.

Let's try using Stata on this problem.

```
. probit foreign b3.repair
note: 1.repair != 0 predicts failure perfectly
 1.repair dropped and 10 obs not used

Iteration 0: log likelihood = -26.992087
Iteration 1: log likelihood = -22.276479
Iteration 2: log likelihood = -22.229184
Iteration 3: log likelihood = -22.229138
Iteration 4: log likelihood = -22.229138

Probit regression Number of obs = 48
 LR chi2(1) = 9.53
 Prob > chi2 = 0.0020
Log likelihood = -22.229138 Pseudo R2 = 0.1765
```

| foreign | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|---------|-----------|-----------|-------|-------|----------------------|-----------|
| repair  |           |           |       |       |                      |           |
| 1       | (empty)   |           |       |       |                      |           |
| 2       | -1.281552 | .4297326  | -2.98 | 0.003 | -2.123812            | -.4392911 |
| _cons   | 1.16e-16  | .295409   | 0.00  | 1.000 | -.578991             | .578991   |

Remember that all the cars with poor repair records (`repair = 1`) are domestic, so the model cannot be fit, or at least it cannot be fit if we restrict ourselves to finite coefficients. Stata noted that fact “note: 1.repair != 0 predicts failure perfectly”. This is Stata’s mathematically precise way of saying what we said in English. When `repair` is 1, the car is domestic.

Stata then went on to say, “1.repair dropped and 10 obs not used”. This is Stata eliminating the problem. First, 1.repair had to be removed from the model because it would have an infinite coefficient. Then the 10 observations that led to the problem had to be eliminated, as well, so as not to bias the remaining coefficients in the model. The 10 observations that are not used are the 10 domestic cars that have poor repair records.

Stata then fit what was left of the model, using the remaining observations. Because no observations remained for cars with poor repair records, Stata reports “(empty)” in the row for `repair = 1`.

## □ Technical note

Stata is pretty smart about catching these problems. It will catch “one-way causation by a dummy variable”, as we demonstrated above.

Stata also watches for “two-way causation”, that is, a variable that perfectly determines the outcome, both successes and failures. Here Stata says that the variable “predicts outcome perfectly” and stops. Statistics dictate that no model can be fit.

Stata also checks your data for collinear variables; it will say “so-and-so omitted because of collinearity”. No observations need to be eliminated here and model fitting will proceed without the offending variable.

It will also catch a subtle problem that can arise with continuous data. For instance, if we were estimating the chances of surviving the first year after an operation, and if we included in our model `age`, and if all the persons over 65 died within the year, Stata will say, “`age > 65` predicts failure perfectly”. It will then inform us about how it resolves the issue and fit what can be fit of our model.

`probit` (and `logit`, `logistic`, and `ivprobit`) will also occasionally fail to converge and then display messages such as

Note: 4 failures and 0 successes completely determined.

The cause of this message and what to do if you see it are described in [\[R\] `logit`](#).



## Saved results

`probit` saves the following in `e()`:

### Scalars

|                            |                                                |
|----------------------------|------------------------------------------------|
| <code>e(N)</code>          | number of observations                         |
| <code>e(N_cds)</code>      | number of completely determined successes      |
| <code>e(N_cdf)</code>      | number of completely determined failures       |
| <code>e(k)</code>          | number of parameters                           |
| <code>e(k_eq)</code>       | number of equations                            |
| <code>e(k_eq_model)</code> | number of equations in model Wald test         |
| <code>e(k_dv)</code>       | number of dependent variables                  |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints |
| <code>e(df_m)</code>       | model degrees of freedom                       |
| <code>e(r2_p)</code>       | pseudo- <i>R</i> -squared                      |
| <code>e(ll)</code>         | log likelihood                                 |
| <code>e(ll_0)</code>       | log likelihood, constant-only model            |
| <code>e(N_clust)</code>    | number of clusters                             |
| <code>e(chi2)</code>       | $\chi^2$                                       |
| <code>e(p)</code>          | significance of model test                     |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                      |
| <code>e(ic)</code>         | number of iterations                           |
| <code>e(rc)</code>         | return code                                    |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                    |

## Macros

|                                 |                                                                          |
|---------------------------------|--------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>probit</code>                                                      |
| <code>e(cmdline)</code>         | command as typed                                                         |
| <code>e(depvar)</code>          | name of dependent variable                                               |
| <code>e(wtype)</code>           | weight type                                                              |
| <code>e(wexp)</code>            | weight expression                                                        |
| <code>e(title)</code>           | title in estimation output                                               |
| <code>e(clustvar)</code>        | name of cluster variable                                                 |
| <code>e(offset)</code>          | offset                                                                   |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                  |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                           |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                            |
| <code>e(opt)</code>             | type of optimization                                                     |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>       | type of ml method                                                        |
| <code>e(user)</code>            | name of likelihood-evaluator program                                     |
| <code>e(technique)</code>       | maximization technique                                                   |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular              |
| <code>e(crittype)</code>        | optimization criterion                                                   |
| <code>e(properties)</code>      | b V                                                                      |
| <code>e(estat_cmd)</code>       | program used to implement <code>estat</code>                             |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                           |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>           |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>           |

## Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(gradient)</code>     | gradient vector                              |
| <code>e(mns)</code>          | vector of means of the independent variables |
| <code>e(rules)</code>        | information about perfect predictors         |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`probit` is implemented as an ado-file.

Probit analysis originated in connection with bioassay, and the word `probit`, a contraction of “probability unit”, was suggested by [Bliss \(1934a, 1934b\)](#). For an introduction to `probit` and `logit`, see, for example, [Aldrich and Nelson \(1984\)](#), [Cameron and Trivedi \(2009\)](#), [Greene \(2008\)](#), [Long \(1997\)](#), [Pampel \(2000\)](#), or [Powers and Xie \(2008\)](#). [Long and Freese \(2006, chap. 4\)](#) and [Jones \(2007, chap. 3\)](#) provide introductions to `probit` and `logit`, along with Stata examples.

The log-likelihood function for `probit` is

$$\ln L = \sum_{j \in S} w_j \ln \Phi(\mathbf{x}_j \boldsymbol{\beta}) + \sum_{j \notin S} w_j \ln \{1 - \Phi(\mathbf{x}_j \boldsymbol{\beta})\}$$

where  $\Phi$  is the cumulative normal and  $w_j$  denotes the optional weights.  $\ln L$  is maximized, as described in [R] [maximize](#).

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#). The scores are calculated as  $\mathbf{u}_j = \{\phi(\mathbf{x}_j\mathbf{b})/\Phi(\mathbf{x}_j\mathbf{b})\}\mathbf{x}_j$  for the positive outcomes and  $-\{\phi(\mathbf{x}_j\mathbf{b})/\{1 - \Phi(\mathbf{x}_j\mathbf{b})\}\}\mathbf{x}_j$  for the negative outcomes, where  $\phi$  is the normal density.

`probit` also supports estimation with survey data. For details on VCEs with survey data, see [SVY] [variance estimation](#).

Chester Ittner Bliss (1899–1979) was born in Ohio. He was educated as an entomologist, earning degrees from Ohio State and Columbia, and was employed by the United States Department of Agriculture until 1933. When he lost his job because of the Depression, Bliss then worked with R. A. Fisher in London and at the Institute of Plant Protection in Leningrad before returning to a post at the Connecticut Agricultural Experiment Station in 1938. He was also a lecturer at Yale for 25 years. Among many contributions to biostatistics, his development and application of probit methods to biological problems are outstanding.

## References

- Aldrich, J. H., and F. D. Nelson. 1984. *Linear Probability, Logit, and Probit Models*. Newbury Park, CA: Sage.
- Berkson, J. 1944. Application of the logistic function to bio-assay. *Journal of the American Statistical Association* 39: 357–365.
- Bliss, C. I. 1934a. The method of probits. *Science* 79: 38–39.
- . 1934b. The method of probits—a correction. *Science* 79: 409–410.
- Cameron, A. C., and P. K. Trivedi. 2009. *Microeconometrics Using Stata*. College Station, TX: Stata Press.
- Cochran, W. G., and D. J. Finney. 1979. Chester Ittner Bliss 1899–1979. *Biometrics* 35: 715–717.
- De Luca, G. 2008. SNP and SML estimation of univariate and bivariate binary-choice models. *Stata Journal* 8: 190–220.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- Hilbe, J. M. 1996. [sg54: Extended probit regression](#). *Stata Technical Bulletin* 32: 20–21. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 131–132. College Station, TX: Stata Press.
- Jones, A. 2007. *Applied Econometrics for Health Economists: A Practical Guide*. 2nd ed. Abingdon, UK: Radcliffe.
- Judge, G. G., W. E. Griffiths, R. C. Hill, H. Lütkepohl, and T.-C. Lee. 1985. *The Theory and Practice of Econometrics*. 2nd ed. New York: Wiley.
- Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.
- Long, J. S., and J. Freese. 2006. *Regression Models for Categorical Dependent Variables Using Stata*. 2nd ed. College Station, TX: Stata Press.
- Miranda, A., and S. Rabe-Hesketh. 2006. Maximum likelihood estimation of endogenous switching and sample selection models for binary, ordinal, and count variables. *Stata Journal* 6: 285–308.
- Pampel, F. C. 2000. *Logistic Regression: A Primer*. Thousand Oaks, CA: Sage.
- Powers, D. A., and Y. Xie. 2008. *Statistical Methods for Categorical Data Analysis*. 2nd ed. Bingley, UK: Emerald.
- Xu, J., and J. S. Long. 2005. Confidence intervals for predicted outcomes in regression models for categorical outcomes. *Stata Journal* 5: 537–559.

## Also see

- [R] **probit postestimation** — Postestimation tools for probit
- [R] **roc** — Receiver operating characteristic (ROC) analysis
- [R] **asmprobit** — Alternative-specific multinomial probit regression
- [R] **brier** — Brier score decomposition
- [R] **biprobit** — Bivariate probit regression
- [R] **glm** — Generalized linear models
- [R] **hetprob** — Heteroskedastic probit model
- [R] **ivprobit** — Probit model with continuous endogenous regressors
- [R] **logistic** — Logistic regression, reporting odds ratios
- [R] **logit** — Logistic regression, reporting coefficients
- [R] **mprobit** — Multinomial probit regression
- [R] **scobit** — Skewed logistic regression
- [SVY] **svy estimation** — Estimation commands for survey data
- [XT] **xtprobit** — Random-effects and population-averaged probit models
- [U] **20 Estimation and postestimation commands**

Description

The following postestimation commands are of special interest after `probit`:

| command                 | description                                                                                              |
|-------------------------|----------------------------------------------------------------------------------------------------------|
| <code>estat clas</code> | <code>estat classification</code> reports various summary statistics, including the classification table |
| <code>estat gof</code>  | Pearson or Hosmer–Lemeshow goodness-of-fit test                                                          |
| <code>lroc</code>       | graphs the ROC curve and calculates the area under the curve                                             |
| <code>lsens</code>      | graphs sensitivity and specificity versus probability cutoff                                             |

These commands are not appropriate after the `svy` prefix.  
For information about these commands, see [\[R\] logistic postestimation](#).

The following standard postestimation commands are also available:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>hausman</code>             | Hausman’s specification test                                                                        |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>            | link test for model specification                                                                   |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

## Syntax for predict

```
predict [type] newvar [if] [in] [, statistic nooffset rules asif]
```

| <i>statistic</i>        | description                                                                |
|-------------------------|----------------------------------------------------------------------------|
| Main                    |                                                                            |
| <code>pr</code>         | probability of a positive outcome; the default                             |
| <code>xb</code>         | linear prediction                                                          |
| <code>stdp</code>       | standard error of the linear prediction                                    |
| * <code>deviance</code> | deviance residual                                                          |
| <code>score</code>      | first derivative of the log likelihood with respect to $\mathbf{x}_j\beta$ |

Unstarred statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample. Starred statistics are calculated only for the estimation sample, even when `if e(sample)` is not specified.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

### Main

`pr`, the default, calculates the probability of a positive outcome.

`xb` calculates the linear prediction.

`stdp` calculates the standard error of the linear prediction.

`deviance` calculates the deviance residual.

`score` calculates the equation-level score,  $\partial \ln L / \partial (\mathbf{x}_j\beta)$ .

`nooffset` is relevant only if you specified `offset(varname)` for `probit`. It modifies the calculations made by `predict` so that they ignore the offset variable; the linear prediction is treated as  $\mathbf{x}_j\mathbf{b}$  rather than as  $\mathbf{x}_j\mathbf{b} + \text{offset}_j$ .

`rules` requests that Stata use any rules that were used to identify the model when making the prediction. By default, Stata calculates missing for excluded observations.

`asif` requests that Stata ignore the rules and exclusion criteria and calculate predictions for all observations possible using the estimated parameter from the model.

## Remarks

Remarks are presented under the following headings:

*Obtaining predicted values*  
*Performing hypothesis tests*

Obtaining predicted values

Once you have fit a probit model, you can obtain the predicted probabilities by using the `predict` command for both the estimation sample and other samples; see [U] 20 Estimation and postestimation commands and [R] `predict`. Here we will make only a few additional comments.

`predict` without arguments calculates the predicted probability of a positive outcome. With the `xb` option, `predict` calculates the linear combination  $x_jb$ , where  $x_j$  are the independent variables in the  $j$ th observation and  $b$  is the estimated parameter vector. This is known as the index function because the cumulative density indexed at this value is the probability of a positive outcome.

In both cases, Stata remembers any rules used to identify the model and calculates missing for excluded observations unless `rules` or `asif` is specified. This is covered in the following example.

With the `stdp` option, `predict` calculates the standard error of the prediction, which is *not* adjusted for replicated covariate patterns in the data.

You can calculate the unadjusted-for-replicated-covariate-patterns diagonal elements of the hat matrix, or leverage, by typing

```
. predict pred
. predict stdp, stdp
. generate hat = stdp^2*pred*(1-pred)
```

► Example 1

In example 4 of [R] `probit`, we fit the probit model `probit foreign b3.repair`. To obtain predicted probabilities, we type

```
. predict p
(option pr assumed; Pr(foreign))
(10 missing values generated)
. summarize foreign p
```

| Variable | Obs | Mean     | Std. Dev. | Min | Max |
|----------|-----|----------|-----------|-----|-----|
| foreign  | 58  | .2068966 | .4086186  | 0   | 1   |
| p        | 48  | .25      | .1956984  | .1  | .5  |

Stata remembers any rules used to identify the model and sets predictions to missing for any excluded observations. In the previous example, `probit` dropped the variable `1.repair` from our model and excluded 10 observations. When we typed `predict p`, those same 10 observations were again excluded and their predictions set to missing.

`predict`'s `rules` option uses the rules in the prediction. During estimation, we were told, "`1.repair` != 0 predicts failure perfectly", so the rule is that when `1.repair` is not zero, we should predict 0 probability of success or a positive outcome:

```
. predict p2, rules
. summarize foreign p p2
```

| Variable | Obs | Mean     | Std. Dev. | Min | Max |
|----------|-----|----------|-----------|-----|-----|
| foreign  | 58  | .2068966 | .4086186  | 0   | 1   |
| p        | 48  | .25      | .1956984  | .1  | .5  |
| p2       | 58  | .2068966 | .2016268  | 0   | .5  |

`predict`'s `asif` option ignores the rules and the exclusion criteria and calculates predictions for all observations possible using the estimated parameters from the model:



```
. predict p3, asif
. summarize for p p2 p3
```

| Variable | Obs | Mean     | Std. Dev. | Min | Max |
|----------|-----|----------|-----------|-----|-----|
| foreign  | 58  | .2068966 | .4086186  | 0   | 1   |
| p        | 48  | .25      | .1956984  | .1  | .5  |
| p2       | 58  | .2068966 | .2016268  | 0   | .5  |
| p3       | 58  | .2931034 | .2016268  | .1  | .5  |

Which is right? By default, `predict` uses the most conservative approach. If many observations had been excluded due to a simple rule, we could be reasonably certain that the `rules` prediction is correct. The `asif` prediction is correct only if the exclusion is a fluke and we would be willing to exclude the variable from the analysis, anyway. Then, however, we should refit the model to include the excluded observations.

◀

## Performing hypothesis tests

After estimation with `probit`, you can perform hypothesis tests by using the `test` or `testnl` command; see [\[U\] 20 Estimation and postestimation commands](#).

## Methods and formulas

All postestimation commands listed above are implemented as `ado`-files.

### predict after probit

Let index  $j$  be used to index observations, not covariate patterns. Define  $M_j$  for each observation as the total number of observations sharing  $j$ 's covariate pattern. Define  $Y_j$  as the total number of positive responses among observations sharing  $j$ 's covariate pattern. Define  $p_j$  as the predicted probability of a positive outcome for observation  $j$ .

For  $M_j > 1$ , the deviance residual  $d_j$  is defined as

$$d_j = \pm \left( 2 \left[ Y_j \ln \left( \frac{Y_j}{M_j p_j} \right) + (M_j - Y_j) \ln \left\{ \frac{M_j - Y_j}{M_j (1 - p_j)} \right\} \right] \right)^{1/2}$$

where the sign is the same as the sign of  $(Y_j - M_j p_j)$ . In the limiting cases, the deviance residual is given by

$$d_j = \begin{cases} -\sqrt{2M_j |\ln(1 - p_j)|} & \text{if } Y_j = 0 \\ \sqrt{2M_j |\ln p_j|} & \text{if } Y_j = M_j \end{cases}$$

## Also see

[\[R\] probit](#) — Probit regression

[\[R\] logistic postestimation](#) — Postestimation tools for logistic

[\[U\] 20 Estimation and postestimation commands](#)

Syntax

proportion *varlist* [*if*] [*in*] [*weight*] [, *options*]

| <i>options</i>                                    | description                                                                                                     |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| Model                                             |                                                                                                                 |
| <u>stdize</u> ( <i>varname</i> )                  | variable identifying strata for standardization                                                                 |
| <u>stdweight</u> ( <i>varname</i> )               | weight variable for standardization                                                                             |
| <u>nostdrescale</u>                               | do not rescale the standard weight variable                                                                     |
| <u>no</u> label                                   | suppress value labels from <i>varlist</i>                                                                       |
| <u>missing</u>                                    | treat missing values like other values                                                                          |
| if/in/over                                        |                                                                                                                 |
| <u>over</u> ( <i>varlist</i> [, <u>no</u> label]) | group over subpopulations defined by <i>varlist</i> ; optionally, suppress group labels                         |
| SE/Cluster                                        |                                                                                                                 |
| <u>vce</u> ( <i>vcetype</i> )                     | <i>vcetype</i> may be <u>analytic</u> , <u>cluster</u> <i>clustvar</i> , <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting                                         |                                                                                                                 |
| <u>level</u> (#)                                  | set confidence level; default is level(95)                                                                      |
| <u>no</u> header                                  | suppress table header                                                                                           |
| <u>no</u> legend                                  | suppress table legend                                                                                           |
| <sup>†</sup> <u>coeflegend</u>                    | display coefficients' legend instead of coefficient table                                                       |

<sup>†</sup>coeflegend does not appear in the dialog box.  
bootstrap, jackknife, mi estimate, rolling, statsby, and svy are allowed; see [U] 11.1.10 Prefix commands.  
vce(bootstrap) and vce(jackknife) are not allowed with the mi estimate prefix.  
Weights are not allowed with the bootstrap prefix.  
vce() and weights are not allowed with the svy prefix.  
fweights, iweights, and pweights are allowed; see [U] 11.1.6 weight.  
proportion shares the features of all estimation commands; see [U] 20 Estimation and postestimation commands.

Menu

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Proportions

Description

proportion produces estimates of proportions, along with standard errors, for the categories identified by the values in each variable of *varlist*.

## Options

### Model

`stdize(varname)` specifies that the point estimates be adjusted by direct standardization across the strata identified by *varname*. This option requires the `stdweight()` option.

`stdweight(varname)` specifies the weight variable associated with the standard strata identified in the `stdize()` option. The standardization weights must be constant within the standard strata.

`nostdrescale` prevents the standardization weights from being rescaled within the `over()` groups. This option requires `stdize()` but is ignored if the `over()` option is not specified.

`nolabel` specifies that value labels attached to the variables in *varlist* be ignored.

`missing` specifies that missing values in *varlist* be treated as valid categories, rather than omitted from the analysis (the default).

### if/in/over

`over(varlist [ , nolabel ])` specifies that estimates be computed for multiple subpopulations, which are identified by the different values of the variables in *varlist*.

When this option is supplied with one variable name, such as `over(varname)`, the value labels of *varname* are used to identify the subpopulations. If *varname* does not have labeled values (or there are unlabeled values), the values themselves are used, provided that they are nonnegative integers. Noninteger values, negative values, and labels that are not valid Stata names are substituted with a default identifier.

When `over()` is supplied with multiple variable names, each subpopulation is assigned a unique default identifier.

`nolabel` requests that value labels attached to the variables identifying the subpopulations be ignored.

### SE/Cluster

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

`vce(analytic)`, the default, uses the analytically derived variance estimator associated with the sample proportion.

### Reporting

`level(#)`; see [R] [estimation options](#).

`noheader` prevents the table header from being displayed. This option implies `nolegend`.

`nolegend` prevents the table legend identifying the subpopulations from being displayed.

The following option is available with `proportion` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

### ► Example 1

We can estimate the proportion of each repair rating in the auto data:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. proportion rep78

Proportion estimation Number of obs = 69
```

|       |   | Proportion | Std. Err. | [95% Conf. Interval] |          |
|-------|---|------------|-----------|----------------------|----------|
| rep78 | 1 | .0289855   | .0203446  | -.0116115            | .0695825 |
|       | 2 | .115942    | .0388245  | .0384689             | .1934152 |
|       | 3 | .4347826   | .0601159  | .3148232             | .554742  |
|       | 4 | .2608696   | .0532498  | .1546113             | .3671278 |
|       | 5 | .1594203   | .0443922  | .070837              | .2480036 |

Here we use the missing option to include missing values as a category of rep78:

```
. proportion rep78, missing

Proportion estimation Number of obs = 74
 _prop_6: rep78 = .
```

|         |   | Proportion | Std. Err. | [95% Conf. Interval] |          |
|---------|---|------------|-----------|----------------------|----------|
| rep78   | 1 | .027027    | .0189796  | -.0107994            | .0648534 |
|         | 2 | .1081081   | .0363433  | .0356761             | .1805401 |
|         | 3 | .4054054   | .0574637  | .2908804             | .5199305 |
|         | 4 | .2432432   | .0502154  | .1431641             | .3433224 |
|         | 5 | .1486486   | .0416364  | .0656674             | .2316299 |
| _prop_6 |   | .0675676   | .0293776  | .0090181             | .1261171 |

➤ Example 2

We can also estimate proportions over groups:

```
. proportion rep78, over(foreign)
Proportion estimation Number of obs = 69
 _prop_1: rep78 = 1
 _prop_2: rep78 = 2
 _prop_3: rep78 = 3
 _prop_4: rep78 = 4
 _prop_5: rep78 = 5
Domestic: foreign = Domestic
Foreign: foreign = Foreign
```

| Over                           | Proportion        | Std. Err. | [95% Conf. Interval] |          |
|--------------------------------|-------------------|-----------|----------------------|----------|
| _prop_1<br>Domestic<br>Foreign | .0416667          | .0291477  | -.0164966            | .0998299 |
|                                | (no observations) |           |                      |          |
| _prop_2<br>Domestic<br>Foreign | .1666667          | .0543607  | .0581916             | .2751417 |
|                                | (no observations) |           |                      |          |
| _prop_3<br>Domestic<br>Foreign | .5625             | .0723605  | .4181069             | .7068931 |
|                                | .1428571          | .0782461  | -.0132805            | .2989948 |
| _prop_4<br>Domestic<br>Foreign | .1875             | .0569329  | .0738921             | .3011079 |
|                                | .4285714          | .1106567  | .2077595             | .6493834 |
| _prop_5<br>Domestic<br>Foreign | .0416667          | .0291477  | -.0164966            | .0998299 |
|                                | .4285714          | .1106567  | .2077595             | .6493834 |



Saved results

proportion saves the following in e():

Scalars

|             |                           |         |                             |
|-------------|---------------------------|---------|-----------------------------|
| e(N)        | number of observations    | e(k_eq) | number of equations in e(b) |
| e(N_over)   | number of subpopulations  | e(df_r) | sample degrees of freedom   |
| e(N_stdize) | number of standard strata | e(rank) | rank of e(V)                |
| e(N_clust)  | number of clusters        |         |                             |

Macros

|              |                                 |                  |                                     |
|--------------|---------------------------------|------------------|-------------------------------------|
| e(cmd)       | proportion                      | e(over_labels)   | labels from over() variables        |
| e(cmdline)   | command as typed                | e(over_namelist) | names from e(over_labels)           |
| e(varlist)   | varlist                         | e(namelist)      | proportion identifiers              |
| e(stdize)    | <i>varname</i> from stdize()    | e(label#)        | labels from #th variable in varlist |
| e(stdweight) | <i>varname</i> from stdweight() | e(vce)           | vcetype specified in vce()          |
| e(wtype)     | weight type                     | e(vctype)        | title used to label Std. Err.       |
| e(wexp)      | weight expression               | e(properties)    | b v                                 |
| e(title)     | title in estimation output      | e(estat_cmd)     | program used to implement estat     |
| e(cluster)   | name of cluster variable        | e(marginsnotok)  | predictions disallowed by margins   |
| e(over)      | varlist from over()             |                  |                                     |

## Matrices

|                            |                                                              |
|----------------------------|--------------------------------------------------------------|
| <code>e(b)</code>          | vector of proportion estimates                               |
| <code>e(V)</code>          | (co)variance estimates                                       |
| <code>e(_N)</code>         | vector of numbers of nonmissing observations                 |
| <code>e(_N_stdsum)</code>  | number of nonmissing observations within the standard strata |
| <code>e(_p_stdsize)</code> | standardizing proportions                                    |
| <code>e(error)</code>      | error code corresponding to <code>e(b)</code>                |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`proportion` is implemented as an ado-file.

Proportions are means of indicator variables; see [\[R\] mean](#).

## References

Cochran, W. G. 1977. *Sampling Techniques*. 3rd ed. New York: Wiley.

Stuart, A., and J. K. Ord. 1994. *Kendall's Advanced Theory of Statistics: Distribution Theory, Vol I*. 6th ed. London: Arnold.

## Also see

[\[R\] proportion postestimation](#) — Postestimation tools for proportion

[\[R\] mean](#) — Estimate means

[\[R\] ratio](#) — Estimate ratios

[\[R\] total](#) — Estimate totals

[\[SVY\] direct standardization](#) — Direct standardization of means, proportions, and ratios

[\[SVY\] poststratification](#) — Poststratification for survey data

[\[SVY\] subpopulation estimation](#) — Subpopulation estimation for survey data

[\[SVY\] svy estimation](#) — Estimation commands for survey data

[\[SVY\] variance estimation](#) — Variance estimation for survey data

[\[U\] 20 Estimation and postestimation commands](#)

# Title

proportion postestimation — Postestimation tools for proportion

## Description

The following postestimation commands are available for `proportion`:

| command                  | description                                                                                         |
|--------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>       | VCE                                                                                                 |
| <code>estat (svy)</code> | postestimation statistics for survey data                                                           |
| <code>estimates</code>   | cataloging estimation results                                                                       |
| <code>lincom</code>      | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>nlcom</code>       | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>test</code>        | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>      | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

- [\[R\] proportion](#) — Estimate proportions
- [\[SVY\] svy postestimation](#) — Postestimation tools for svy
- [\[U\] 20 Estimation and postestimation commands](#)

# Title

prtest — One- and two-sample tests of proportions

## Syntax

*One-sample test of proportion*

```
prtest varname == #p [if] [in] [, level(#)]
```

*Two-sample test of proportion*

```
prtest varname1 == varname2 [if] [in] [, level(#)]
```

*Two-group test of proportion*

```
prtest varname [if] [in], by(groupvar) [level(#)]
```

*Immediate form of one-sample test of proportion*

```
prtesti #obs1 #p1 #p2 [, level(#) count]
```

*Immediate form of two-sample test of proportion*

```
prtesti #obs1 #p1 #obs2 #p2 [, level(#) count]
```

*by* is allowed with *prtest*; see [\[D\]](#) *by*.

## Menu

### one-sample

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > One-sample proportion test

### two-sample

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > Two-sample proportion test

### two-group

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > Two-group proportion test

### immediate command: one-sample

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > One-sample proportion calculator

### immediate command: two-sample

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > Two-sample proportion calculator



## Description

`prtest` performs tests on the equality of proportions using large-sample statistics.

In the first form, `prtest` tests that *varname* has a proportion of  $\#_p$ . In the second form, `prtest` tests that *varname*<sub>1</sub> and *varname*<sub>2</sub> have the same proportion. In the third form, `prtest` tests that *varname* has the same proportion within the two groups defined by *groupvar*.

`prtesti` is the immediate form of `prtest`; see [U] 19 Immediate commands.

The `bitest` command is a better version of the first form of `prtest` in that it gives exact *p*-values. Researchers should use `bitest` when possible, especially for small samples; see [R] [bitest](#).

## Options

### Main

`by(groupvar)` specifies a numeric variable that contains the group information for a given observation.

This variable must have only two values. Do not confuse the `by()` option with the `by` prefix; both may be specified.

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see [U] 20.7 Specifying the width of confidence intervals.

`count` specifies that integer counts instead of proportions be used in the immediate forms of `prtest`.

In the first syntax, `prtesti` expects that  $\#_{\text{obs1}}$  and  $\#_{p1}$  are counts— $\#_{p1} \leq \#_{\text{obs1}}$ —and  $\#_{p2}$  is a proportion. In the second syntax, `prtesti` expects that all four numbers are integer counts, that  $\#_{\text{obs1}} \geq \#_{p1}$ , and that  $\#_{\text{obs2}} \geq \#_{p2}$ .

## Remarks

The `prtest` output follows the output of `ttest` in providing a lot of information. Each proportion is presented along with a confidence interval. The appropriate one- or two-sample test is performed, and the two-sided and both one-sided results are included at the bottom of the output. For a two-sample test, the calculated difference is also presented with its confidence interval. This command may be used for both large-sample testing and large-sample interval estimation.

### ► Example 1: One-sample test of proportion

In the first form, `prtest` tests whether the mean of the sample is equal to a known constant. Assume that we have a sample of 74 automobiles. We wish to test whether the proportion of automobiles that are foreign is different from 40%.

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
```

```
. prtest foreign == .4
```

```
One-sample test of proportion foreign: Number of obs = 74
```

| Variable | Mean     | Std. Err. | [95% Conf. Interval] |          |
|----------|----------|-----------|----------------------|----------|
| foreign  | .2972973 | .0531331  | .1931583             | .4014363 |

```
 p = proportion(foreign) z = -1.8034
Ho: p = 0.4
```

```
 Ha: p < 0.4 Ha: p != 0.4 Ha: p > 0.4
Pr(Z < z) = 0.0357 Pr(|Z| > |z|) = 0.0713 Pr(Z > z) = 0.9643
```

△

► Example 2: Two-sample test of proportion

We have two headache remedies that we give to patients. Each remedy's effect is recorded as 0 for failing to relieve the headache and 1 for relieving the headache. We wish to test the equality of the proportion of people relieved by the two treatments.

```
. use http://www.stata-press.com/data/r11/cure
. prtest cure1 == cure2
```

```
cure2: Number of obs = 59
```

| Variable | Mean      | Std. Err. | z     | P> z  | [95% Conf. Interval] |
|----------|-----------|-----------|-------|-------|----------------------|
| cure1    | .52       | .0706541  |       |       | .3815205 .6584795    |
| cure2    | .7118644  | .0589618  |       |       | .5963013 .8274275    |
| diff     | -.1918644 | .0920245  |       |       | -.372229 -.0114998   |
|          | under Ho: | .0931155  | -2.06 | 0.039 |                      |

```

 diff = prop(cure1) - prop(cure2) z = -2.0605
Ho: diff = 0
Ha: diff < 0 Ha: diff != 0 Ha: diff > 0
Pr(Z < z) = 0.0197 Pr(|Z| < |z|) = 0.0394 Pr(Z > z) = 0.9803

```

◀

► Example 3: Immediate form of one-sample test of proportion

`prtesti` is like `prtest`, except that you specify summary statistics rather than variables as arguments. For instance, we are reading an article that reports the proportion of registered voters among 50 randomly selected eligible voters as 0.52. We wish to test whether the proportion is 0.7:

```
. prtesti 50 .52 .70
```

---

| Variable | Mean | Std. Err. | [95% Conf. Interval] |          |
|----------|------|-----------|----------------------|----------|
| x        | .52  | .0706541  | .3815205             | .6584795 |

|                    |                        |                    |
|--------------------|------------------------|--------------------|
| p = proportion(x)  |                        | z = -2.7775        |
| Ho: p = 0.7        |                        |                    |
| Ha: p < 0.7        | Ha: p != 0.7           | Ha: p > 0.7        |
| Pr(Z < z) = 0.0027 | Pr( Z  >  z ) = 0.0055 | Pr(Z > z) = 0.9973 |

△

### ► Example 4: Immediate form of two-sample test of proportion

To judge teacher effectiveness, we wish to test whether the same proportion of people from two classes will answer an advanced question correctly. In the first classroom of 30 students, 40% answered the question correctly, whereas in the second classroom of 45 students, 67% answered the question correctly.

```
. prtesti 30 .4 45 .67
```

Two-sample test of proportion

x: Number of obs = 30

y: Number of obs = 45

| Variable | Mean      | Std. Err. | z     | P> z  | [95% Conf. Interval] |
|----------|-----------|-----------|-------|-------|----------------------|
| x        | .4        | .0894427  |       |       | .2246955 .5753045    |
| y        | .67       | .0700952  |       |       | .532616 .807384      |
| diff     | -.27      | .1136368  |       |       | -.4927241 -.0472759  |
|          | under Ho: | .1169416  | -2.31 | 0.021 |                      |

diff = prop(x) - prop(y)

z = -2.3088

Ho: diff = 0

Ha: diff < 0

Pr(Z < z) = 0.0105

Ha: diff != 0

Pr(|Z| < |z|) = 0.0210

Ha: diff > 0

Pr(Z > z) = 0.9895

## Saved results

prtest and prtesti save the following in r():

Scalars

r(z) z statistic

r(N\_#) number of observations for variable #

r(P\_#) proportion for variable #

## Methods and formulas

prtest and prtesti are implemented as ado-files.

See [Acock \(2008, 136–142\)](#) for additional examples of tests of proportions using Stata.

A large-sample  $(1 - \alpha)100\%$  confidence interval for a proportion  $p$  is

$$\hat{p} \pm z_{1-\alpha/2} \sqrt{\frac{\hat{p}\hat{q}}{n}}$$

and a  $(1 - \alpha)100\%$  confidence for the difference of two proportions is given by

$$(\hat{p}_1 - \hat{p}_2) \pm z_{1-\alpha/2} \sqrt{\frac{\hat{p}_1\hat{q}_1}{n_1} + \frac{\hat{p}_2\hat{q}_2}{n_2}}$$

where  $\hat{q} = 1 - \hat{p}$  and  $z$  is calculated from the inverse normal distribution.

The one-tailed and two-tailed tests of a population proportion use a normally distributed test statistic calculated as

$$z = \frac{\hat{p} - p_0}{\sqrt{p_0q_0/n}}$$

where  $p_0$  is the hypothesized proportion. A test of the difference of two proportions also uses a normally distributed test statistic calculated as

$$z = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\hat{p}_p \hat{q}_p (1/n_1 + 1/n_2)}}$$

where

$$\hat{p}_p = \frac{x_1 + x_2}{n_1 + n_2}$$

and  $x_1$  and  $x_2$  are the total number of successes in the two populations.

## References

- Acock, A. C. 2008. *A Gentle Introduction to Stata*. 2nd ed. College Station, TX: Stata Press.
- Wang, D. 2000. [sg154: Confidence intervals for the ratio of two binomial proportions by Koopman's method](#). *Stata Technical Bulletin* 58: 16–19. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 244–247. College Station, TX: Stata Press.

## Also see

- [\[R\] \*\*bitest\*\*](#) — Binomial probability test
- [\[R\] \*\*proportion\*\*](#) — Estimate proportions
- [\[R\] \*\*ttest\*\*](#) — Mean-comparison tests
- [\[MV\] \*\*hotelling\*\*](#) — Hotelling's T-squared generalized means test

Syntax

Draw a *c* chart

```
cchart defect_var unit_var [, cchart_options]
```

Draw a *p* (fraction-defective) chart

```
pchart reject_var unit_var ssize_var [, pchart_options]
```

Draw an *R* (range or dispersion) chart

```
rchart varlist [if] [in] [, rchart_options]
```

Draw an  $\bar{X}$  (control line) chart

```
xchart varlist [if] [in] [, xchart_options]
```

Draw vertically aligned  $\bar{X}$  and *R* charts

```
shewhart varlist [if] [in] [, shewhart_options]
```

| <i>cchart_options</i>                   | description                                                                                      |
|-----------------------------------------|--------------------------------------------------------------------------------------------------|
| Main                                    |                                                                                                  |
| <u>nograph</u>                          | suppress graph                                                                                   |
| Plot                                    |                                                                                                  |
| <i>connect_options</i>                  | affect rendition of the plotted points                                                           |
| <i>marker_options</i>                   | change look of markers (color, size, etc.)                                                       |
| <i>marker_label_options</i>             | add marker labels; change look or position                                                       |
| Control limits                          |                                                                                                  |
| <u>clopts</u> ( <i>ccline_options</i> ) | affect rendition of the control limits                                                           |
| Add plots                               |                                                                                                  |
| <u>addplot</u> ( <i>plot</i> )          | add other plots to the generated graph                                                           |
| Y axis, X axis, Titles, Legend, Overall |                                                                                                  |
| <i>twoway_options</i>                   | any options other than <code>by()</code> documented in <a href="#">[G]</a> <i>twoway_options</i> |

| <i>pchart_options</i>                                                                                 | description                                                                                      |
|-------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| <hr/>                                                                                                 |                                                                                                  |
| Main                                                                                                  |                                                                                                  |
| <u>stabilized</u>                                                                                     | stabilize the p chart when sample sizes are unequal                                              |
| <u>nograph</u>                                                                                        | suppress graph                                                                                   |
| <u>generate</u> ( <i>newvar<sub>f</sub></i> <i>newvar<sub>lcl</sub></i> <i>newvar<sub>ucl</sub></i> ) | store the fractions of defective elements and the lower and upper control limits                 |
| <hr/>                                                                                                 |                                                                                                  |
| Plot                                                                                                  |                                                                                                  |
| <i>connect_options</i>                                                                                | affect rendition of the plotted points                                                           |
| <i>marker_options</i>                                                                                 | change look of markers (color, size, etc.)                                                       |
| <i>marker_label_options</i>                                                                           | add marker labels; change look or position                                                       |
| <hr/>                                                                                                 |                                                                                                  |
| Control limits                                                                                        |                                                                                                  |
| <u>clopts</u> ( <i>cline_options</i> )                                                                | affect rendition of the control limits                                                           |
| <hr/>                                                                                                 |                                                                                                  |
| Add plots                                                                                             |                                                                                                  |
| <u>addplot</u> ( <i>plot</i> )                                                                        | add other plots to the generated graph                                                           |
| <hr/>                                                                                                 |                                                                                                  |
| Y axis, X axis, Titles, Legend, Overall                                                               |                                                                                                  |
| <i>twoway_options</i>                                                                                 | any options other than <code>by()</code> documented in <a href="#">[G] <i>twoway_options</i></a> |
| <hr/>                                                                                                 |                                                                                                  |

| <i>rchart_options</i>                   | description                                                                                      |
|-----------------------------------------|--------------------------------------------------------------------------------------------------|
| <hr/>                                   |                                                                                                  |
| Main                                    |                                                                                                  |
| <u>std</u> (#)                          | user-specified standard deviation                                                                |
| <u>nograph</u>                          | suppress graph                                                                                   |
| <hr/>                                   |                                                                                                  |
| Plot                                    |                                                                                                  |
| <i>connect_options</i>                  | affect rendition of the plotted points                                                           |
| <i>marker_options</i>                   | change look of markers (color, size, etc.)                                                       |
| <i>marker_label_options</i>             | add marker labels; change look or position                                                       |
| <hr/>                                   |                                                                                                  |
| Control limits                          |                                                                                                  |
| <u>clopts</u> ( <i>cline_options</i> )  | affect rendition of the control limits                                                           |
| <hr/>                                   |                                                                                                  |
| Add plots                               |                                                                                                  |
| <u>addplot</u> ( <i>plot</i> )          | add other plots to the generated graph                                                           |
| <hr/>                                   |                                                                                                  |
| Y axis, X axis, Titles, Legend, Overall |                                                                                                  |
| <i>twoway_options</i>                   | any options other than <code>by()</code> documented in <a href="#">[G] <i>twoway_options</i></a> |
| <hr/>                                   |                                                                                                  |

| <i>xchart_options</i>                   | description                                                                                      |
|-----------------------------------------|--------------------------------------------------------------------------------------------------|
| Main                                    |                                                                                                  |
| <u>std</u> (#)                          | user-specified standard deviation                                                                |
| <u>mean</u> (#)                         | user-specified mean                                                                              |
| <u>lower</u> (#) <u>upper</u> (#)       | lower and upper limits of the X-bar limits                                                       |
| <u>nograph</u>                          | suppress graph                                                                                   |
| Plot                                    |                                                                                                  |
| <i>connect_options</i>                  | affect rendition of the plotted points                                                           |
| <i>marker_options</i>                   | change look of markers (color, size, etc.)                                                       |
| <i>marker_label_options</i>             | add marker labels; change look or position                                                       |
| Control limits                          |                                                                                                  |
| <u>clopts</u> ( <i>cline_options</i> )  | affect rendition of the control limits                                                           |
| Add plots                               |                                                                                                  |
| <u>addplot</u> ( <i>plot</i> )          | add other plots to the generated graph                                                           |
| Y axis, X axis, Titles, Legend, Overall |                                                                                                  |
| <i>twoway_options</i>                   | any options other than <code>by()</code> documented in <a href="#">[G] <i>twoway_options</i></a> |

| <i>shewhart_options</i>                 | description                                                        |
|-----------------------------------------|--------------------------------------------------------------------|
| Main                                    |                                                                    |
| <u>std</u> (#)                          | user-specified standard deviation                                  |
| <u>mean</u> (#)                         | user-specified mean                                                |
| <u>nograph</u>                          | suppress graph                                                     |
| Plot                                    |                                                                    |
| <i>connect_options</i>                  | affect rendition of the plotted points                             |
| <i>marker_options</i>                   | change look of markers (color, size, etc.)                         |
| <i>marker_label_options</i>             | add marker labels; change look or position                         |
| Control limits                          |                                                                    |
| <u>clopts</u> ( <i>cline_options</i> )  | affect rendition of the control limits                             |
| Y axis, X axis, Titles, Legend, Overall |                                                                    |
| <i>combine_options</i>                  | any options documented in <a href="#">[G] <i>graph combine</i></a> |

## Menu

### cchart

Statistics > Other > Quality control > C chart

### pchart

Statistics > Other > Quality control > P chart

## rchart

Statistics > Other > Quality control > R chart

## xchart

Statistics > Other > Quality control > X-bar chart

## shewhart

Statistics > Other > Quality control > Vertically aligned X-bar and R chart

# Description

These commands provide standard quality-control charts. `cchart` draws a `c` chart; `pchart`, a `p` (fraction-defective) chart; `rchart`, an `R` (range or dispersion) chart; `xchart`, an  $\bar{X}$  (control line) chart; and `shewhart`, vertically aligned  $\bar{X}$  and `R` charts.

# Options

### Main

`stabilized` stabilizes the `p` chart when sample sizes are unequal.

`std(#)` specifies the standard deviation of the process. The `R` chart is calculated (based on the range) if this option is not specified.

`mean(#)` specifies the grand mean, which is calculated if not specified.

`lower(#)` and `upper(#)` must be specified together or not at all. They specify the lower and upper limits of the  $\bar{X}$  chart. Calculations based on the mean and standard deviation (whether specified by option or calculated) are used otherwise.

`nograph` suppresses the graph.

`generate(newvarf newvarlcl newvarucl)` stores the plotted values in the `p` chart. `newvarf` will contain the fractions of defective elements; `newvarlcl` and `newvarucl` will contain the lower and upper control limits, respectively.

### Plot

`connect_options` affect whether lines connect the plotted points and the rendition of those lines; see [\[G\] connect\\_options](#).

`marker_options` affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [\[G\] marker\\_options](#).

`marker_label_options` specify if and how the markers are to be labeled; see [\[G\] marker\\_label\\_options](#).

### Control limits

`clopts(cline_options)` affects the rendition of the control limits; see [\[G\] cline\\_options](#).

### Add plots

`addplot(plot)` provides a way to add other plots to the generated graph. See [\[G\] addplot\\_option](#).



Y axis, X axis, Titles, Legend, Overall

`twoway_options` are any of the options documented in [G] [twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

`combine_options` (`shewhart` only) are any of the options documented in [G] [graph combine](#). These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

Remarks

Control charts may be used to define the goal of a repetitive process, to control that process, and to determine if the goal has been achieved. Walter A. Shewhart of Bell Telephone Laboratories devised the first control chart in 1924. In 1931, Shewhart published *Economic Control of Quality of Manufactured Product*. According to Burr, “Few fields of knowledge have ever been so completely explored and charted in the first exposition” (1976, 29). Shewhart states that “a phenomenon will be said to be controlled when, through the use of past experience, we can predict, at least within limits, how the phenomenon may be expected to vary in the future. Here it is understood that prediction within limits means that we can state, at least approximately, the probability that the observed phenomenon will fall within given limits” (1931, 6).

For more information on quality-control charts, see Burr (1976), Duncan (1986), Harris (1999), or Ryan (2000).

► Example 1: cchart

`cchart` graphs a c chart showing the number of nonconformities in a unit, where `defect_var` records the number of defects in each inspection unit and `unit_var` records the unit number. The unit numbers need not be in order. For instance, consider the following example dataset from Ryan (2000, 156):

```
. use http://www.stata-press.com/data/r11/ncu
. describe
Contains data from http://www.stata-press.com/data/r11/ncu.dta
obs: 30
vars: 2 31 Mar 2009 03:56
size: 360 (99.9% of memory free)
```

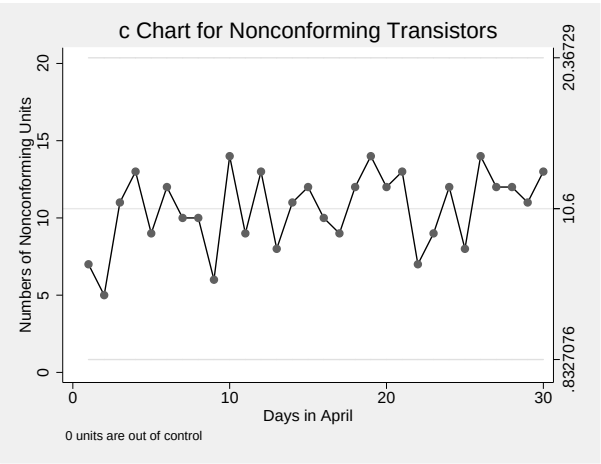
| variable name | storage type | display format | value label | variable label                 |
|---------------|--------------|----------------|-------------|--------------------------------|
| day           | float        | %9.0g          |             | Days in April                  |
| defects       | float        | %9.0g          |             | Numbers of Nonconforming Units |

Sorted by:

```
. list in 1/5
```

|    | day | defects |
|----|-----|---------|
| 1. | 1   | 7       |
| 2. | 2   | 5       |
| 3. | 3   | 11      |
| 4. | 4   | 13      |
| 5. | 5   | 9       |

```
. cchart defects day, title(c Chart for Nonconforming Transistors)
```



The expected number of defects is 10.6, with lower and upper control limits of 0.8327 and 20.37, respectively. No units are out of control.



➤ Example 2: pchart

pchart graphs a p chart, which shows the fraction of nonconforming items in a subgroup, where *reject\_var* records the number rejected in each inspection unit, *unit\_var* records the inspection unit number, and *ssize\_var* records the number inspected in each unit.

Consider the example dataset from [Ryan \(2000, 156\)](#) of the number of nonconforming transistors out of 1,000 inspected each day during the month of April:

```
. use http://www.stata-press.com/data/r11/ncu2
. describe
Contains data from http://www.stata-press.com/data/r11/ncu2.dta
 obs: 30
 vars: 3 31 Mar 2009 14:13
size: 480 (99.9% of memory free)
```

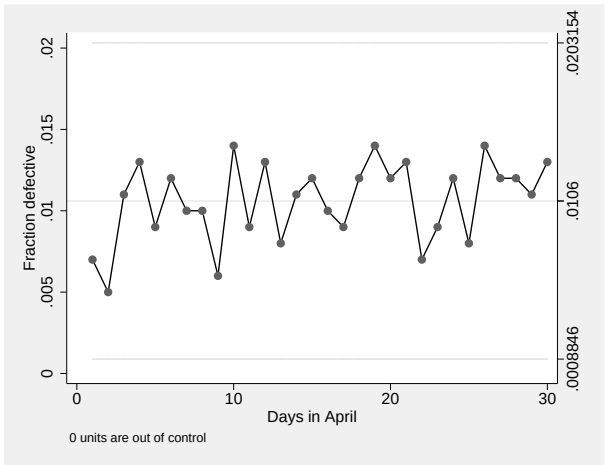
| variable name | storage type | display format | value label | variable label                 |
|---------------|--------------|----------------|-------------|--------------------------------|
| day           | float        | %9.0g          |             | Days in April                  |
| rejects       | float        | %9.0g          |             | Numbers of Nonconforming Units |
| ssize         | float        | %9.0g          |             | Sample size                    |

Sorted by:

```
. list in 1/5
```

|    | day | rejects | ssize |
|----|-----|---------|-------|
| 1. | 1   | 7       | 1000  |
| 2. | 2   | 5       | 1000  |
| 3. | 3   | 11      | 1000  |
| 4. | 4   | 13      | 1000  |
| 5. | 5   | 9       | 1000  |

```
. pchart rejects day ssize
```



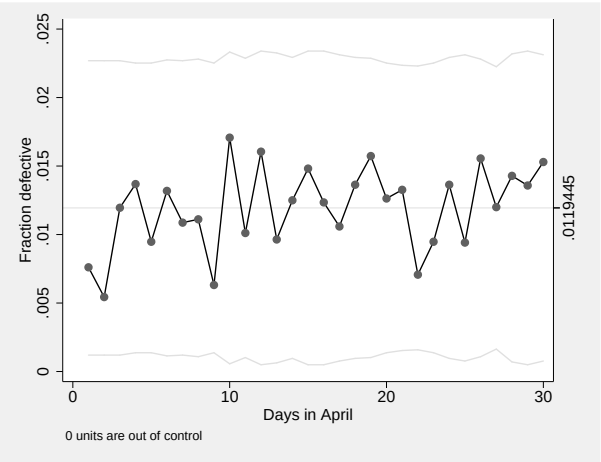
All the points are within the control limits, which are 0.0009 for the lower limit and 0.0203 for the upper limit.

Here the sample sizes are fixed at 1,000, so the `ssize` variable contains 1,000 for each observation. Sample sizes need not be fixed, however. Say that our data were slightly different:

```
. use http://www.stata-press.com/data/r11/ncu3
. list in 1/5
```

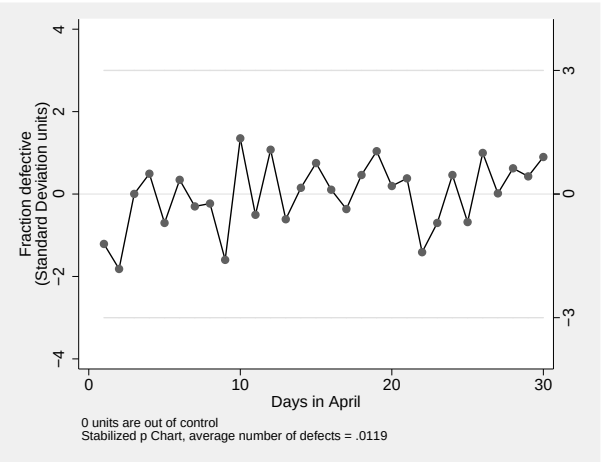
|    | day | rejects | ssize |
|----|-----|---------|-------|
| 1. | 1   | 7       | 920   |
| 2. | 2   | 5       | 920   |
| 3. | 3   | 11      | 920   |
| 4. | 4   | 13      | 950   |
| 5. | 5   | 9       | 950   |

```
. pchart rejects day ssize
```



Here the control limits are, like the sample size, no longer constant. The `stabilize` option will stabilize the control chart:

```
. pchart rejects day ssize, stabilize
```



### ► Example 3: rchart

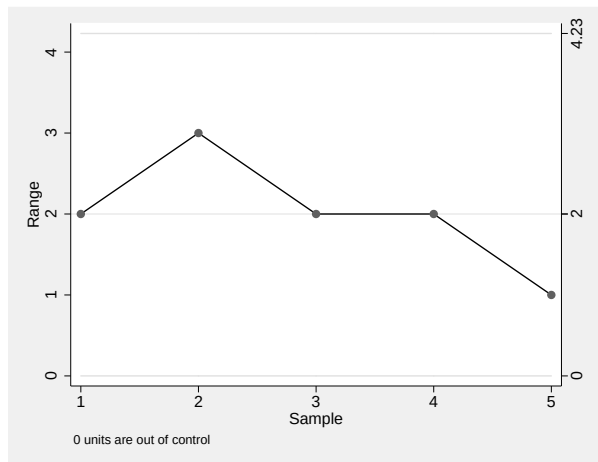
`rchart` displays an R chart showing the range for repeated measurements at various times. Variables within observations record measurements. Observations represent different samples.

For instance, say that we take five samples of 5 observations each. In our first sample, our measurements are 10, 11, 10, 11, and 12. The data are

```
. list
```

|    | m1 | m2 | m3 | m4 | m5 |
|----|----|----|----|----|----|
| 1. | 10 | 11 | 10 | 11 | 12 |
| 2. | 12 | 10 | 9  | 10 | 9  |
| 3. | 10 | 11 | 10 | 12 | 10 |
| 4. | 9  | 9  | 9  | 10 | 11 |
| 5. | 12 | 12 | 12 | 12 | 13 |

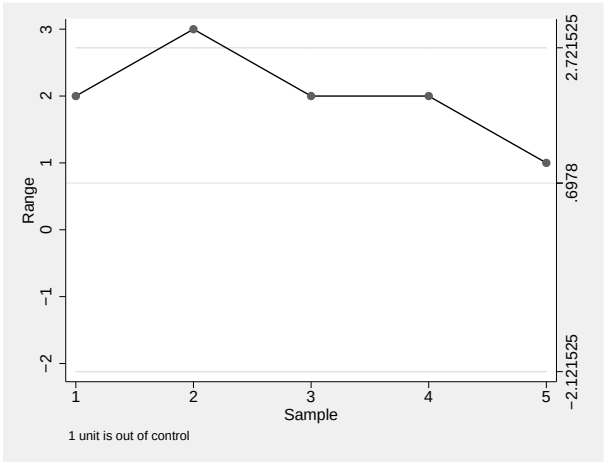
```
. rchart m1-m5, connect(1)
```



The expected range in each sample is 2 with lower and upper control limits of 0 and 4.23, respectively. If we know that the process standard deviation is 0.3, we could specify

*(Continued on next page)*

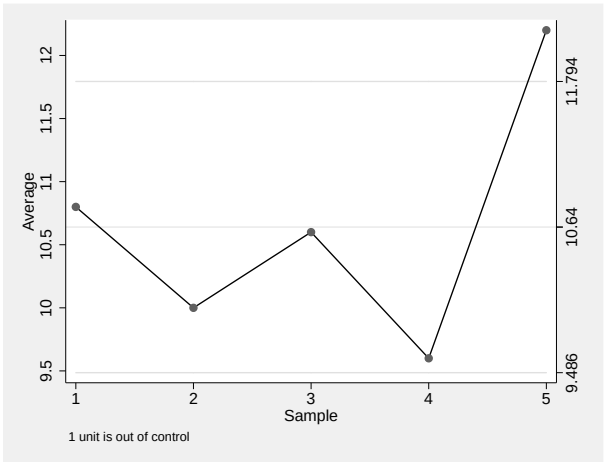
```
. rchart m1-m5, connect(1) std(.3)
```



### Example 4: xchart

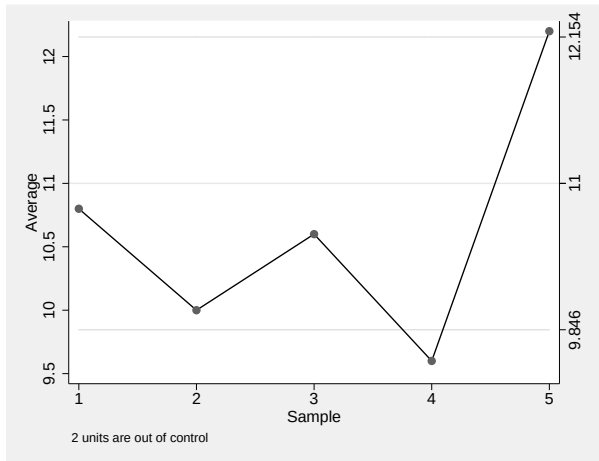
`xchart` graphs an  $\bar{X}$  chart for repeated measurements at various times. Variables within observations record measurements, and observations represent different samples. Using the same data as in the previous example, we type

```
. xchart m1-m5, connect(1)
```



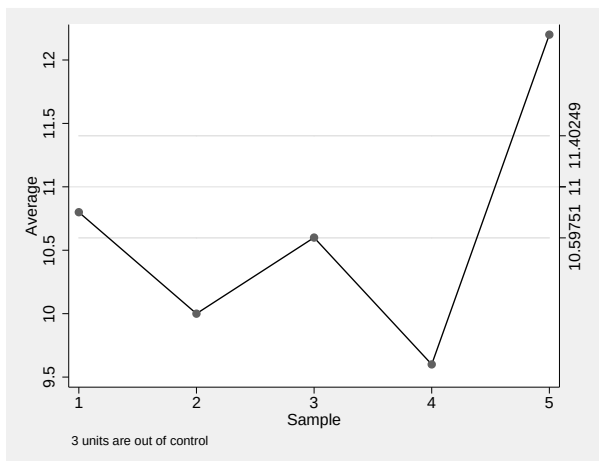
The average measurement in the sample is 10.64, and the lower and upper control limits are 9.486 and 11.794, respectively. Suppose that we knew from prior information that the mean of the process is 11. Then we would type

```
. xchart m1-m5, connect(1) mean(11)
```



If we also know that the standard deviation of the process is 0.3, we could type

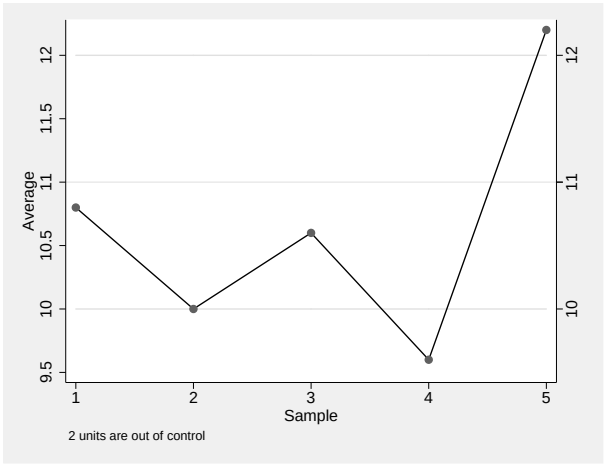
```
. xchart m1-m5, connect(1) mean(11) std(.3)
```



Finally, `xchart` allows us to specify our own control limits:

*(Continued on next page)*

```
. xchart m1-m5, connect(1) mean(11) lower(10) upper(12)
```



Walter Andrew Shewhart (1891–1967) was born in Illinois and educated as a physicist, with degrees from the Universities of Illinois and California. After a brief period teaching physics, he worked for the Western Electric Company and (from 1925) the Bell Telephone Laboratories. His name is most associated with control charts used in quality controls, but his many other interests ranged from quality assurance generally to the philosophy of science.

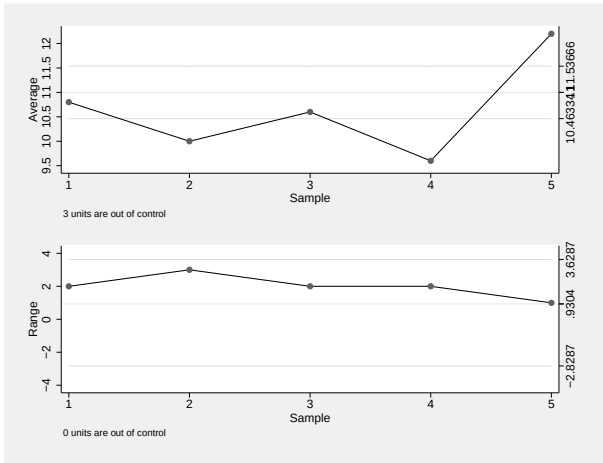
➤ Example 5: shewhart

`shewhart` displays a vertically aligned  $\bar{X}$  and R chart in the same image. To produce the best-looking combined image possible, you will want to use the `xchart` and `rchart` commands separately and then combine the graphs. `shewhart`, however, is more convenient.

Using the same data as previously, but realizing that the standard deviation should have been 0.4, we type



```
. shewhart m1-m5, connect(1) mean(11) std(.4)
```



## Saved results

**cchart** saves the following in **r()**:

Scalars

|                 |                                    |                   |                                       |
|-----------------|------------------------------------|-------------------|---------------------------------------|
| <b>r(cbar)</b>  | expected number of nonconformities | <b>r(out_c)</b>   | number of units out of control        |
| <b>r(lcl_c)</b> | lower control limit                | <b>r(below_c)</b> | number of units below the lower limit |
| <b>r(ucl_c)</b> | upper control limit                | <b>r(above_c)</b> | number of units above the upper limit |
| <b>r(N)</b>     | number of observations             |                   |                                       |

**pchart** saves the following in **r()**:

Scalars

|                 |                                     |                   |                                       |
|-----------------|-------------------------------------|-------------------|---------------------------------------|
| <b>r(pbar)</b>  | average fraction of nonconformities | <b>r(out_p)</b>   | number of units out of control        |
| <b>r(lcl_p)</b> | lower control limit                 | <b>r(below_p)</b> | number of units below the lower limit |
| <b>r(ucl_p)</b> | upper control limit                 | <b>r(above_p)</b> | number of units above the upper limit |
| <b>r(N)</b>     | number of observations              |                   |                                       |

**rchart** saves the following in **r()**:

Scalars

|                        |                              |                   |                                       |
|------------------------|------------------------------|-------------------|---------------------------------------|
| <b>r(central_line)</b> | ordinate of the central line | <b>r(out_r)</b>   | number of units out of control        |
| <b>r(lcl_r)</b>        | lower control limit          | <b>r(below_r)</b> | number of units below the lower limit |
| <b>r(ucl_r)</b>        | upper control limit          | <b>r(above_r)</b> | number of units above the upper limit |
| <b>r(N)</b>            | number of observations       |                   |                                       |

**xchart** saves the following in **r()**:

Scalars

|                 |                        |                   |                                       |
|-----------------|------------------------|-------------------|---------------------------------------|
| <b>r(xbar)</b>  | grand mean             | <b>r(out_x)</b>   | number of units out of control        |
| <b>r(lcl_x)</b> | lower control limit    | <b>r(below_x)</b> | number of units below the lower limit |
| <b>r(ucl_x)</b> | upper control limit    | <b>r(above_x)</b> | number of units above the upper limit |
| <b>r(N)</b>     | number of observations |                   |                                       |

**shewhart** saves in **r()** the combination of saved results from **xchart** and **rchart**.

## Methods and formulas

`cchart`, `pchart`, `rchart`, `xchart`, and `shewhart` are implemented as ado-files.

For the `c` chart, the number of defects per unit,  $C$ , is taken to be a value of a random variable having a Poisson distribution. If  $k$  is the number of units available for estimating  $\lambda$ , the parameter of the Poisson distribution, and if  $C_i$  is the number of defects in the  $i$ th unit, then  $\lambda$  is estimated by  $\bar{C} = \sum_i C_i/k$ . Then

$$\begin{aligned}\text{central line} &= \bar{C} \\ \text{UCL} &= \bar{C} + 3\sqrt{\bar{C}} \\ \text{LCL} &= \bar{C} - 3\sqrt{\bar{C}}\end{aligned}$$

Control limits for the `p` chart are based on the sampling theory for proportions, using the normal approximation to the binomial. If  $k$  samples are taken, the estimator of  $p$  is given by  $\bar{p} = \sum_i \hat{p}_i/k$ , where  $\hat{p}_i = x_i/n_i$ , and  $x_i$  is the number of defects in the  $i$ th sample of size  $n_i$ . The central line and the control limits are given by

$$\begin{aligned}\text{central line} &= \bar{p} \\ \text{UCL} &= \bar{p} + 3\sqrt{\bar{p}(1-\bar{p})/n_i} \\ \text{LCL} &= \bar{p} - 3\sqrt{\bar{p}(1-\bar{p})/n_i}\end{aligned}$$

Control limits for the `R` chart are based on the distribution of the range of samples of size  $n$  from a normal population. If the standard deviation of the process,  $\sigma$ , is known,

$$\begin{aligned}\text{central line} &= d_2\sigma \\ \text{UCL} &= D_2\sigma \\ \text{LCL} &= D_1\sigma\end{aligned}$$

where  $d_2$ ,  $D_1$ , and  $D_2$  are functions of the number of observations in the sample and are obtained from the table published in [Beyer \(1976\)](#).

When  $\sigma$  is unknown,

$$\begin{aligned}\text{central line} &= \bar{R} \\ \text{UCL} &= (D_2/d_2)\bar{R} \\ \text{LCL} &= (D_1/d_2)\bar{R}\end{aligned}$$

where  $\bar{R} = \sum_i R_i/k$  is the range of the  $k$  sample ranges  $R_i$ .

Control limits for the  $\bar{X}$  chart are given by

$$\text{central line} = \bar{x}$$

$$\text{UCL} = \bar{x} + (3/\sqrt{n})\sigma$$

$$\text{LCL} = \bar{x} - (3/\sqrt{n})\sigma$$

if  $\sigma$  is known. If  $\sigma$  is unknown,

$$\text{central line} = \bar{x}$$

$$\text{UCL} = \bar{x} + A_2\bar{R}$$

$$\text{LCL} = \bar{x} - A_2\bar{R}$$

where  $\bar{R}$  is the average range as defined above and  $A_2$  is a function (op. cit.) of the number of observations in the sample.

## References

- Bayart, D. 2001. Walter Andrew Shewhart. In *Statisticians of the Centuries*, ed. C. C. Heyde and E. Seneta, 398–401. New York: Springer.
- Beyer, W. H. 1976. Factors for computing control limits. In Vol. 2 of *Handbook of Tables for Probability and Statistics*, ed. W. H. Beyer, 451–465. Cleveland, OH: The Chemical Rubber Company.
- Burr, I. W. 1976. *Statistical Quality Control Methods*. New York: Dekker.
- Caulcutt, R. 2004. Control charts in practice. *Significance* 1: 81–84.
- Duncan, A. J. 1986. *Quality Control and Industrial Statistics*. 5th ed. Homewood, IL: Irwin.
- Harris, R. L. 1999. *Information Graphics: A Comprehensive Illustrated Reference*. New York: Oxford University Press.
- Ryan, T. P. 2000. *Statistical Methods for Quality Improvement*. 2nd ed. New York: Wiley.
- Saw, S. L. C., and T. W. Soon. 1994. [sqc1: Estimating process capability indices with Stata](#). *Stata Technical Bulletin* 17: 18–19. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 174–175. College Station, TX: Stata Press.
- Shewhart, W. A. 1931. *Economic Control of Quality of Manufactured Product*. New York: Van Nostrand.

## Also see

[R] [serrbar](#) — Graph standard-error bar chart

Syntax

Quantile regression

```
qreg depvar [indepvars] [if] [in] [weight] [, qreg_options]
```

Interquantile range regression

```
iqreg depvar [indepvars] [if] [in] [, iqreg_options]
```

Simultaneous-quantile regression

```
sqreg depvar [indepvars] [if] [in] [, sqreg_options]
```

Quantile regression with bootstrap standard errors

```
bsqreg depvar [indepvars] [if] [in] [, bsqreg_options]
```

Internal estimation command for quantile regression

```
_qreg [depvar [indepvars] [if] [in] [weight]] [, _qreg_options]
```

| qreg_options         | description                                                                            |
|----------------------|----------------------------------------------------------------------------------------|
| Model                |                                                                                        |
| quantile(#)          | estimate # quantile; default is quantile(.5)                                           |
| Reporting            |                                                                                        |
| level(#)             | set confidence level; default is level(95)                                             |
| Optimization         |                                                                                        |
| optimization_options | control the optimization process; seldom used                                          |
| wlsiter(#)           | attempt # weighted least-squares iterations before doing linear programming iterations |

| iqreg_options  | description                                           |
|----------------|-------------------------------------------------------|
| Model          |                                                       |
| quantiles(# #) | interquantile range; default is quantiles(.25 .75)    |
| reps(#)        | perform # bootstrap replications; default is reps(20) |
| Reporting      |                                                       |
| level(#)       | set confidence level; default is level(95)            |
| nodots         | suppress display of the replication dots              |

| <i>sqreg_options</i>                   | description                                                        |
|----------------------------------------|--------------------------------------------------------------------|
| Model                                  |                                                                    |
| <code>quantiles(# [# [# ... ]])</code> | estimate # quantiles; default is <code>quantiles(.5)</code>        |
| <code>reps(#)</code>                   | perform # bootstrap replications; default is <code>reps(20)</code> |
| Reporting                              |                                                                    |
| <code>level(#)</code>                  | set confidence level; default is <code>level(95)</code>            |
| <code>nodots</code>                    | suppress display of the replication dots                           |

| <i>bsqreg_options</i>    | description                                                        |
|--------------------------|--------------------------------------------------------------------|
| Model                    |                                                                    |
| <code>quantile(#)</code> | estimate # quantile; default is <code>quantile(.5)</code>          |
| <code>reps(#)</code>     | perform # bootstrap replications; default is <code>reps(20)</code> |
| Reporting                |                                                                    |
| <code>level(#)</code>    | set confidence level; default is <code>level(95)</code>            |

| <i>_qreg_options</i>        | description                                                                             |
|-----------------------------|-----------------------------------------------------------------------------------------|
| <code>quantile(#)</code>    | estimate # quantile; default is <code>quantile(.5)</code>                               |
| <code>level(#)</code>       | set confidence level; default is <code>level(95)</code>                                 |
| <code>accuracy(#)</code>    | relative accuracy required for linear programming algorithm;<br>should not be specified |
| <i>optimization_options</i> | control the optimization process; seldom used                                           |

by, mi estimate, rolling, statsby, and xi are allowed by `qreg`, `iqreg`, `sqreg`, and `bsqreg`; `fracpoly`, `mfp`, `nestreg`, and `stepwise` are allowed only with `qreg`; see [U] 11.1.10 Prefix commands.

`qreg` and `_qreg` allow `aweight`s and `fweight`s; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

## Menu

### qreg

Statistics > Nonparametric analysis > Quantile regression

### iqreg

Statistics > Nonparametric analysis > Interquantile regression

### sqreg

Statistics > Nonparametric analysis > Simultaneous-quantile regression

### bsqreg

Statistics > Nonparametric analysis > Bootstrapped quantile regression

## Description

**qreg** fits quantile (including median) regression models, also known as least–absolute-value models (LAV or MAD) and minimum L1-norm models.

**iqreg** estimates interquantile range regressions, regressions of the difference in quantiles. The estimated variance–covariance matrix of the estimators (VCE) is obtained via bootstrapping.

**sqreg** estimates simultaneous-quantile regression. It produces the same coefficients as **qreg** for each quantile. Reported standard errors will be similar, but **sqreg** obtains an estimate of the VCE via bootstrapping, and the VCE includes between-quantile blocks. Thus you can test and construct confidence intervals comparing coefficients describing different quantiles.

**bsqreg** is equivalent to **sqreg** with one quantile.

**\_qreg** is the internal estimation command for quantile regression. **\_qreg** is not intended to be used directly; see [Methods and formulas](#) below.

## Options for qreg

### Model

**quantile(#)** specifies the quantile to be estimated and should be a number between 0 and 1, exclusive. Numbers larger than 1 are interpreted as percentages. The default value of 0.5 corresponds to the median.

### Reporting

**level(#)**; see [\[R\] estimation options](#).

### Optimization

**optimization\_options**: **iterate(#)**, **[no]log**, **trace**. **iterate()** specifies the maximum number of iterations; **log/nolog** specifies whether to show the iteration log; and **trace** specifies that the iteration log should include the current parameter vector. These options are seldom used.

**wlsiter(#)** specifies the number of weighted least-squares iterations that will be attempted before the linear programming iterations are started. The default value is 1. If there are convergence problems, increasing this number should help.

## Options for iqreg

### Model

**quantiles(# #)** specifies the quantiles to be compared. The first number must be less than the second, and both should be between 0 and 1, exclusive. Numbers larger than 1 are interpreted as percentages. Not specifying this option is equivalent to specifying **quantiles(.25 .75)**, meaning the interquartile range.

**reps(#)** specifies the number of bootstrap replications to be used to obtain an estimate of the variance–covariance matrix of the estimators (standard errors). **reps(20)** is the default and is arguably too small. **reps(100)** would perform 100 bootstrap replications. **reps(1000)** would perform 1,000 replications.

### Reporting

**level(#)**; see [\[R\] estimation options](#).

**nodots** suppresses display of the replication dots.

## Options for `sqreg`

### Model

`quantiles(# [ # [ # ... ] ])` specifies the quantiles to be estimated and should contain numbers between 0 and 1, exclusive. Numbers larger than 1 are interpreted as percentages. The default value of 0.5 corresponds to the median.

`reps(#)` specifies the number of bootstrap replications to be used to obtain an estimate of the variance–covariance matrix of the estimators (standard errors). `reps(20)` is the default and is arguably too small. `reps(100)` would perform 100 bootstrap replications. `reps(1000)` would perform 1,000 replications.

### Reporting

`level(#)`; see [\[R\] estimation options](#).

`nodots` suppresses display of the replication dots.

## Options for `bsqreg`

### Model

`quantile(#)` specifies the quantile to be estimated and should be a number between 0 and 1, exclusive. Numbers larger than 1 are interpreted as percentages. The default value of 0.5 corresponds to the median.

`reps(#)` specifies the number of bootstrap replications to be used to obtain an estimate of the variance–covariance matrix of the estimators (standard errors). `reps(20)` is the default and is arguably too small. `reps(100)` would perform 100 bootstrap replications. `reps(1000)` would perform 1,000 replications.

### Reporting

`level(#)`; see [\[R\] estimation options](#).

## Options for `_qreg`

`quantile(#)` specifies the quantile to be estimated and should be a number between 0 and 1, exclusive. The default value of 0.5 corresponds to the median.

`level(#)`; see [\[R\] estimation options](#).

`accuracy(#)` should not be specified; it specifies the relative accuracy required for the linear programming algorithm. If the potential for improving the sum of weighted deviations by deleting an observation from the basis is less than this on a percentage basis, the algorithm will be said to have converged. The default value is  $10^{-10}$ .

*optimization\_options*: `iterate(#)`, `[no]log`, `trace`. `iterate()` specifies the maximum number of iterations; `log/nolog` specifies whether to show the iteration log; and `trace` specifies that the iteration log should include the current parameter vector. These options are seldom used.

## Remarks

Remarks are presented under the following headings:

- [Median regression](#)
- [Generalized quantile regression](#)
- [Estimated standard errors](#)
- [Interquantile and simultaneous-quantile regression](#)

## Median regression

`qreg` without options fits quantile regression models. The most common form is median regression, where the object is to estimate the median of the dependent variable, conditional on the values of the independent variables. This method is similar to ordinary regression, where the objective is to estimate the mean of the dependent variable. Simply put, median regression finds a line through the data that minimizes the sum of the *absolute* residuals rather than the sum of the *squares* of the residuals, as in ordinary regression. [Cameron and Trivedi \(2009, chap. 7\)](#) provide a nice introduction to quantile regression using Stata.

### ► Example 1

Consider a two-group experimental design with 5 observations per group:

```
. use http://www.stata-press.com/data/r11/twogrp
. list
```

|     | x | y  |
|-----|---|----|
| 1.  | 0 | 0  |
| 2.  | 0 | 1  |
| 3.  | 0 | 3  |
| 4.  | 0 | 4  |
| 5.  | 0 | 95 |
| 6.  | 1 | 14 |
| 7.  | 1 | 19 |
| 8.  | 1 | 20 |
| 9.  | 1 | 22 |
| 10. | 1 | 23 |

```
. qreg y x
Iteration 1: WLS sum of weighted deviations = 121.88268
Iteration 1: sum of abs. weighted deviations = 111
Iteration 2: sum of abs. weighted deviations = 110
Median regression
Raw sum of deviations 157 (about 14)
Min sum of deviations 110
Number of obs = 10
Pseudo R2 = 0.2994
```

| y     | Coef. | Std. Err. | t    | P> t  | [95% Conf. Interval] |         |
|-------|-------|-----------|------|-------|----------------------|---------|
| x     | 17    | 3.924233  | 4.33 | 0.003 | 7.950702             | 26.0493 |
| _cons | 3     | 2.774852  | 1.08 | 0.311 | -3.39882             | 9.39882 |



We have estimated the equation

$$y_{\text{median}} = 3 + 17x$$

We look back at our data.  $x$  takes on the values 0 and 1, so the median for the  $x = 0$  group is 3, whereas for  $x = 1$  it is  $3 + 17 = 20$ . The output reports that the raw sum of absolute deviations about 14 is 157; that is, the sum of  $|y - 14|$  is 157. Fourteen is the unconditional median of  $y$ , although in these data, any value between 14 and 19 could also be considered an unconditional median (we have an even number of observations, so the median is bracketed by those two values). In any case, the raw sum of deviations of  $y$  about the median would be the same no matter what number we choose between 14 and 19. (With a “median” of 14, the raw sum of deviations is 157. Now think of choosing a slightly larger number for the median and recalculating the sum. Half the observations will have larger negative residuals, but the other half will have smaller positive residuals, resulting in no net change.)

We turn now to the actual estimated equation. The sum of the absolute deviations about the solution  $y_{\text{median}} = 3 + 17x$  is 110. The pseudo- $R^2$  is calculated as  $1 - 110/157 \approx 0.2994$ . This result is based on the idea that the median regression is the maximum likelihood estimate for the double-exponential distribution.



□ Technical note

`qreg` is an alternative to regular regression or robust regression—see [R] [regress](#) and [R] [rreg](#). Let’s compare the results:

|               |        |           |            |                         |                      |          |
|---------------|--------|-----------|------------|-------------------------|----------------------|----------|
| . regress y x |        |           |            |                         |                      |          |
| Source        | SS     | df        | MS         | Number of obs = 10      |                      |          |
| Model         | 2.5    | 1         | 2.5        | F( 1, 8) = 0.00         |                      |          |
| Residual      | 6978.4 | 8         | 872.3      | Prob > F = 0.9586       |                      |          |
| Total         | 6980.9 | 9         | 775.655556 | R-squared = 0.0004      |                      |          |
|               |        |           |            | Adj R-squared = -0.1246 |                      |          |
|               |        |           |            | Root MSE = 29.535       |                      |          |
| y             | Coef.  | Std. Err. | t          | P> t                    | [95% Conf. Interval] |          |
| x             | -1     | 18.6794   | -0.05      | 0.959                   | -44.07477            | 42.07477 |
| _cons         | 20.6   | 13.20833  | 1.56       | 0.157                   | -9.858465            | 51.05847 |

Unlike `qreg`, `regress` fits ordinary linear regression and is concerned with predicting the mean rather than the median, so both results are, in a technical sense, correct. Putting aside those technicalities, however, we tend to use either regression to describe the central tendency of the data, of which the mean is one measure and the median another. Thus we can ask, “which method better describes the central tendency of these data?”

Means—and therefore ordinary linear regression—are sensitive to outliers, and our data were purposely designed to contain two such outliers: 95 for  $x = 0$  and 14 for  $x = 1$ . These two outliers dominated the ordinary regression and produced results that do not reflect the central tendency well—you are invited to enter the data and graph  $y$  against  $x$ .

Robust regression attempts to correct the outlier-sensitivity deficiency in ordinary regression:

```
. rreg y x, genwt(wt)
 Huber iteration 1: maximum difference in weights = .7311828
 Huber iteration 2: maximum difference in weights = .17695779
 Huber iteration 3: maximum difference in weights = .03149585
Biweight iteration 4: maximum difference in weights = .1979335
Biweight iteration 5: maximum difference in weights = .23332905
Biweight iteration 6: maximum difference in weights = .09960067
Biweight iteration 7: maximum difference in weights = .02691458
Biweight iteration 8: maximum difference in weights = .0009113
Robust regression Number of obs = 10
 F(1, 8) = 80.63
 Prob > F = 0.0000
```

| y     | Coef.    | Std. Err. | t    | P> t  | [95% Conf. Interval] |          |
|-------|----------|-----------|------|-------|----------------------|----------|
| x     | 18.16597 | 2.023114  | 8.98 | 0.000 | 13.50066             | 22.83128 |
| _cons | 2.000003 | 1.430558  | 1.40 | 0.200 | -1.298869            | 5.298875 |

Here `rreg` discarded the first outlier completely. (We know this because we included the `genwt()` option on `rreg` and, after fitting the robust regression, examined the weights.) For the other “outlier”, `rreg` produced a weight of 0.47.

In any case, the answers produced by `qreg` and `rreg` to describe the central tendency are similar, but the standard errors are different. In general, robust regression will have smaller standard errors because it is not as sensitive to the exact placement of observations near the median. Also, some authors (Rousseeuw and Leroy 1987, 11) have noted that quantile regression, unlike the median, may be sensitive to even one outlier, if its leverage is high enough. □

➤ **Example 2**

Let’s now consider a less artificial example using the automobile data described in [U] [1.2.2 Example datasets](#). Using median regression, we will regress each car’s price on its weight and length and whether it is of foreign manufacture:

```
. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)

. qreg price weight length foreign
Iteration 1: WLS sum of weighted deviations = 112795.66
Iteration 1: sum of abs. weighted deviations = 111901
Iteration 2: sum of abs. weighted deviations = 110529.43
(output omitted)
Iteration 8: sum of abs. weighted deviations = 108822.59
Median regression Number of obs = 74
Raw sum of deviations 142205 (about 4934)
Min sum of deviations 108822.6 Pseudo R2 = 0.2347
```

| price   | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|---------|-----------|-----------|-------|-------|----------------------|----------|
| weight  | 3.933588  | .8602183  | 4.57  | 0.000 | 2.217937             | 5.649239 |
| length  | -41.25191 | 28.8693   | -1.43 | 0.157 | -98.82991            | 16.32609 |
| foreign | 3377.771  | 577.3391  | 5.85  | 0.000 | 2226.305             | 4529.237 |
| _cons   | 344.6494  | 3260.244  | 0.11  | 0.916 | -6157.702            | 6847.001 |

The estimated equation is

$$\text{price}_{\text{median}} = 3.93 \text{ weight} - 41.25 \text{ length} + 3377.8 \text{ foreign} + 344.65$$

The output may be interpreted in the same way as linear regression output; see [R] [regress](#). The variables `weight` and `foreign` are significant, but `length` is not significant. The median price of the cars in these data is \$4,934. This value is a median (one of the two center observations), not *the* median, which would typically be defined as the midpoint of the two center observations.

◀

## Generalized quantile regression

Generalized quantile regression is similar to median regression in that it estimates an equation describing a quantile other than the 0.5 (median) quantile. For example, specifying `quant(.25)` estimates the 25th percentile or the first quartile.

### ► Example 3

Again we will begin with the 10-observation artificial dataset we used at the beginning of [Median regression](#) above. We will estimate the 0.6667 quantile:

```
. use http://www.stata-press.com/data/r11/twogrp
. qreg y x, quant(0.6667)
Iteration 1: WLS sum of weighted deviations = 152.32472
Iteration 1: sum of abs. weighted deviations = 138.0054
Iteration 2: sum of abs. weighted deviations = 136.6714
.6667 Quantile regression Number of obs = 10
Raw sum of deviations 159.3334 (about 20)
Min sum of deviations 136.6714 Pseudo R2 = 0.1422
```

|  | y     | Coef. | Std. Err. | t    | P> t  | [95% Conf. Interval] |          |
|--|-------|-------|-----------|------|-------|----------------------|----------|
|  | x     | 18    | 54.21918  | 0.33 | 0.748 | -107.0297            | 143.0297 |
|  | _cons | 4     | 38.33875  | 0.10 | 0.919 | -84.40932            | 92.40932 |

The 0.6667 quantile in the data is 20. The estimated values are 4 for  $x = 0$  and 22 for  $x = 1$ . These values are appropriate because the usual convention is to “count in”  $(n + 1) \times$  quantile observations.

◀

### ► Example 4

Returning to real data, the equation for the 25th percentile of `price` based on `weight`, `length`, and `foreign` in our automobile data is

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. qreg price weight length foreign, quant(.25)
Iteration 1: WLS sum of weighted deviations = 98938.466
Iteration 1: sum of abs. weighted deviations = 99457.766
Iteration 2: sum of abs. weighted deviations = 91339.779
(output omitted)
Iteration 10: sum of abs. weighted deviations = 69603.554
```

| .25 Quantile regression                    |           |           |       |       | Number of obs =      | 74       |
|--------------------------------------------|-----------|-----------|-------|-------|----------------------|----------|
| Raw sum of deviations 83825.5 (about 4187) |           |           |       |       | Pseudo R2            | = 0.1697 |
| Min sum of deviations 69603.55             |           |           |       |       |                      |          |
| price                                      | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
| weight                                     | 1.831789  | .668093   | 2.74  | 0.008 | .4993194             | 3.164258 |
| length                                     | 2.845558  | 24.78057  | 0.11  | 0.909 | -46.57773            | 52.26885 |
| foreign                                    | 2209.925  | 434.631   | 5.08  | 0.000 | 1343.081             | 3076.769 |
| _cons                                      | -1879.775 | 2808.067  | -0.67 | 0.505 | -7480.287            | 3720.737 |

Compared with our previous median regression, the coefficient on `length` now has a positive sign, and the coefficients on `foreign` and `weight` are reduced. The actual lower quantile is \$4,187, substantially less than the median \$4,934. It appears that the factors are weaker in this part of the distribution.

We can also estimate the upper quantile as a function of the same three variables:

| . qreg price weight length foreign, quant(.75)           |           |           |       |       |                      |           |
|----------------------------------------------------------|-----------|-----------|-------|-------|----------------------|-----------|
| Iteration 1: WLS sum of weighted deviations = 110931.48  |           |           |       |       |                      |           |
| Iteration 1: sum of abs. weighted deviations = 111305.91 |           |           |       |       |                      |           |
| Iteration 2: sum of abs. weighted deviations = 105989.57 |           |           |       |       |                      |           |
| (output omitted)                                         |           |           |       |       |                      |           |
| Iteration 7: sum of abs. weighted deviations = 98395.935 |           |           |       |       |                      |           |
| .75 Quantile regression                                  |           |           |       |       | Number of obs =      | 74        |
| Raw sum of deviations 159721.5 (about 6342)              |           |           |       |       | Pseudo R2            | = 0.3840  |
| Min sum of deviations 98395.94                           |           |           |       |       |                      |           |
| price                                                    | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
| weight                                                   | 9.22291   | 2.653579  | 3.48  | 0.001 | 3.930515             | 14.51531  |
| length                                                   | -220.7833 | 80.13907  | -2.76 | 0.007 | -380.6156            | -60.95096 |
| foreign                                                  | 3595.133  | 1727.704  | 2.08  | 0.041 | 149.3355             | 7040.931  |
| _cons                                                    | 20242.9   | 8534.529  | 2.37  | 0.020 | 3221.323             | 37264.49  |

This result tells a different story: `weight` is much more important, and `length` is now significant—with a negative coefficient! The prices of high-priced cars seem to be determined by factors different from those affecting the prices of low-priced cars.

□ **Technical note**

One explanation for having substantially different regression functions for different quantiles is that the data are heteroskedastic, as we will demonstrate below. The following statements create a sharply heteroskedastic set of data:

```
. drop _all
. set obs 10000
obs was 0, now 10000
. set seed 50550
. gen x = .1 + .9 * runiform()
. gen y = x * runiform()^2
```

Let's now fit the regressions for the 5th and 95th quantiles:

```
. qreg y x, quant(.05)
Iteration 1: WLS sum of weighted deviations = 1080.7273
Iteration 1: sum of abs. weighted deviations = 1078.3192
Iteration 2: sum of abs. weighted deviations = 282.73545
(output omitted)
Iteration 9: sum of abs. weighted deviations = 182.25244
.05 Quantile regression Number of obs = 10000
Raw sum of deviations 182.357 (about .0009234)
Min sum of deviations 182.2524 Pseudo R2 = 0.0006
```

| y     | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|-------|-----------|-----------|-------|-------|----------------------|----------|
| x     | .002601   | .0002737  | 9.50  | 0.000 | .0020646             | .0031374 |
| _cons | -.0001393 | .0001666  | -0.84 | 0.403 | -.000466             | .0001874 |

```
. qreg y x, quant(.95)
Iteration 1: WLS sum of weighted deviations = 1237.5569
Iteration 1: sum of abs. weighted deviations = 1238.0014
Iteration 2: sum of abs. weighted deviations = 456.65044
(output omitted)
Iteration 5: sum of abs. weighted deviations = 338.4389
.95 Quantile regression Number of obs = 10000
Raw sum of deviations 554.6889 (about .61326343)
Min sum of deviations 338.4389 Pseudo R2 = 0.3899
```

| y     | Coef.    | Std. Err. | t      | P> t  | [95% Conf. Interval] |          |
|-------|----------|-----------|--------|-------|----------------------|----------|
| x     | .8898259 | .0060398  | 147.33 | 0.000 | .8779867             | .901665  |
| _cons | .0021514 | .0036623  | 0.59   | 0.557 | -.0050275            | .0093302 |

The coefficient on  $x$ , in particular, differs markedly between the two estimates. For the mathematically inclined, it is not too difficult to show that the theoretical lines are  $y = 0.0025x$  for the 5th percentile and  $y = 0.9025x$  for the 95th, numbers in close agreement with our numerical results.



## Estimated standard errors

`qreg` estimates the variance–covariance matrix of the coefficients by using a method of [Koenker and Bassett Jr. \(1982\)](#) and [Rogers \(1993\)](#). This approach is described in [Methods and formulas](#) below. [Rogers \(1992\)](#) reports that, although this method seems adequate for homoskedastic errors, it appears to understate the standard errors for heteroskedastic errors. The irony is that exploring heteroskedastic errors is one of the major benefits of quantile regression. [Gould \(1992, 1997b\)](#) introduced generalized versions of `qreg` that obtain estimates of the standard errors by using bootstrap resampling (see [Efron and Tibshirani \[1993\]](#) or [Wu \[1986\]](#) for an introduction to bootstrap standard errors). The `iqreg`, `sqreg`, and `bsqreg` commands provide a bootstrapped estimate of the entire variance–covariance matrix of the estimators.

## ► Example 5

The first example of `qreg` on real data above was a median regression of `price` on `weight`, `length`, and `foreign` using the automobile data. Here is the result of repeating the estimation using bootstrap standard errors:

```
. set seed 1001
. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)
. bsqreg price weight length foreign
(fitting base model)
(bootstrapping)
```

|                                           |                 |        |
|-------------------------------------------|-----------------|--------|
| Median regression, bootstrap(20) SEs      | Number of obs = | 74     |
| Raw sum of deviations 142205 (about 4934) |                 |        |
| Min sum of deviations 108822.6            | Pseudo R2 =     | 0.2347 |

| price   | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|---------|-----------|-----------|-------|-------|----------------------|----------|
| weight  | 3.933588  | 3.12446   | 1.26  | 0.212 | -2.297951            | 10.16513 |
| length  | -41.25191 | 83.71266  | -0.49 | 0.624 | -208.2116            | 125.7077 |
| foreign | 3377.771  | 1057.281  | 3.19  | 0.002 | 1269.09              | 5486.452 |
| _cons   | 344.6494  | 7053.301  | 0.05  | 0.961 | -13722.72            | 14412.01 |

The coefficient estimates are the same—indeed, they are obtained using the same technique. Only the standard errors differ. Therefore, the *t* statistics, significance levels, and confidence intervals also differ.

Because `bsqreg` (as well as `sqreg` and `iqreg`) obtains standard errors by randomly resampling the data, the standard errors it produces will not be the same from run to run, unless we first set the random-number seed to the same number; see [R] [set seed](#).

By default, `bsqreg`, `sqreg`, and `iqreg` use 20 replications. We can control the number of replications by specifying the `reps()` option:

```
. bsqreg price weight length foreign, reps(1000)
(fitting base model)
(bootstrapping(output omitted)...)
Median regression, bootstrap(1000) SEs Number of obs = 74
Raw sum of deviations 142205 (about 4934)
Min sum of deviations 108822.6 Pseudo R2 = 0.2347
```

| price   | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|---------|-----------|-----------|-------|-------|----------------------|----------|
| weight  | 3.933588  | 2.659381  | 1.48  | 0.144 | -1.370379            | 9.237555 |
| length  | -41.25191 | 69.29771  | -0.60 | 0.554 | -179.4618            | 96.95802 |
| foreign | 3377.771  | 1094.947  | 3.08  | 0.003 | 1193.967             | 5561.575 |
| _cons   | 344.6494  | 5916.906  | 0.06  | 0.954 | -11456.25            | 12145.55 |

A comparison of the standard errors is informative:

| variable | qreg  | bsqreg<br>reps(20) | bsqreg<br>reps(1000) |
|----------|-------|--------------------|----------------------|
| weight   | .8602 | 3.124              | 2.670                |
| length   | 28.87 | 83.71              | 69.65                |
| foreign  | 577.3 | 1057.              | 1094.                |
| _cons    | 3260. | 7053.              | 5945.                |

The results shown above are typical for models with heteroskedastic errors. (Our dependent variable is `price`; if our model had been in terms of `ln(price)`, the standard errors estimated by `qreg` and `bsqreg` would have been nearly identical.) Also, even for heteroskedastic errors, 20 replications is generally sufficient for hypothesis tests against 0.

## Interquantile and simultaneous-quantile regression

Consider a quantile-regression model where the  $q$ th quantile is given by

$$Q_q(y) = a_q + b_{q,1}x_1 + b_{q,2}x_2$$

For instance, the 75th and 25th quantiles are given by

$$Q_{0.75}(y) = a_{0.75} + b_{0.75,1}x_1 + b_{0.75,2}x_2$$

$$Q_{0.25}(y) = a_{0.25} + b_{0.25,1}x_1 + b_{0.25,2}x_2$$

The difference in the quantiles is then

$$Q_{0.75}(y) - Q_{0.25}(y) = (a_{0.75} - a_{0.25}) + (b_{0.75,1} - b_{0.25,1})x_1 + (b_{0.75,2} - b_{0.25,2})x_2$$

**qreg** fits models such as  $Q_{0.75}(y)$  and  $Q_{0.25}(y)$ . **iqreg** fits interquantile models, such as  $Q_{0.75}(y) - Q_{0.25}(y)$ . The relationships of the coefficients estimated by **qreg** and **iqreg** are exactly as shown: **iqreg** reports coefficients that are the difference in coefficients of two **qreg** models, and, of course, **iqreg** reports the appropriate standard errors, which it obtains by bootstrapping.

**sqreg** is like **qreg** in that it estimates the equations for the quantiles

$$Q_{0.75}(y) = a_{0.75} + b_{0.75,1}x_1 + b_{0.75,2}x_2$$

$$Q_{0.25}(y) = a_{0.25} + b_{0.25,1}x_1 + b_{0.25,2}x_2$$

The coefficients it obtains are the same that would be obtained by estimating each equation separately using **qreg**. **sqreg** differs from **qreg** in that it estimates the equations simultaneously and obtains an estimate of the entire variance–covariance matrix of the estimators by bootstrapping. Thus you can perform hypothesis tests concerning coefficients both within and across equations.

For example, to fit the above model, you could type

```
. qreg y x1 x2, q(.25)
. qreg y x1 x2, q(.75)
```

Doing this, you would obtain estimates of the parameters, but you could not test whether  $b_{0.25,1} = b_{0.75,1}$  or, equivalently,  $b_{0.75,1} - b_{0.25,1} = 0$ . If your interest really is in the difference of coefficients, you could type

```
. iqreg y x1 x2, q(.25 .75)
```

The “coefficients” reported would be the difference in quantile coefficients. You could also estimate both quantiles simultaneously and then test the equality of the coefficients:

```
. sqreg y x1 x2, q(.25 .75)
. test [q25]x1 = [q75]x1
```

Whether you use **iqreg** or **sqreg** makes no difference for this test. **sqreg**, however, because it estimates the quantiles simultaneously, allows you to test other hypotheses. **iqreg**, by focusing on quantile differences, presents results in a way that is easier to read.

Finally, **sqreg** can estimate quantiles singly,

```
. sqreg y x1 x2, q(.5)
```

and can thereby be used as a substitute for the slower **bsqreg**. (Gould [1997b] presents timings demonstrating that **sqreg** is faster than **bsqreg**.) **sqreg** can also estimate more than two quantiles simultaneously:

```
. sqreg y x1 x2, q(.25 .5 .75)
```

➤ Example 6

In demonstrating `qreg`, we performed quantile regressions using the automobile data. We discovered that the regression of `price` on `weight`, `length`, and `foreign` produced vastly different coefficients for the 0.25, 0.5, and 0.75 quantile regressions. Here are the coefficients that we obtained:

| Variable | 25th percentile | 50th percentile | 75th percentile |
|----------|-----------------|-----------------|-----------------|
| weight   | 1.83            | 3.93            | 9.22            |
| length   | 2.85            | −41.25          | −220.8          |
| foreign  | 2209.9          | 3377.8          | 3595.1          |
| _cons    | −1879.8         | 344.6           | 20242.9         |

All we can say, having estimated these equations separately, is that `price` seems to depend differently on the `weight`, `length`, and `foreign` variables depending on the portion of the `price` distribution we examine. We cannot be more precise because the estimates have been made separately. With `sqreg`, however, we can estimate all the effects simultaneously:

```
. sqreg price weight length foreign, q(.25 .5 .75) reps(100)
(fitting base model)
(bootstrapping (output omitted))
```

Simultaneous quantile regression

bootstrap(100) SEs

Number of obs =

.25 Pseudo R2 =

.50 Pseudo R2 =

.75 Pseudo R2 =

74

0.1697

0.2347

0.3840

| price |         | Coef.     | Bootstrap Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|-------|---------|-----------|---------------------|-------|-------|----------------------|-----------|
| q25   | weight  | 1.831789  | 1.244947            | 1.47  | 0.146 | −.6511803            | 4.314758  |
|       | length  | 2.845558  | 27.91648            | 0.10  | 0.919 | −52.8321             | 58.52322  |
|       | foreign | 2209.925  | 911.6566            | 2.42  | 0.018 | 391.6836             | 4028.167  |
|       | _cons   | −1879.775 | 2756.871            | −0.68 | 0.498 | −7378.18             | 3618.63   |
|       |         |           |                     |       |       |                      |           |
| q50   | weight  | 3.933588  | 2.732408            | 1.44  | 0.154 | −1.516029            | 9.383205  |
|       | length  | −41.25191 | 75.8087             | −0.54 | 0.588 | −192.4476            | 109.9438  |
|       | foreign | 3377.771  | 921.578             | 3.67  | 0.000 | 1539.742             | 5215.8    |
|       | _cons   | 344.6494  | 6810.32             | 0.05  | 0.960 | −13238.1             | 13927.4   |
|       |         |           |                     |       |       |                      |           |
| q75   | weight  | 9.22291   | 2.732795            | 3.37  | 0.001 | 3.772523             | 14.6733   |
|       | length  | −220.7833 | 87.38042            | −2.53 | 0.014 | −395.058             | −46.50854 |
|       | foreign | 3595.133  | 1153.239            | 3.12  | 0.003 | 1295.07              | 5895.196  |
|       | _cons   | 20242.9   | 9000.697            | 2.25  | 0.028 | 2291.579             | 38194.23  |
|       |         |           |                     |       |       |                      |           |

The coefficient estimates above are the same as those previously estimated, although the standard error estimates are a little different. `sqreg` obtains estimates of variance by bootstrapping. [Rogers \(1992\)](#) provides evidence that, for quantile regression, the bootstrap standard errors are better than those calculated analytically by Stata.

The important thing here, however, is that the full covariance matrix of the estimators has been estimated and stored, and thus it is now possible to perform hypothesis tests. Are the effects of `weight` the same at the 25th and 75th percentiles?



```
. test [q25]weight = [q75]weight
(1) [q25]weight - [q75]weight = 0
 F(1, 70) = 8.29
 Prob > F = 0.0053
```

It appears that they are not. We can obtain a confidence interval for the difference by using `lincom`:

```
. lincom [q75]weight-[q25]weight
(1) - [q25]weight + [q75]weight = 0
```

| price | Coef.    | Std. Err. | t    | P> t  | [95% Conf. Interval] |          |
|-------|----------|-----------|------|-------|----------------------|----------|
| (1)   | 7.391121 | 2.567684  | 2.88 | 0.005 | 2.270036             | 12.51221 |

Indeed, we could test whether the `weight` and `length` sets of coefficients are equal at the three quantiles estimated:

```
. quietly test [q25]weight = [q50]weight
. quietly test [q25]weight = [q75]weight, accum
. quietly test [q25]length = [q50]length, accum
. test [q25]length = [q75]length, accum
(1) [q25]weight - [q50]weight = 0
(2) [q25]weight - [q75]weight = 0
(3) [q25]length - [q50]length = 0
(4) [q25]length - [q75]length = 0
 F(4, 70) = 2.21
 Prob > F = 0.0767
```

`iqreg` focuses on one quantile comparison but presents results that are more easily interpreted:

```
. set seed 1001
. iqreg price weight length foreign, q(.25 .75) reps(100) nodots
.75-.25 Interquantile regression Number of obs = 74
bootstrap(100) SEs .75 Pseudo R2 = 0.3840
 .25 Pseudo R2 = 0.1697
```

| price   | Coef.     | Bootstrap<br>Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|---------|-----------|------------------------|-------|-------|----------------------|-----------|
| weight  | 7.391121  | 2.467548               | 3.00  | 0.004 | 2.469752             | 12.31249  |
| length  | -223.6288 | 83.09868               | -2.69 | 0.009 | -389.3639            | -57.89376 |
| foreign | 1385.208  | 1193.557               | 1.16  | 0.250 | -995.2672            | 3765.683  |
| _cons   | 22122.68  | 9009.159               | 2.46  | 0.017 | 4154.478             | 40090.88  |

Looking only at the 0.25 and 0.75 quantiles (the interquartile range), the `iqreg` command output is easily interpreted. Increases in `weight` correspond significantly to increases in `price` dispersion. Increases in `length` correspond to decreases in `price` dispersion. The `foreign` variable does not significantly change `price` dispersion.

Do not make too much of these results; the purpose of this example is simply to illustrate the `sqreg` and `iqreg` commands and to do so in a context that suggests why analyzing dispersion might be of interest.

lincom after sqreg produced the same  $t$  statistic for the interquartile range of `weight`, as did the iqreg command above. In general, they will not agree exactly because of the randomness of bootstrapping, unless the random-number seed is set to the same value before estimation (as was done here).



Gould (1997a) presents simulation results showing that the coverage—the actual percentage of confidence intervals containing the true value—for iqreg is appropriate.

## Saved results

qreg saves the following in `e()`:

|                              |                                                               |
|------------------------------|---------------------------------------------------------------|
| Scalars                      |                                                               |
| <code>e(N)</code>            | number of observations                                        |
| <code>e(df_m)</code>         | model degrees of freedom                                      |
| <code>e(df_r)</code>         | residual degrees of freedom                                   |
| <code>e(q)</code>            | quantile requested                                            |
| <code>e(q_v)</code>          | value of the quantile                                         |
| <code>e(sum_adev)</code>     | sum of absolute deviations                                    |
| <code>e(sum_rdev)</code>     | sum of raw deviations                                         |
| <code>e(f_r)</code>          | residual density estimate                                     |
| <code>e(rank)</code>         | rank of <code>e(V)</code>                                     |
| <code>e(convcode)</code>     | 0 if converged; otherwise, return code for why nonconvergence |
| Macros                       |                                                               |
| <code>e(cmd)</code>          | <code>qreg</code>                                             |
| <code>e(cmdline)</code>      | command as typed                                              |
| <code>e(depvar)</code>       | name of dependent variable                                    |
| <code>e(properties)</code>   | <code>b V</code>                                              |
| <code>e(predict)</code>      | program used to implement <code>predict</code>                |
| <code>e(marginsnotok)</code> | predictions disallowed by <code>margins</code>                |
| Matrices                     |                                                               |
| <code>e(b)</code>            | coefficient vector                                            |
| <code>e(V)</code>            | variance–covariance matrix of the estimators                  |
| Functions                    |                                                               |
| <code>e(sample)</code>       | marks estimation sample                                       |

`iqreg` saves the following in `e()`:

#### Scalars

|                          |                                                               |
|--------------------------|---------------------------------------------------------------|
| <code>e(N)</code>        | number of observations                                        |
| <code>e(df_r)</code>     | residual degrees of freedom                                   |
| <code>e(q0)</code>       | lower quantile requested                                      |
| <code>e(q1)</code>       | upper quantile requested                                      |
| <code>e(reps)</code>     | number of replications                                        |
| <code>e(sumrdev0)</code> | lower quantile sum of raw deviations                          |
| <code>e(sumrdev1)</code> | upper quantile sum of raw deviations                          |
| <code>e(sumadev0)</code> | lower quantile sum of absolute deviations                     |
| <code>e(sumadev1)</code> | upper quantile sum of absolute deviations                     |
| <code>e(rank)</code>     | rank of <code>e(V)</code>                                     |
| <code>e(convcode)</code> | 0 if converged; otherwise, return code for why nonconvergence |

#### Macros

|                              |                                                |
|------------------------------|------------------------------------------------|
| <code>e(cmd)</code>          | <code>iqreg</code>                             |
| <code>e(cmdline)</code>      | command as typed                               |
| <code>e(depvar)</code>       | name of dependent variable                     |
| <code>e(vcetype)</code>      | title used to label Std. Err.                  |
| <code>e(properties)</code>   | <code>b V</code>                               |
| <code>e(predict)</code>      | program used to implement <code>predict</code> |
| <code>e(marginsnotok)</code> | predictions disallowed by <code>margins</code> |

#### Matrices

|                   |                                              |
|-------------------|----------------------------------------------|
| <code>e(b)</code> | coefficient vector                           |
| <code>e(V)</code> | variance–covariance matrix of the estimators |

#### Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

*(Continued on next page)*

**sqreg** saves the following in **e()**:

Scalars

|                    |                                                               |
|--------------------|---------------------------------------------------------------|
| <b>e(N)</b>        | number of observations                                        |
| <b>e(df_r)</b>     | residual degrees of freedom                                   |
| <b>e(n_q)</b>      | number of quantiles requested                                 |
| <b>e(q#)</b>       | the quantiles requested                                       |
| <b>e(reps)</b>     | number of replications                                        |
| <b>e(sumrdv#)</b>  | sum of raw deviations for q#                                  |
| <b>e(sumadv#)</b>  | sum of absolute deviations for q#                             |
| <b>e(rank)</b>     | rank of <b>e(V)</b>                                           |
| <b>e(convcode)</b> | 0 if converged; otherwise, return code for why nonconvergence |

Macros

|                        |                                          |
|------------------------|------------------------------------------|
| <b>e(cmd)</b>          | <b>sqreg</b>                             |
| <b>e(cmdline)</b>      | command as typed                         |
| <b>e(depvar)</b>       | name of dependent variable               |
| <b>e(eqnames)</b>      | names of equations                       |
| <b>e(vctype)</b>       | title used to label Std. Err.            |
| <b>e(properties)</b>   | <b>b V</b>                               |
| <b>e(predict)</b>      | program used to implement <b>predict</b> |
| <b>e(marginsnotok)</b> | predictions disallowed by <b>margins</b> |

Matrices

|             |                                              |
|-------------|----------------------------------------------|
| <b>e(b)</b> | coefficient vector                           |
| <b>e(V)</b> | variance–covariance matrix of the estimators |

Functions

|                  |                         |
|------------------|-------------------------|
| <b>e(sample)</b> | marks estimation sample |
|------------------|-------------------------|

**bsqreg** saves the following in **e()**:

Scalars

|                    |                                                               |
|--------------------|---------------------------------------------------------------|
| <b>e(N)</b>        | number of observations                                        |
| <b>e(df_r)</b>     | residual degrees of freedom                                   |
| <b>e(q)</b>        | quantile requested                                            |
| <b>e(q_v)</b>      | value of the quantile                                         |
| <b>e(reps)</b>     | number of replications                                        |
| <b>e(sum_adev)</b> | sum of absolute deviations                                    |
| <b>e(sum_rdev)</b> | sum of raw deviations                                         |
| <b>e(rank)</b>     | rank of <b>e(V)</b>                                           |
| <b>e(convcode)</b> | 0 if converged; otherwise, return code for why nonconvergence |

Macros

|                        |                                          |
|------------------------|------------------------------------------|
| <b>e(cmd)</b>          | <b>bsqreg</b>                            |
| <b>e(cmdline)</b>      | command as typed                         |
| <b>e(depvar)</b>       | name of dependent variable               |
| <b>e(properties)</b>   | <b>b V</b>                               |
| <b>e(predict)</b>      | program used to implement <b>predict</b> |
| <b>e(marginsnotok)</b> | predictions disallowed by <b>margins</b> |

Matrices

|             |                                              |
|-------------|----------------------------------------------|
| <b>e(b)</b> | coefficient vector                           |
| <b>e(V)</b> | variance–covariance matrix of the estimators |

Functions

|                  |                         |
|------------------|-------------------------|
| <b>e(sample)</b> | marks estimation sample |
|------------------|-------------------------|

`_qreg` saves the following in `r()`:

Scalars

|                          |                             |
|--------------------------|-----------------------------|
| <code>r(N)</code>        | number of observations      |
| <code>r(df_m)</code>     | model degrees of freedom    |
| <code>r(q)</code>        | quantile requested          |
| <code>r(q_v)</code>      | value of the quantile       |
| <code>r(sum_w)</code>    | sum of the weights          |
| <code>r(sum_adev)</code> | sum of absolute deviations  |
| <code>r(sum_rdev)</code> | sum of raw deviations       |
| <code>r(f_r)</code>      | residual density estimate   |
| <code>r(ic)</code>       | number of iterations        |
| <code>r(convcode)</code> | 1 if converged, 0 otherwise |

## Methods and formulas

`qreg`, `iqreg`, `sqreg`, and `bsqreg` are implemented as ado-files.

According to [Stuart and Ord \(1991, 1084\)](#), the method of minimum absolute deviations was first proposed by Boscovich in 1757 and was later developed by Laplace; [Stigler \(1986, 39–55\)](#) and [Hald \(1998, 97–103, 112–116\)](#) provide historical details. According to [Bloomfield and Steiger \(1980\)](#), [Harris \(1950\)](#) later observed that the problem of minimum absolute deviations could be turned into the linear programming problem that was first implemented by [Wagner \(1959\)](#). Interest has grown in this method because of interest in robust methods. Statistical and computational properties of minimum absolute deviation estimators are surveyed by [Narula and Wellington \(1982\)](#). [Hao and Naiman \(2007\)](#) provide an excellent introduction to quantile-regression methods.

Define  $q$  as the quantile to be estimated; the median is  $q = 0.5$ . For each observation  $i$ , let  $r_i$  be the residual

$$r_i = y_i - \sum_j \beta_j x_{ij}$$

Define the multiplier  $h_i$

$$h_i = \begin{cases} 2q & \text{if } r_i > 0 \\ 2(1 - q) & \text{otherwise} \end{cases}$$

The quantity being minimized with respect to  $\beta_j$  is  $\sum_i |r_i| h_i$ , so quantiles other than the median are estimated by weighting the residuals. For example, if we want to estimate the 75th percentile, we weight the negative residuals by 0.50 and the positive residuals by 1.50. It can be shown that the criterion is minimized when 75% of the residuals are negative.

This is set up as a linear programming problem and is solved via linear programming techniques, as suggested by [Armstrong, Frome, and Kung \(1979\)](#) and used by courtesy of Marcel Dekker, Inc. The definition of convergence is exact in the sense that no amount of added iterations could improve the solution. Each step is described by a set of observations through which the regression plane passes, called the *basis*. A step is taken by replacing a point in the basis if the sum of weighted absolute deviations can be improved. If this occurs, a line is printed in the iteration log. The linear programming method is started by doing a weighted least-squares (WLS) regression to identify a good set of observations to use as a starting basis. The WLS algorithm for  $q = 0.5$  is taken from [Schlossmacher \(1973\)](#) with a generalization for  $0 < q < 1$  implied from [Hunter and Lange \(2000\)](#).

The variances are estimated using a method suggested by [Koenker and Bassett Jr. \(1982\)](#). This method can be put into a form recommended by [Huber \(1967\)](#) for  $M$  estimates, where

$$\text{cov}(\beta) = \mathbf{R}_2^{-1} \mathbf{R}_1 \mathbf{R}_2^{-1}$$

$\mathbf{R}_1 = \mathbf{X}'\mathbf{W}\mathbf{W}'\mathbf{X}$  (in the Huber formulation),  $\mathbf{W}$  is a diagonal matrix with elements

$$W_{ii} = \begin{cases} q/f_{\text{residuals}}(0) & \text{if } r > 0 \\ (1 - q)/f_{\text{residuals}}(0) & \text{if } r < 0 \\ 0 & \text{otherwise} \end{cases}$$

and  $\mathbf{R}_2$  is the design matrix  $\mathbf{X}'\mathbf{X}$ . This is derived from formula 3.11 in Koenker and Bassett, although their notation is much different.  $f_{\text{residuals}}()$  refers to the density of the true residuals. Koenker and Bassett leave much unspecified, including how to obtain a density estimate for the errors in real data. At this point, we offer our contribution (Rogers 1993).

We first sort the residuals and locate the observation in the residuals corresponding to the quantile in question, taking into account weights if they are applied. We then calculate  $w_n$ , the square root of the sum of the weights. Unweighted data are equivalent to weighted data in which each observation has weight 1, resulting in  $w_n = \sqrt{n}$ . For analytically weighted data, the weights are rescaled so that the sum of the weights is the number of observations, resulting in  $\sqrt{n}$  again. For frequency-weighted data,  $w_n$  literally is the square root of the sum of the weights.

We locate the closest observation in each direction, such that the sum of weights for all closer observations is  $w_n$ . If we run off the end of the dataset, we stop. We calculate  $w_s$ , the sum of weights for all observations in this middle space. Typically,  $w_s$  is slightly greater than  $w_n$ .

If there are  $k$  parameters, then exactly  $k$  of the residuals must be zero. Thus we calculate an adjusted weight  $w_a = w_s - k$ . The density estimate is the distance spanned by these observations divided by  $w_a$ . Because the distance spanned by this mechanism converges toward zero, this estimate of density converges in probability to the true density.

The pseudo- $R^2$  is calculated as

$$1 - \frac{\text{sum of weighted deviations about estimated quantile}}{\text{sum of weighted deviations about raw quantile}}$$

This is based on the likelihood for a double-exponential distribution  $e^{h_i|r_i|}$ .

## References

- Angrist, J. D., and J.-S. Pischke. 2009. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton, NJ: Princeton University Press.
- Armstrong, R. D., E. L. Frome, and D. S. Kung. 1979. Algorithm 79-01: A revised simplex algorithm for the absolute deviation curve fitting problem. *Communications in Statistics, Simulation and Computation* 8: 175–190.
- Bloomfield, P., and W. Steiger. 1980. Least absolute deviations curve-fitting. *SIAM Journal on Scientific Computing* 1: 290–301.
- Cameron, A. C., and P. K. Trivedi. 2009. *Microeconometrics Using Stata*. College Station, TX: Stata Press.
- Efron, B., and R. J. Tibshirani. 1993. *An Introduction to the Bootstrap*. New York: Chapman & Hall/CRC.
- Gould, W. W. 1992. [sg11.1: Quantile regression with bootstrapped standard errors](#). *Stata Technical Bulletin* 9: 19–21. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 137–139. College Station, TX: Stata Press.
- . 1997a. [crc46: Better numerical derivatives and integrals](#). *Stata Technical Bulletin* 35: 3–5. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 8–12. College Station, TX: Stata Press.
- . 1997b. [sg70: Interquantile and simultaneous-quantile regression](#). *Stata Technical Bulletin* 38: 14–22. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 167–176. College Station, TX: Stata Press.
- Gould, W. W., and W. H. Rogers. 1994. Quantile regression as an alternative to robust regression. In *1994 Proceedings of the Statistical Computing Section*. Alexandria, VA: American Statistical Association.

- Hald, A. 1998. *A History of Mathematical Statistics from 1750 to 1930*. New York: Wiley.
- Hao, L., and D. Q. Naiman. 2007. *Quantile Regression*. Thousand Oaks, CA: Sage.
- Harris, T. 1950. Regression using minimum absolute deviations. *American Statistician* 4: 14–15.
- Huber, P. J. 1967. The behavior of maximum likelihood estimates under nonstandard conditions. In Vol. 1 of *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 221–233. Berkeley: University of California Press.
- . 1981. *Robust Statistics*. New York: Wiley.
- Hunter, D. R., and K. Lange. 2000. Quantile regression via an MM algorithm. *Journal of Computational and Graphical Statistics* 9: 60–77.
- Jolliffe, D., B. Krushelnysky, and A. Semykina. 2000. [sg153: Censored least absolute deviations estimator: CLAD](#). *Stata Technical Bulletin* 58: 13–16. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 240–244. College Station, TX: Stata Press.
- Koenker, R., and G. Bassett Jr. 1982. Robust tests for heteroscedasticity based on regression quantiles. *Econometrica* 50: 43–61.
- Koenker, R., and K. Hallock. 2001. Quantile regression. *Journal of Economic Perspectives* 15: 143–156.
- Narula, S. C., and J. F. Wellington. 1982. The minimum sum of absolute errors regression: A state of the art survey. *International Statistical Review* 50: 317–326.
- Rogers, W. H. 1992. [sg11: Quantile regression standard errors](#). *Stata Technical Bulletin* 9: 16–19. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 133–137. College Station, TX: Stata Press.
- . 1993. [sg11.2: Calculation of quantile regression standard errors](#). *Stata Technical Bulletin* 13: 18–19. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 77–78. College Station, TX: Stata Press.
- Rousseeuw, P. J., and A. M. Leroy. 1987. *Robust Regression and Outlier Detection*. New York: Wiley.
- Schlossmacher, E. J. 1973. An iterative technique for absolute deviations curve fitting. *Journal of the American Statistical Association* 68: 857–859.
- Stigler, S. M. 1986. *The History of Statistics: The Measurement of Uncertainty before 1900*. Cambridge, MA: Belknap Press.
- Stuart, A., and J. K. Ord. 1991. *Kendall's Advanced Theory of Statistics: Distribution Theory, Vol I*. 5th ed. New York: Oxford University Press.
- Wagner, H. M. 1959. Linear programming techniques for regression analysis. *Journal of the American Statistical Association* 54: 206–212.
- Wu, C. F. J. 1986. Jackknife, bootstrap and other resampling methods in regression analysis. *Annals of Statistics* 14: 1261–1350 (including discussions and rejoinder).

## Also see

- [R] [qreg postestimation](#) — Postestimation tools for qreg, iqreg, sqreg, and bsqreg
- [R] [bootstrap](#) — Bootstrap sampling and estimation
- [R] [regress](#) — Linear regression
- [R] [rreg](#) — Robust regression
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `qreg`, `iqreg`, `bsqreg`, and `sqreg`:

| command                | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>     | VCE and estimation sample summary                                                                   |
| <code>estimates</code> | cataloging estimation results                                                                       |
| <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>  | link test for model specification                                                                   |
| <code>margins</code>   | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>   | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code> | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details.

Syntax for predict

For `qreg`, `iqreg`, and `bsqreg`

```
predict [type] newvar [if] [in] [, [xb|stdp|residuals]]
```

For `sqreg`

```
predict [type] newvar [if] [in] [, equation(eqno[,eqno]) statistic]
```

| statistic              | description                                            |
|------------------------|--------------------------------------------------------|
| Main                   |                                                        |
| <code>xb</code>        | linear prediction; the default                         |
| <code>stdp</code>      | standard error of the linear prediction                |
| <code>stddp</code>     | standard error of the difference in linear predictions |
| <code>residuals</code> | residuals                                              |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.



## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

`xb`, the default, calculates the linear prediction.

`stdp` calculates the standard error of the linear prediction.

`stddp` is allowed only after you have fit a model using `sqreg`. The standard error of the difference in linear predictions ( $\mathbf{x}_{1j}\mathbf{b} - \mathbf{x}_{2j}\mathbf{b}$ ) between equations 1 and 2 is calculated.

`residuals` calculates the residuals, that is,  $y_j - \mathbf{x}_j\mathbf{b}$ .

`equation(eqno [, eqno])` specifies the equation to which you are making the calculation.

`equation()` is filled in with one *eqno* for the `xb`, `stdp`, and `residuals` options. `equation(#1)` would mean that the calculation is to be made for the first equation, `equation(#2)` would mean the second, and so on. You could also refer to the equations by their names. `equation(income)` would refer to the equation named `income` and `equation(hours)` to the equation named `hours`.

If you do not specify `equation()`, results are the same as if you had specified `equation(#1)`.

To use `stddp`, you must specify two equations. You might specify `equation(#1, #2)` or `equation(q80, q20)` to indicate the 80th and 20th quantiles.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [qreg](#) — Quantile regression

[U] [20 Estimation and postestimation commands](#)

# Title

query — Display system parameters

## Syntax

`query` [ `memory` | `output` | `interface` | `graphics` | `efficiency` | `network` |  
`update` | `trace` | `mata` | `other` ]

## Description

`query` displays the settings of various Stata parameters.

## Remarks

`query` provides more system information than you will ever want to know. You do not need to understand every line of output that `query` produces if all you need is one piece of information. Here is what happens when you type `query`:

```
. query
```

|                                                                           |        |                        |
|---------------------------------------------------------------------------|--------|------------------------|
| Memory settings (also see <b>query memory</b> for a more complete report) |        |                        |
| set memory                                                                | 50M    |                        |
| set maxvar                                                                | 5000   |                        |
| set matsize                                                               | 400    |                        |
| Output settings                                                           |        |                        |
| set more                                                                  | on     |                        |
| set rmsg                                                                  | off    |                        |
| set dp                                                                    | period | may be period or comma |
| set linesize                                                              | 79     | characters             |
| set pagesize                                                              | 23     | lines                  |
| set logtype smcl may be smcl or text                                      |        |                        |
| Interface settings                                                        |        |                        |
| set dockable                                                              | off    |                        |
| set dockingguides                                                         | on     |                        |
| set floatresults                                                          | off    |                        |
| set floatwindows                                                          | off    |                        |
| set locksplitters                                                         | off    |                        |
| set pinnable                                                              | off    |                        |
| set persistfv                                                             | off    |                        |
| set persistvtopic                                                         | off    |                        |
| set doublebuffer                                                          | on     |                        |
| set linegap                                                               | 1      | pixels                 |
| set scrollbufsize                                                         | 200000 | characters             |
| set reventries                                                            | 5000   | lines                  |
| set maxdb                                                                 | 50     | dialog boxes           |

|                                        |                                      |                                        |  |
|----------------------------------------|--------------------------------------|----------------------------------------|--|
| Graphics settings                      |                                      |                                        |  |
| set graphics                           | on                                   |                                        |  |
| set autotabgraphs                      | off                                  |                                        |  |
| set scheme                             | s2color                              |                                        |  |
| set printcolor                         | automatic                            | may be automatic, asis, gs1, gs2, gs3  |  |
| set copycolor                          | automatic                            | may be automatic, asis, gs1, gs2, gs3  |  |
| <hr/>                                  |                                      |                                        |  |
| Efficiency settings                    |                                      |                                        |  |
| set adosize                            | 1000                                 | kilobytes                              |  |
| set virtual                            | off                                  |                                        |  |
| <hr/>                                  |                                      |                                        |  |
| Network settings                       |                                      |                                        |  |
| set checksum                           | off                                  |                                        |  |
| set timeout1                           | 120                                  | seconds                                |  |
| set timeout2                           | 300                                  | seconds                                |  |
| <hr/>                                  |                                      |                                        |  |
| set httpproxy                          | off                                  |                                        |  |
| set httpproxyhost                      |                                      |                                        |  |
| set httpproxyport                      | 80                                   |                                        |  |
| <hr/>                                  |                                      |                                        |  |
| set httpproxyauth                      | off                                  |                                        |  |
| set httpproxyuser                      |                                      |                                        |  |
| set httpproxypw                        |                                      |                                        |  |
| <hr/>                                  |                                      |                                        |  |
| Update settings                        |                                      |                                        |  |
| set update_query                       | on                                   |                                        |  |
| set update_interval                    | 7                                    |                                        |  |
| set update_prompt                      | on                                   |                                        |  |
| <hr/>                                  |                                      |                                        |  |
| Trace (programming debugging) settings |                                      |                                        |  |
| set trace                              | off                                  |                                        |  |
| set tracedepth                         | 32000                                |                                        |  |
| set traceexpand                        | on                                   |                                        |  |
| set tracesep                           | on                                   |                                        |  |
| set traceindent                        | on                                   |                                        |  |
| set tracenumber                        | off                                  |                                        |  |
| set tracehilite                        |                                      |                                        |  |
| <hr/>                                  |                                      |                                        |  |
| Mata settings                          |                                      |                                        |  |
| set matastrict                         | off                                  |                                        |  |
| set matalnum                           | off                                  |                                        |  |
| set mataoptimize                       | on                                   |                                        |  |
| set matafavor                          | space                                | may be space or speed                  |  |
| set matacache                          | 400                                  | kilobytes                              |  |
| set matalibs                           | lmatatabase;lmataado;lmataopt        |                                        |  |
| set matamofirst                        | off                                  |                                        |  |
| <hr/>                                  |                                      |                                        |  |
| Other settings                         |                                      |                                        |  |
| set type                               | float                                | may be float or double                 |  |
| set level                              | 95                                   | percent confidence intervals           |  |
| set maxiter                            | 16000                                | max iterations for estimation commands |  |
| set searchdefault                      | local                                | may be local, net, or all              |  |
| set seed                               | X075bcd15f123bb5159a55e50022865746ad |                                        |  |
| set varabbrev                          | on                                   |                                        |  |
| set emptycells                         | keep                                 | may be keep or drop                    |  |
| set processors                         | 1                                    |                                        |  |

The output is broken into several divisions: memory, output, interface, graphics, efficiency, network, update, trace, mata, and other settings. We will discuss each one in turn.

We generated the output above using Stata/SE for Windows. Here is what happens when we type query and we are running Stata/IC for Linux(GUI):

. query

|                     |           |  |                                         |
|---------------------|-----------|--|-----------------------------------------|
| Memory settings     |           |  |                                         |
| set memory          | 10M       |  |                                         |
| set maxvar          | 2048      |  | (not settable in this version of Stata) |
| set matsize         | 400       |  |                                         |
| Output settings     |           |  |                                         |
| set more            | on        |  |                                         |
| set rmsg            | off       |  |                                         |
| set dp              | period    |  | may be period or comma                  |
| set linesize        | 79        |  | characters                              |
| set pagesize        | 23        |  | lines                                   |
| set logtype         | smcl      |  | may be smcl or text                     |
| Interface settings  |           |  |                                         |
| set linegap         | 1         |  | pixels                                  |
| set scrollbufsize   | 200000    |  | characters                              |
| set fastscroll      | on        |  |                                         |
| set varlabelpos     | 32        |  | column number                           |
| set reventries      | 5000      |  | lines                                   |
| set maxdb           | 50        |  | dialog boxes                            |
| Graphics settings   |           |  |                                         |
| set graphics        | on        |  |                                         |
| set scheme          | s2color   |  |                                         |
| set printcolor      | automatic |  | may be automatic, asis, gs1, gs2, gs3   |
| Efficiency settings |           |  |                                         |
| set adosize         | 1000      |  | kilobytes                               |
| set virtual         | off       |  |                                         |
| Network settings    |           |  |                                         |
| set checksum        | off       |  |                                         |
| set timeout1        | 120       |  | seconds                                 |
| set timeout2        | 300       |  | seconds                                 |
| set httpproxy       | off       |  |                                         |
| set httpproxyhost   |           |  |                                         |
| set httpproxyport   | 8080      |  |                                         |
| set httpproxyauth   | off       |  |                                         |
| set httpproxyuser   |           |  |                                         |
| set httpproxypw     |           |  |                                         |
| Update settings     |           |  |                                         |
| set update_query    |           |  | (not relevant)                          |
| set update_interval |           |  | (not relevant)                          |
| set update_prompt   |           |  | (not relevant)                          |

## Trace (programming debugging) settings

```

set trace off
set tracedepth 32000
set traceexpand on
set tracesep on
set traceindent on
set tracenumber off
set tracehilit

```

---

## Mata settings

```

set matastrict off
set matalnum off
set mataoptimize on
set matafavor space may be space or speed
set matacache 400 kilobytes
set matalibs lmatbase;lmatopt;lmatado
set matamofirst off

```

---

## Other settings

```

set type float may be float or double
set level 95 percent confidence intervals
set maxiter 16000 max iterations for estimation commands
set searchdefault local may be local, net, or all
set seed X075bcd15f123bb5159a55e50022865746ad
set varabbrev on
set emptycells keep may be keep or drop
set processors 1
set odbcmgr iodbc

```

---

## Memory settings

Memory settings indicate how much memory is allocated to Stata, the maximum number of variables, and the maximum size of a matrix.

For more information, see

```

memory [D] memory
maxvar [D] memory
matsize [R] matsize

```

## Output settings

Output settings show how Stata displays output on the screen and in log files.

For more information, see

```

more [R] more
rmmsg [P] rmmsg
dp [D] format
linesize [R] log
pagesize [R] more
logtype [R] log
eolchar [R] set
notifyuser [R] set
playsnd [R] set

```

## Interface settings

Interface settings control how Stata's interface works.

For more information, see

|               |                         |
|---------------|-------------------------|
| dockable      | [R] <a href="#">set</a> |
| dockingguides | [R] <a href="#">set</a> |
| floatresults  | [R] <a href="#">set</a> |
| floatwindows  | [R] <a href="#">set</a> |
| locksplitters | [R] <a href="#">set</a> |
| pinnable      | [R] <a href="#">set</a> |
| persistfv     | [R] <a href="#">set</a> |
| persistvtopic | [R] <a href="#">set</a> |
| doublebuffer  | [R] <a href="#">set</a> |
| smoothfonts   | [R] <a href="#">set</a> |
| linegap       | [R] <a href="#">set</a> |
| revkeyboard   | [R] <a href="#">set</a> |
| varkeyboard   | [R] <a href="#">set</a> |
| scrollbufsize | [R] <a href="#">set</a> |
| fastscroll    | [R] <a href="#">set</a> |
| varlabelpos   | [R] <a href="#">set</a> |
| reventries    | [R] <a href="#">set</a> |
| maxdb         | [R] <a href="#">db</a>  |

## Graphics settings

Graphics settings indicate how Stata's graphics are displayed.

For more information, see

|               |                                    |
|---------------|------------------------------------|
| graphics      | [G] <a href="#">set graphics</a>   |
| autotabgraphs | [R] <a href="#">set</a>            |
| scheme        | [G] <a href="#">set scheme</a>     |
| printcolor    | [G] <a href="#">set printcolor</a> |
| copycolor     | [G] <a href="#">set printcolor</a> |

## Efficiency settings

The efficiency settings set the maximum amount of memory allocated to automatically loaded do-files, the maximum number of remembered-contents dialog boxes, and the use of virtual memory.

For more information, see

|         |                            |
|---------|----------------------------|
| adosize | [P] <a href="#">sysdir</a> |
| virtual | [D] <a href="#">memory</a> |

## Network settings

Network settings determine how Stata interacts with the Internet.

For more information, see [\[R\] netio](#).

## Update settings

Update settings determine how Stata performs updates.

For more information, see [\[R\] update](#).

## Trace settings

Trace settings adjust Stata's behavior and are particularly useful in debugging code.

For more information, see [\[P\] trace](#).

## Mata settings

Mata settings affect Mata's system parameters.

For more information, see [\[M-3\] mata set](#).

## Other settings

The other settings are a miscellaneous collection.

For more information, see

|               |                              |
|---------------|------------------------------|
| type          | <a href="#">[D] generate</a> |
| level         | <a href="#">[R] level</a>    |
| maxiter       | <a href="#">[R] maximize</a> |
| searchdefault | <a href="#">[R] search</a>   |
| seed          | <a href="#">[R] set seed</a> |
| varabbrev     | <a href="#">[R] set</a>      |
| emptycells    | <a href="#">[R] set</a>      |
| processors    | <a href="#">[R] set</a>      |
| odbcmgr       | <a href="#">[D] odbc</a>     |

In general, the parameters displayed by `query` can be changed by `set`; see [\[R\] set](#).

## Also see

[\[R\] set](#) — Overview of system parameters

[\[P\] creturn](#) — Return c-class values

[\[M-3\] mata set](#) — Set and display Mata system parameters

Syntax

*Wilcoxon rank-sum test*

```
ranksum varname [if] [in] , by(groupvar) [porder]
```

*Nonparametric equality-of-medians test*

```
median varname [if] [in] [weight] , by(groupvar) [median_options]
```

| ranksum options        | description                                                                        |
|------------------------|------------------------------------------------------------------------------------|
| Main                   |                                                                                    |
| *by( <i>groupvar</i> ) | grouping variable                                                                  |
| porder                 | probability that variable for first group is larger than variable for second group |

| <i>median_options</i>  | description                                                     |
|------------------------|-----------------------------------------------------------------|
| Main                   |                                                                 |
| *by( <i>groupvar</i> ) | grouping variable                                               |
| exact                  | perform Fisher's exact test                                     |
| medianties(below)      | assign values equal to the median to below group                |
| medianties(above)      | assign values equal to the median to above group                |
| medianties(drop)       | drop values equal to the median from the analysis               |
| medianties(split)      | split values equal to the median equally between the two groups |

\*by(*groupvar*) is required.  
by is allowed with ranksum and median; see [D] by.  
fweights are allowed with median; see [U] 11.1.6 weight.

Menu

ranksum

Statistics > Nonparametric analysis > Tests of hypotheses > Wilcoxon rank-sum test

median

Statistics > Nonparametric analysis > Tests of hypotheses > K-sample equality-of-medians test

Description

ranksum tests the hypothesis that two independent samples (i.e., *unmatched* data) are from populations with the same distribution by using the Wilcoxon rank-sum test, which is also known as the Mann–Whitney two-sample statistic (Wilcoxon 1945; Mann and Whitney 1947).



`median` performs a nonparametric  $k$ -sample test on the equality of medians. It tests the null hypothesis that the  $k$  samples were drawn from populations with the same median. For two samples, the chi-squared test statistic is computed both with and without a continuity correction.

`ranksum` and `median` are for use with *unmatched* data. For equality tests on matched data, see [R] [signrank](#).

## Options for ranksum

### Main

`by(groupvar)` is required. It specifies the name of the grouping variable.

`porder` displays an estimate of the probability that a random draw from the first population is larger than a random draw from the second population.

## Options for median

### Main

`by(groupvar)` is required. It specifies the name of the grouping variable.

`exact` displays the significance calculated by Fisher's exact test. For two samples, both one- and two-sided probabilities are displayed.

`medianties(below|above|drop|split)` specifies how values equal to the overall median are to be handled. The median test computes the median for *varname* by using all observations and then divides the observations into those falling above the median and those falling below the median. When values for an observation are equal to the sample median, they can be dropped from the analysis by specifying `medianties(drop)`; added to the group above or below the median by specifying `medianties(above)` or `medianties(below)`, respectively; or if there is more than 1 observation with values equal to the median, they can be equally divided into the two groups by specifying `medianties(split)`. If this option is not specified, `medianties(below)` is assumed.

## Remarks

### ► Example 1

We are testing the effectiveness of a new fuel additive. We run an experiment with 24 cars: 12 cars with the fuel treatment and 12 cars without. We input these data by creating a dataset with 24 observations. `mpg` records the mileage rating, and `treat` records 0 if the mileage corresponds to untreated fuel and 1 if it corresponds to treated fuel.

```
. use http://www.stata-press.com/data/r11/fuel2
. ranksum mpg, by(treat)
```

Two-sample Wilcoxon rank-sum (Mann-Whitney) test

| treat    | obs | rank sum | expected |
|----------|-----|----------|----------|
| 0        | 12  | 128      | 150      |
| 1        | 12  | 172      | 150      |
| combined | 24  | 300      | 300      |

```
unadjusted variance 300.00
adjustment for ties -4.04

adjusted variance 295.96
Ho: mpg(treat==0) = mpg(treat==1)
 z = -1.279
 Prob > |z| = 0.2010
```

These results indicate that the medians are not statistically different at any level smaller than 20.1%. Similarly, the median test,

```
. median mpg, by(treat) exact
Median test
```

| Greater<br>than the<br>median | treat |    | Total |
|-------------------------------|-------|----|-------|
|                               | 0     | 1  |       |
| no                            | 7     | 5  | 12    |
| yes                           | 5     | 7  | 12    |
| Total                         | 12    | 12 | 24    |

```
 Pearson chi2(1) = 0.6667 Pr = 0.414
 Fisher's exact = 0.684
 1-sided Fisher's exact = 0.342
Continuity corrected:
 Pearson chi2(1) = 0.1667 Pr = 0.683
```

fails to reject the null hypothesis that there is no difference between the two fuel additives.

Compare these results from these two tests with those obtained from the `signrank` and `signtest` where we found significant differences; see [R] [signrank](#). An experiment run on 24 different cars is not as powerful as a before-and-after comparison using the same 12 cars.



## Saved results

`ranksum` saves the following in `r()`:

|                       |                   |                         |                                       |
|-----------------------|-------------------|-------------------------|---------------------------------------|
| Scalars               |                   |                         |                                       |
| <code>r(N_1)</code>   | sample size $n_1$ | <code>r(group1)</code>  | value of variable for first group     |
| <code>r(N_2)</code>   | sample size $n_2$ | <code>r(sum_obs)</code> | actual sum of ranks for first group   |
| <code>r(z)</code>     | $z$ statistic     | <code>r(sum_exp)</code> | expected sum of ranks for first group |
| <code>r(Var_a)</code> | adjusted variance |                         |                                       |

`median` saves the following in `r()`:

|                         |                                    |                          |                                         |
|-------------------------|------------------------------------|--------------------------|-----------------------------------------|
| Scalars                 |                                    |                          |                                         |
| <code>r(N)</code>       | sample size                        | <code>r(groups)</code>   | number of groups compared               |
| <code>r(chi2)</code>    | Pearson's $\chi^2$                 | <code>r(chi2_cc)</code>  | continuity-corrected Pearson's $\chi^2$ |
| <code>r(p)</code>       | significance of Pearson's $\chi^2$ | <code>r(p_cc)</code>     | continuity-corrected significance       |
| <code>r(p_exact)</code> | Fisher's exact $p$                 | <code>r(p1_exact)</code> | one-sided Fisher's exact $p$            |

## Methods and formulas

`ranksum` and `median` are implemented as ado-files.

For a practical introduction to these techniques with an emphasis on examples rather than theory, see [Acock \(2008\)](#), [Bland \(2000\)](#), or [Sprent and Smeeton \(2007\)](#). For a summary of these tests, see [Snedecor and Cochran \(1989\)](#).

Methods and formulas are presented under the following headings:

[ranksum](#)  
[median](#)

### ranksum

For the Wilcoxon rank-sum test, there are two independent random variables,  $X_1$  and  $X_2$ , and we test the null hypothesis that  $X_1 \sim X_2$ . We have a sample of size  $n_1$  from  $X_1$  and another of size  $n_2$  from  $X_2$ .

The data are then ranked without regard to the sample to which they belong. If the data are tied, averaged ranks are used. Wilcoxon's test statistic ([1945](#)) is the sum of the ranks for the observations in the first sample:

$$T = \sum_{i=1}^{n_1} R_{1i}$$

Mann and Whitney's  $U$  statistic ([1947](#)) is the number of pairs  $(X_{1i}, X_{2j})$  such that  $X_{1i} > X_{2j}$ . These statistics differ only by a constant:

$$U = T - \frac{n_1(n_1 + 1)}{2}$$

Again Fisher's principle of randomization provides a method for calculating the distribution of the test statistic, ties or not. The randomization distribution consists of the  $\binom{n}{n_1}$  ways to choose  $n_1$  ranks from the set of all  $n = n_1 + n_2$  ranks and assign them to the first sample.

It is a straightforward exercise to verify that

$$E(T) = \frac{n_1(n+1)}{2} \quad \text{and} \quad \text{Var}(T) = \frac{n_1 n_2 s^2}{n}$$

where  $s$  is the standard deviation of the combined ranks,  $r_i$ , for both groups:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (r_i - \bar{r})^2$$

This formula for the variance is exact and holds both when there are no ties and when there are ties and we use averaged ranks. (Indeed, the variance formula holds for the randomization distribution of choosing  $n_1$  numbers from any set of  $n$  numbers.)

Using a normal approximation, we calculate

$$z = \frac{T - E(T)}{\sqrt{\text{Var}(T)}}$$

When the `porder` option is specified, the probability

$$p = \frac{U}{n_1 n_2}$$

is computed.

## median

The median test examines whether it is likely that two or more samples came from populations with the same median. The null hypothesis is that the samples were drawn from populations with the same median. The alternative hypothesis is that at least one sample was drawn from a population with a different median. The test should be used only with ordinal or interval data.

Assume that there are score values for  $k$  independent samples to be compared. The median test is performed by first computing the median score for all observations combined, regardless of the sample group. Each score is compared with this computed grand median and is classified as being above the grand median, below the grand median, or equal to the grand median. Observations with scores equal to the grand median can be dropped, added to the “above” group, added to the “below” group, or split between the two groups.

Once all observations are classified, the data are cast into a  $2 \times k$  contingency table, and a Pearson’s chi-squared test or Fisher’s exact test is performed.

Henry Berthold Mann (1905–2000) was born in Vienna, where he completed a doctorate in algebraic number theory. He moved to the United States in 1938 and for several years earned a living by tutoring in New York. During this time, he proved a celebrated conjecture in number theory and studied statistics at Columbia with Abraham Wald, with whom he wrote three papers. After the war, he taught at Ohio State and the Universities of Wisconsin and Arizona. In addition to his work in number theory and statistics, he made major contributions to algebra and combinatorics.

Donald Ransom Whitney (1915–2007) studied at Oberlin, Princeton, and Ohio State Universities and worked at the latter throughout his career. His PhD thesis under Henry Mann was on nonparametric statistics and produced the test that bears his name.

## References

- Acock, A. C. 2008. *A Gentle Introduction to Stata*. 2nd ed. College Station, TX: Stata Press.
- Bland, M. 2000. *An Introduction to Medical Statistics*. 3rd ed. Oxford: Oxford University Press.
- Feiveson, A. H. 2002. [Power by simulation](#). *Stata Journal* 2: 107–124.
- Fisher, R. A. 1935. *The Design of Experiments*. Edinburgh: Oliver & Boyd.
- Goldstein, R. 1997. [sg69: Immediate Mann–Whitney and binomial effect-size display](#). *Stata Technical Bulletin* 36: 29–31. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 187–189. College Station, TX: Stata Press.
- Kruskal, W. H. 1957. Historical notes on the Wilcoxon unpaired two-sample test. *Journal of the American Statistical Association* 52: 356–360.
- Mann, H. B., and D. R. Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics* 18: 50–60.
- Newson, R. 2000a. [snp15: somersd—Confidence intervals for nonparametric statistics and their differences](#). *Stata Technical Bulletin* 55: 47–55. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 312–322. College Station, TX: Stata Press.
- . 2000b. [snp15.1: Update to somersd](#). *Stata Technical Bulletin* 57: 35. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 322–323. College Station, TX: Stata Press.
- . 2000c. [snp15.2: Update to somersd](#). *Stata Technical Bulletin* 58: 30. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, p. 323. College Station, TX: Stata Press.
- . 2001. [snp15.3: Update to somersd](#). *Stata Technical Bulletin* 61: 22. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, p. 324. College Station, TX: Stata Press.
- . 2003. [snp15\\_4: Software update for somersd](#). *Stata Journal* 3: 325.

- . 2005. [snp15\\_5](#): Software update for `somersd`. *Stata Journal* 5: 470.
- Perkins, A. M. 1998. [snp14](#): A two-sample multivariate nonparametric test. *Stata Technical Bulletin* 42: 47–49. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 243–245. College Station, TX: Stata Press.
- Snedecor, G. W., and W. G. Cochran. 1989. *Statistical Methods*. 8th ed. Ames, IA: Iowa State University Press.
- Sprent, P., and N. C. Smeeton. 2007. *Applied Nonparametric Statistical Methods*. 4th ed. Boca Raton, FL: Chapman & Hall/CRC.
- Sribney, W. M. 1995. `crc40`: Correcting for ties and zeros in sign and rank tests. *Stata Technical Bulletin* 26: 2–4. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 5–8. College Station, TX: Stata Press.
- Wilcoxon, F. 1945. Individual comparisons by ranking methods. *Biometrics* 1: 80–83.

## Also see

- [R] [signrank](#) — Equality tests on matched data
- [R] [ttest](#) — Mean-comparison tests

Syntax

Basic syntax

```
ratio [name:] varname [/] varname
```

Full syntax

```
ratio ([name:] varname [/] varname)
 [([name:] varname [/] varname) ...] [if] [in] [weight] [, options]
```

| options                               | description                                                                                                                        |
|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| Model                                 |                                                                                                                                    |
| <code>stdize(varname)</code>          | variable identifying strata for standardization                                                                                    |
| <code>stdweight(varname)</code>       | weight variable for standardization                                                                                                |
| <code>nostdrescale</code>             | do not rescale the standard weight variable                                                                                        |
| if/in/over                            |                                                                                                                                    |
| <code>over(varlist[, nolabel])</code> | group over subpopulations defined by <i>varlist</i> ; optionally, suppress group labels                                            |
| SE/Cluster                            |                                                                                                                                    |
| <code>vce(vctype)</code>              | <i>vcetype</i> may be <code>linearized</code> , <code>cluster clustvar</code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                             |                                                                                                                                    |
| <code>level(#)</code>                 | set confidence level; default is <code>level(95)</code>                                                                            |
| <code>noheader</code>                 | suppress table header                                                                                                              |
| <code>nolegend</code>                 | suppress table legend                                                                                                              |
| <sup>†</sup> <code>coeflegend</code>  | display coefficients' legend instead of coefficient table                                                                          |

<sup>†</sup>`coeflegend` does not appear in the dialog box.  
bootstrap, jackknife, mi estimate, rolling, statsby, and svy are allowed; see [U] 11.1.10 Prefix commands.  
`vce(bootstrap)` and `vce(jackknife)` are not allowed with the mi estimate prefix.  
Weights are not allowed with the bootstrap prefix.  
`vce()` and weights are not allowed with the svy prefix.  
`fweights`, `iweights`, and `pweights` are allowed; see [U] 11.1.6 weight.  
See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Ratios

## Description

`ratio` produces estimates of ratios, along with standard errors.

## Options

### Model

`stdize(varname)` specifies that the point estimates be adjusted by direct standardization across the strata identified by *varname*. This option requires the `stdweight()` option.

`stdweight(varname)` specifies the weight variable associated with the standard strata identified in the `stdize()` option. The standardization weights must be constant within the standard strata.

`nostdrescale` prevents the standardization weights from being rescaled within the `over()` groups. This option requires `stdize()` but is ignored if the `over()` option is not specified.

### if/in/over

`over(varlist [ , no label ])` specifies that estimates be computed for multiple subpopulations, which are identified by the different values of the variables in *varlist*.

When this option is supplied with one variable name, such as `over(varname)`, the value labels of *varname* are used to identify the subpopulations. If *varname* does not have labeled values (or there are unlabeled values), the values themselves are used, provided that they are nonnegative integers. Noninteger values, negative values, and labels that are not valid Stata names are substituted with a default identifier.

When `over()` is supplied with multiple variable names, each subpopulation is assigned a unique default identifier.

`no label` requests that value labels attached to the variables identifying the subpopulations be ignored.

### SE/Cluster

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

`vce(linearized)`, the default, uses the linearized or sandwich estimator of variance.

### Reporting

`level(#)`; see [R] [estimation options](#).

`noheader` prevents the table header from being displayed. This option implies `nolegend`.

`nolegend` prevents the table legend identifying the subpopulations from being displayed.

The following option is available with `ratio` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

### ► Example 1

Using the fuel data from [example 2](#) of [\[R\] ttest](#), we estimate the ratio of mileage for the cars without the fuel treatment (mpg1) to those with the fuel treatment (mpg2).

```
. use http://www.stata-press.com/data/r11/fuel
. ratio myratio: mpg1/mpg2
Ratio estimation Number of obs = 12
 myratio: mpg1/mpg2
```

|         | Linearized |           |                      |          |
|---------|------------|-----------|----------------------|----------|
|         | Ratio      | Std. Err. | [95% Conf. Interval] |          |
| myratio | .9230769   | .032493   | .8515603             | .9945936 |

Using these results, we can test to see if this ratio is significantly different from one.

```
. test _b[myratio] = 1
(1) myratio = 1
 F(1, 11) = 5.60
 Prob > F = 0.0373
```

We find that the ratio is different from one at the 5% significance level but not at the 1% significance level.



### ► Example 2

Using state-level census data, we want to test whether the marriage rate is equal to the death rate.

```
. use http://www.stata-press.com/data/r11/census2
(1980 Census data by state)
. ratio (deathrate: death/pop) (marrate: marriage/pop)
Ratio estimation Number of obs = 50
 deathrate: death/pop
 marrate: marriage/pop
```

|           | Linearized |           |                      |          |
|-----------|------------|-----------|----------------------|----------|
|           | Ratio      | Std. Err. | [95% Conf. Interval] |          |
| deathrate | .0087368   | .0002052  | .0083244             | .0091492 |
| marrate   | .0105577   | .0006184  | .009315              | .0118005 |

```
. test _b[deathrate] = _b[marrate]
(1) deathrate - marrate = 0
 F(1, 49) = 6.93
 Prob > F = 0.0113
```





## Saved results

ratio saves the following in `e()`:

### Scalars

|                          |                           |                      |                                          |
|--------------------------|---------------------------|----------------------|------------------------------------------|
| <code>e(N)</code>        | number of observations    | <code>e(k_eq)</code> | number of equations in <code>e(b)</code> |
| <code>e(N_over)</code>   | number of subpopulations  | <code>e(df_r)</code> | sample degrees of freedom                |
| <code>e(N_stdize)</code> | number of standard strata | <code>e(rank)</code> | rank of <code>e(V)</code>                |
| <code>e(N_clust)</code>  | number of clusters        |                      |                                          |

### Macros

|                           |                                              |                               |                                                |
|---------------------------|----------------------------------------------|-------------------------------|------------------------------------------------|
| <code>e(cmd)</code>       | ratio                                        | <code>e(over)</code>          | <i>varlist</i> from <code>over()</code>        |
| <code>e(cmdline)</code>   | command as typed                             | <code>e(over_labels)</code>   | labels from <code>over()</code> variables      |
| <code>e(varlist)</code>   | <i>varlist</i>                               | <code>e(over_namelist)</code> | names from <code>e(over_labels)</code>         |
| <code>e(stdize)</code>    | <i>varname</i> from <code>stdize()</code>    | <code>e(namelist)</code>      | ratio identifiers                              |
| <code>e(stdweight)</code> | <i>varname</i> from <code>stdweight()</code> | <code>e(vce)</code>           | <i>vcetype</i> specified in <code>vce()</code> |
| <code>e(wtype)</code>     | weight type                                  | <code>e(vcetype)</code>       | title used to label Std. Err.                  |
| <code>e(wexp)</code>      | weight expression                            | <code>e(properties)</code>    | <b>b</b> <i>V</i>                              |
| <code>e(title)</code>     | title in estimation output                   | <code>e(estat_cmd)</code>     | program used to implement <code>estat</code>   |
| <code>e(cluster)</code>   | name of cluster variable                     | <code>e(marginsnotok)</code>  | predictions disallowed by <code>margins</code> |

### Matrices

|                           |                                                              |
|---------------------------|--------------------------------------------------------------|
| <code>e(b)</code>         | vector of mean estimates                                     |
| <code>e(V)</code>         | (co)variance estimates                                       |
| <code>e(_N)</code>        | vector of numbers of nonmissing observations                 |
| <code>e(_N_stdsum)</code> | number of nonmissing observations within the standard strata |
| <code>e(_p_stdize)</code> | standardizing proportions                                    |
| <code>e(error)</code>     | error code corresponding to <code>e(b)</code>                |

### Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

ratio is implemented as an ado-file.

Methods and formulas are presented under the following headings:

*The ratio estimator*  
*Survey data*  
*The survey ratio estimator*  
*The standardized ratio estimator*  
*The poststratified ratio estimator*  
*The standardized poststratified ratio estimator*  
*Subpopulation estimation*

## The ratio estimator

Let  $R = Y/X$  be the ratio to be estimated, where  $Y$  and  $X$  are totals; see [R] **total**. The estimate for  $R$  is  $\hat{R} = \hat{Y}/\hat{X}$  (the ratio of the sample totals). From the delta method (i.e., a first-order Taylor expansion), the approximate variance of the sampling distribution of the linearized  $\hat{R}$  is

$$V(\hat{R}) \approx \frac{1}{X^2} \left\{ V(\hat{Y}) - 2R\text{Cov}(\hat{Y}, \hat{X}) + R^2 V(\hat{X}) \right\}$$

Direct substitution of  $\hat{X}$ ,  $\hat{R}$ , and the estimated variances and covariance of  $\hat{X}$  and  $\hat{Y}$  leads to the following variance estimator:

$$\hat{V}(\hat{R}) = \frac{1}{\hat{X}^2} \left\{ \hat{V}(\hat{Y}) - 2\hat{R}\widehat{\text{Cov}}(\hat{Y}, \hat{X}) + \hat{R}^2\hat{V}(\hat{X}) \right\} \quad (1)$$

## Survey data

See [\[SVY\] variance estimation](#), [\[SVY\] direct standardization](#), and [\[SVY\] poststratification](#) for discussions that provide background information for the following formulas.

## The survey ratio estimator

Let  $Y_j$  and  $X_j$  be survey items for the  $j$ th individual in the population, where  $j = 1, \dots, M$  and  $M$  is the size of the population. The associated population ratio for the items of interest is  $R = Y/X$  where

$$Y = \sum_{j=1}^M Y_j \quad \text{and} \quad X = \sum_{j=1}^M X_j$$

Let  $y_j$  and  $x_j$  be the corresponding survey items for the  $j$ th sampled individual from the population, where  $j = 1, \dots, m$  and  $m$  is the number of observations in the sample.

The estimator  $\hat{R}$  for the population ratio  $R$  is  $\hat{R} = \hat{Y}/\hat{X}$ , where

$$\hat{Y} = \sum_{j=1}^m w_j y_j \quad \text{and} \quad \hat{X} = \sum_{j=1}^m w_j x_j$$

and  $w_j$  is a sampling weight. The score variable for the ratio estimator is

$$z_j(\hat{R}) = \frac{y_j - \hat{R}x_j}{\hat{X}} = \frac{\hat{X}y_j - \hat{Y}x_j}{\hat{X}^2}$$

## The standardized ratio estimator

Let  $D_g$  denote the set of sampled observations that belong to the  $g$ th standard stratum and define  $I_{D_g}(j)$  to indicate if the  $j$ th observation is a member of the  $g$ th standard stratum; where  $g = 1, \dots, L_D$  and  $L_D$  is the number of standard strata. Also, let  $\pi_g$  denote the fraction of the population that belongs to the  $g$ th standard stratum, thus  $\pi_1 + \dots + \pi_{L_D} = 1$ . Note that  $\pi_g$  is derived from the `stdweight()` option.

The estimator for the standardized ratio is

$$\hat{R}^D = \sum_{g=1}^{L_D} \pi_g \frac{\hat{Y}_g}{\hat{X}_g}$$

where

$$\hat{Y}_g = \sum_{j=1}^m I_{D_g}(j) w_j y_j$$

and  $\hat{X}_g$  is similarly defined. The score variable for the standardized ratio is

$$z_j(\hat{R}^D) = \sum_{g=1}^{L_D} \pi_g I_{D_g}(j) \frac{\hat{X}_g y_j - \hat{Y}_g x_j}{\hat{X}_g^2}$$

## The poststratified ratio estimator

Let  $P_k$  denote the set of sampled observations that belong to poststratum  $k$ , and define  $I_{P_k}(j)$  to indicate if the  $j$ th observation is a member of poststratum  $k$ , where  $k = 1, \dots, L_P$  and  $L_P$  is the number of poststrata. Also, let  $M_k$  denote the population size for poststratum  $k$ .  $P_k$  and  $M_k$  are identified by specifying the `poststrata()` and `postweight()` options on `svyset`.

The estimator for the poststratified ratio is

$$\hat{R}^P = \frac{\hat{Y}^P}{\hat{X}^P}$$

where

$$\hat{Y}^P = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \hat{Y}_k = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \sum_{j=1}^m I_{P_k}(j) w_j y_j$$

and  $\hat{X}^P$  is similarly defined. The score variable for the poststratified ratio is

$$z_j(\hat{R}^P) = \frac{z_j(\hat{Y}^P) - \hat{R}^P z_j(\hat{X}^P)}{\hat{X}^P} = \frac{\hat{X}^P z_j(\hat{Y}^P) - \hat{Y}^P z_j(\hat{X}^P)}{(\hat{X}^P)^2}$$

where

$$z_j(\hat{Y}^P) = \sum_{k=1}^{L_P} I_{P_k}(j) \frac{M_k}{\widehat{M}_k} \left( y_j - \frac{\hat{Y}_k}{\widehat{M}_k} \right)$$

and  $z_j(\hat{X}^P)$  is similarly defined.

## The standardized poststratified ratio estimator

The estimator for the standardized poststratified ratio is

$$\hat{R}^{DP} = \sum_{g=1}^{L_D} \pi_g \frac{\hat{Y}_g^P}{\hat{X}_g^P}$$

where

$$\hat{Y}_g^P = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \hat{Y}_{g,k} = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \sum_{j=1}^m I_{D_g}(j) I_{P_k}(j) w_j y_j$$

and  $\hat{X}_g^P$  is similarly defined. The score variable for the standardized poststratified ratio is

$$z_j(\hat{R}^{DP}) = \sum_{g=1}^{L_D} \pi_g \frac{\hat{X}_g^P z_j(\hat{Y}_g^P) - \hat{Y}_g^P z_j(\hat{X}_g^P)}{(\hat{X}_g^P)^2}$$

where

$$z_j(\hat{Y}_g^P) = \sum_{k=1}^{L_P} I_{P_k}(j) \frac{M_k}{\widehat{M}_k} \left\{ I_{D_g}(j) y_j - \frac{\hat{Y}_{g,k}}{\widehat{M}_k} \right\}$$

and  $z_j(\hat{X}_g^P)$  is similarly defined.

## Subpopulation estimation

Let  $S$  denote the set of sampled observations that belong to the subpopulation of interest, and define  $I_S(j)$  to indicate if the  $j$ th observation falls within the subpopulation.

The estimator for the subpopulation ratio is  $\hat{R}^S = \hat{Y}^S / \hat{X}^S$ , where

$$\hat{Y}^S = \sum_{j=1}^m I_S(j) w_j y_j \quad \text{and} \quad \hat{X}^S = \sum_{j=1}^m I_S(j) w_j x_j$$

Its score variable is

$$z_j(\hat{R}^S) = I_S(j) \frac{y_j - \hat{R}^S x_j}{\hat{X}^S} = I_S(j) \frac{\hat{X}^S y_j - \hat{Y}^S x_j}{(\hat{X}^S)^2}$$

The estimator for the standardized subpopulation ratio is

$$\hat{R}^{DS} = \sum_{g=1}^{L_D} \pi_g \frac{\hat{Y}_g^S}{\hat{X}_g^S}$$

where

$$\hat{Y}_g^S = \sum_{j=1}^m I_{D_g}(j) I_S(j) w_j y_j$$

and  $\hat{X}_g^S$  is similarly defined. Its score variable is

$$z_j(\hat{R}^{DS}) = \sum_{g=1}^{L_D} \pi_g I_{D_g}(j) I_S(j) \frac{\hat{X}_g^S y_j - \hat{Y}_g^S x_j}{(\hat{X}_g^S)^2}$$

The estimator for the poststratified subpopulation ratio is

$$\hat{R}^{PS} = \frac{\hat{Y}^{PS}}{\hat{X}^{PS}}$$

where

$$\hat{Y}^{PS} = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M_k}} \hat{Y}_k^S = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M_k}} \sum_{j=1}^m I_{P_k}(j) I_S(j) w_j y_j$$

and  $\hat{X}^{PS}$  is similarly defined. Its score variable is

$$z_j(\hat{R}^{PS}) = \frac{\hat{X}^{PS} z_j(\hat{Y}^{PS}) - \hat{Y}^{PS} z_j(\hat{X}^{PS})}{(\hat{X}^{PS})^2}$$

where

$$z_j(\hat{Y}^{PS}) = \sum_{k=1}^{L_P} I_{P_k}(j) \frac{M_k}{\widehat{M_k}} \left\{ I_S(j) y_j - \frac{\hat{Y}_k^S}{\widehat{M_k}} \right\}$$

and  $z_j(\hat{X}^{PS})$  is similarly defined.

The estimator for the standardized poststratified subpopulation ratio is

$$\hat{R}^{DPS} = \sum_{g=1}^{L_D} \pi_g \frac{\hat{Y}_g^{PS}}{\hat{X}_g^{PS}}$$

where

$$\hat{Y}_g^{PS} = \sum_{k=1}^{L_p} \frac{M_k}{\widehat{M_k}} \hat{Y}_{g,k}^S = \sum_{k=1}^{L_p} \frac{M_k}{\widehat{M_k}} \sum_{j=1}^m I_{D_g}(j) I_{P_k}(j) I_S(j) w_j y_j$$

and  $\hat{X}_g^{PS}$  is similarly defined. Its score variable is

$$z_j(\hat{R}^{DPS}) = \sum_{g=1}^{L_D} \pi_g \frac{\hat{X}_g^{PS} z_j(\hat{Y}_g^{PS}) - \hat{Y}_g^{PS} z_j(\hat{X}_g^{PS})}{(\hat{X}_g^{PS})^2}$$

where

$$z_j(\hat{Y}_g^{PS}) = \sum_{k=1}^{L_P} I_{P_k}(j) \frac{M_k}{\widehat{M_k}} \left\{ I_{D_g}(j) I_S(j) y_j - \frac{\hat{Y}_{g,k}^S}{\widehat{M_k}} \right\}$$

and  $z_j(\hat{X}_g^{PS})$  is similarly defined.

## References

- Cochran, W. G. 1977. *Sampling Techniques*. 3rd ed. New York: Wiley.
- Stuart, A., and J. K. Ord. 1994. *Kendall's Advanced Theory of Statistics: Distribution Theory*, Vol I. 6th ed. London: Arnold.

## Also see

[R] **ratio postestimation** — Postestimation tools for ratio

[R] **mean** — Estimate means

[R] **proportion** — Estimate proportions

[R] **total** — Estimate totals

[SVY] **direct standardization** — Direct standardization of means, proportions, and ratios

[SVY] **poststratification** — Poststratification for survey data

[SVY] **subpopulation estimation** — Subpopulation estimation for survey data

[SVY] **svy estimation** — Estimation commands for survey data

[SVY] **variance estimation** — Variance estimation for survey data

[U] **20 Estimation and postestimation commands**

# Title

ratio postestimation — Postestimation tools for ratio

## Description

The following postestimation commands are available for `ratio`:

| command                  | description                                                                                         |
|--------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>       | VCE                                                                                                 |
| <code>estat (svy)</code> | postestimation statistics for survey data                                                           |
| <code>estimates</code>   | cataloging estimation results                                                                       |
| <code>lincom</code>      | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>nlcom</code>       | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>test</code>        | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>      | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

## Remarks

For examples of the use of `test` after `ratio`, see [\[R\] ratio](#).

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

- [\[R\] ratio](#) — Estimate ratios
- [\[SVY\] svy postestimation](#) — Postestimation tools for svy
- [\[U\] 20 Estimation and postestimation commands](#)

Syntax

Basic syntax

```
reg3 (depvar1 varlist1) (depvar2 varlist2) ... (depvarN varlistN) [if] [in] [weight]
```

Full syntax

```
reg3 ([eqname1:]depvar1a [depvar1b ...=]varlist1 [, noconstant])
 ([eqname2:]depvar2a [depvar2b ...=]varlist2 [, noconstant])
 ...
 ([eqnameN:]depvarNa [depvarNb ...=]varlistN [, noconstant])
 [if] [in] [weight] [, options]
```

| <i>options</i>                            | description                                                                                     |
|-------------------------------------------|-------------------------------------------------------------------------------------------------|
| Model                                     |                                                                                                 |
| <u>ireg3</u>                              | iterate until estimates converge                                                                |
| <u>constraints</u> ( <i>constraints</i> ) | apply specified linear constraints                                                              |
| Model 2                                   |                                                                                                 |
| <u>exog</u> ( <i>varlist</i> )            | exogenous variables not specified in system equations                                           |
| <u>endog</u> ( <i>varlist</i> )           | additional right-hand-side endogenous variables                                                 |
| <u>inst</u> ( <i>varlist</i> )            | full list of exogenous variables                                                                |
| <u>allexog</u>                            | all right-hand-side variables are exogenous                                                     |
| <u>noconstant</u>                         | suppress constant from instrument list                                                          |
| Est. method                               |                                                                                                 |
| <u>3sls</u>                               | three-stage least squares; the default                                                          |
| <u>2sls</u>                               | two-stage least squares                                                                         |
| <u>ols</u>                                | ordinary least squares (OLS)                                                                    |
| <u>sure</u>                               | seemingly unrelated regression (SURE)                                                           |
| <u>mvreg</u>                              | <u>sure</u> with OLS degrees-of-freedom adjustment                                              |
| <u>corr</u> ( <i>correlation</i> )        | <u>unstructured</u> or <u>independent</u> correlation structure; default is <u>unstructured</u> |
| df adj.                                   |                                                                                                 |
| <u>small</u>                              | report small-sample statistics                                                                  |
| <u>dfk</u>                                | use small-sample adjustment                                                                     |
| <u>dfk2</u>                               | use alternate adjustment                                                                        |



## Reporting

|                              |                                                                           |
|------------------------------|---------------------------------------------------------------------------|
| <code>level(#)</code>        | set confidence level; default is <code>level(95)</code>                   |
| <code>first</code>           | report first-stage regression                                             |
| <code>nocnsreport</code>     | do not display constraints                                                |
| <code>display_options</code> | control spacing and display of omitted variables and base and empty cells |

## Optimization

|                                   |                                                           |
|-----------------------------------|-----------------------------------------------------------|
| <code>optimization_options</code> | control the optimization process; seldom used             |
| <code>† noheader</code>           | suppress display of header                                |
| <code>† notable</code>            | suppress display of coefficient table                     |
| <code>† nofooter</code>           | suppress display of footer                                |
| <code>† coeflegend</code>         | display coefficients' legend instead of coefficient table |

† `noheader`, `notable`, `nofooter`, and `coeflegend` do not appear in the dialog box.

`varlist1, ..., varlistN` and the `exog()` and the `inst()` `varlist` may contain factor variables; see

[U] 11.4.3 **Factor variables**. You must have the same levels of factor variables in all equations that have factor variables.

`depvar` and `varlist` may contain time-series operators; see [U] 11.4.4 **Time-series varlists**.

`bootstrap`, `by`, `jackknife`, `rolling`, and `statsby` are allowed; see [U] 11.1.10 **Prefix commands**.

Weights are not allowed with the `bootstrap` prefix.

`aweight`s are not allowed with the `jackknife` prefix.

`aweight`s and `fweight`s are allowed; see [U] 11.1.6 **weight**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Explicit equation naming (`eqname:`) cannot be combined with multiple dependent variables in an equation specification.

## Menu

Statistics > Endogenous covariates > Three-stage least squares

## Description

`reg3` estimates a system of structural equations, where some equations contain endogenous variables among the explanatory variables. Estimation is via three-stage least squares (3SLS); see Zellner and Theil (1962). Typically, the endogenous explanatory variables are dependent variables from other equations in the system. `reg3` supports iterated GLS estimation and linear constraints.

`reg3` can also estimate systems of equations by seemingly unrelated regression estimation (SURE), multivariate regression (MVREG), and equation-by-equation ordinary least squares (OLS) or two-stage least squares (2SLS).

## Nomenclature

Under 3SLS or 2SLS estimation, a *structural equation* is defined as one of the equations specified in the system. A *dependent variable* will have its usual interpretation as the left-hand-side variable in an equation with an associated disturbance term. All dependent variables are explicitly taken to be *endogenous* to the system and are treated as correlated with the disturbances in the system's equations. Unless specified in an `endog()` option, all other variables in the system are treated as *exogenous* to the system and uncorrelated with the disturbances. The exogenous variables are taken to be *instruments* for the endogenous variables.

## Options

### Model

**ireg3** causes **reg3** to iterate over the estimated disturbance covariance matrix and parameter estimates until the parameter estimates converge. Although the iteration is usually successful, there is no guarantee that it will converge to a stable point. Under SURE, this iteration converges to the maximum likelihood estimates.

**constraints**(*constraints*); see [\[R\] estimation options](#).

### Model 2

**exog**(*varlist*) specifies additional exogenous variables that are included in none of the system equations. This can occur when the system contains identities that are not estimated. If implicitly exogenous variables from the equations are listed here, **reg3** will just ignore the additional information. Specified variables will be added to the exogenous variables in the system and used in the first stage as instruments for the endogenous variables. By specifying dependent variables from the structural equations, you can use **exog**() to override their endogeneity.

**endog**(*varlist*) identifies variables in the system that are not dependent variables but are endogenous to the system. These variables must appear in the variable list of at least one equation in the system. Again the need for this identification often occurs when the system contains identities. For example, a variable that is the sum of an exogenous variable and a dependent variable may appear as an explanatory variable in some equations.

**inst**(*varlist*) specifies a full list of all exogenous variables and may not be used with the **endog**() or **exog**() options. It must contain a full list of variables to be used as instruments for the endogenous regressors. Like **exog**(), the list may contain variables not specified in the system of equations. This option can be used to achieve the same results as the **endog**() and **exog**() options, and the choice is a matter of convenience. Any variable not specified in the *varlist* of the **inst**() option is assumed to be endogenous to the system. As with **exog**(), including the dependent variables from the structural equations will override their endogeneity.

**allexog** indicates that all right-hand-side variables are to be treated as exogenous—even if they appear as the dependent variable of another equation in the system. This option can be used to enforce a SURE or MVREG estimation even when some dependent variables appear as regressors.

**noconstant**; see [\[R\] estimation options](#).

### Est. method

**3sls** specifies the full 3SLS estimation of the system and is the default for **reg3**.

**2sls** causes **reg3** to perform equation-by-equation 2SLS on the full system of equations. This option implies **dfk**, **small**, and **corr(independent)**.

Cross-equation testing should not be performed after estimation with this option. With **2sls**, no covariance is estimated between the parameters of the equations. For cross-equation testing, use **3sls**.

**ols** causes **reg3** to perform equation-by-equation OLS on the system—even if dependent variables appear as regressors or the regressors differ for each equation; see [\[R\] mvreg](#). **ols** implies **allexog**, **dfk**, **small**, and **corr(independent)**; **nodfk** and **nosmall** may be specified to override **dfk** and **small**.

The covariance of the coefficients between equations is not estimated under this option, and cross-equation tests should not be performed after estimation with **ols**. For cross-equation testing, use **sure** or **3sls** (the default).

`sure` causes `reg3` to perform a SURE of the system—even if dependent variables from some equations appear as regressors in other equations; see [R] [sureg](#). `sure` is a synonym for `allexog`.

`mvreg` is identical to `sure`, except that the disturbance covariance matrix is estimated with an OLS degrees-of-freedom adjustment—the `dfk` option. If the regressors are identical for all equations, the parameter point estimates will be the standard MVREG results. If any of the regressors differ, the point estimates are those for SURE with an OLS degrees-of-freedom adjustment in computing the covariance matrix. `nodfk` and `nosmall` may be specified to override `dfk` and `small`.

`corr`(*correlation*) specifies the assumed form of the correlation structure of the equation disturbances and is rarely requested explicitly. For the family of models fit by `reg3`, the only two allowable correlation structures are `unstructured` and `independent`. The default is `unstructured`.

This option is used almost exclusively to estimate a system of equations by 2SLS or to perform OLS regression with `reg3` on multiple equations. In these cases, the correlation is set to `independent`, forcing `reg3` to treat the covariance matrix of equation disturbances as diagonal in estimating model parameters. Thus a set of two-stage coefficient estimates can be obtained if the system contains endogenous right-hand-side variables, or OLS regression can be imposed, even if the regressors differ across equations. Without imposing independent disturbances, `reg3` would estimate the former by 3SLS and the latter by SURE.

Any tests performed after estimation with the `independent` option will treat coefficients in different equations as having no covariance; cross-equation tests should not be used after specifying `corr(independent)`.

#### df adj.

`small` specifies that small-sample statistics be computed. It shifts the test statistics from  $\chi^2$  and  $z$  statistics to  $F$  statistics and  $t$  statistics. This option is intended primarily to support MVREG. Although the standard errors from each equation are computed using the degrees of freedom for the equation, the degrees of freedom for the  $t$  statistics are all taken to be those for the first equation. This approach poses no problem under MVREG because the regressors are the same across equations.

`dfk` specifies the use of an alternative divisor in computing the covariance matrix for the equation residuals. As an asymptotically justified estimator, `reg3` by default uses the number of sample observations  $n$  as a divisor. When the `dfk` option is set, a small-sample adjustment is made, and the divisor is taken to be  $\sqrt{(n - k_i)(n - k_j)}$ , where  $k_i$  and  $k_j$  are the numbers of parameters in equations  $i$  and  $j$ , respectively.

`dfk2` specifies the use of an alternative divisor in computing the covariance matrix for the equation errors. When the `dfk2` option is set, the divisor is taken to be the mean of the residual degrees of freedom from the individual equations.

#### Reporting

`level(#)`; see [R] [estimation options](#).

`first` requests that the first-stage regression results be displayed during estimation.

`nocnsreport`; see [R] [estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

## Optimization

`optimization_options` control the iterative process that minimizes the sum of squared errors when `ireg3` is specified. These options are seldom used.

`iterate(#)` specifies the maximum number of iterations. When the number of iterations equals `#`, the optimizer stops and presents the current results, even if the convergence tolerance has not been reached. The default value of `iterate()` is the current value of `set maxiter`, which is `iterate(16000)` if `maxiter` has not been changed.

`trace` adds to the iteration log a display of the current parameter vector

`nolog` suppresses the display of the iteration log.

`tolerance(#)` specifies the tolerance for the coefficient vector. When the relative change in the coefficient vector from one iteration to the next is less than or equal to `#`, the optimization process is stopped. `tolerance(1e-6)` is the default.

The following options are available with `reg3` but are not shown in the dialog box:

`noheader` suppresses display of the header reporting the estimation method and the table of equation summary statistics.

`notable` suppresses display of the coefficient table.

`nofooter` suppresses display of the footer reporting the list of endogenous variables in the model.

`coeflegend`; see [R] [estimation options](#).

## Remarks

`reg3` estimates systems of structural equations where some equations contain endogenous variables among the explanatory variables. Generally, these endogenous variables are the dependent variables of other equations in the system, though not always. The disturbance is correlated with the endogenous variables—violating the assumptions of OLS. Further, because some of the explanatory variables are the dependent variables of other equations in the system, the error terms among the equations are expected to be correlated. `reg3` uses an instrumental-variables approach to produce consistent estimates and generalized least squares (GLS) to account for the correlation structure in the disturbances across the equations. Good general references on three-stage estimation include [Davidson and MacKinnon \(1993, 651–661\)](#) and [Greene \(2008, 381–383\)](#).

Three-stage least squares can be thought of as producing estimates from a three-step process.

**Step 1.** Develop instrumented values for all endogenous variables. These instrumented values can simply be considered as the predicted values resulting from a regression of each endogenous variable on all exogenous variables in the system. This stage is identical to the first step in 2SLS and is critical for the consistency of the parameter estimates.

**Step 2.** Obtain a consistent estimate for the covariance matrix of the equation disturbances. These estimates are based on the residuals from a 2SLS estimation of each structural equation.

**Step 3.** Perform a GLS-type estimation using the covariance matrix estimated in the second stage and with the instrumented values in place of the right-hand-side endogenous variables.

### Technical note

The estimation and use of the covariance matrix of disturbances in three-stage estimation is almost identical to the SURE method—`sureg`. As with SURE, using this covariance matrix improves the efficiency of the three-stage estimator. Even without the covariance matrix, the estimates would be

consistent. (They would be 2SLS estimates.) This improvement in efficiency comes with a caveat. All the parameter estimates now depend on the consistency of the covariance matrix estimates. If one equation in the system is misspecified, the disturbance covariance estimates will be inconsistent, and the resulting coefficients will be biased and inconsistent. Alternatively, if each equation is estimated separately by 2SLS ([R] [regress](#)), only the coefficients in the misspecified equation are affected.

□

## □ Technical note

If an equation is just identified, the 3SLS point estimates for that equation are identical to the 2SLS estimates. However, as with `sureg`, even if all equations are just identified, fitting the model via `reg3` has at least one advantage over fitting each equation separately via `ivregress`; by using `reg3`, tests involving coefficients in different equations can be performed easily using `test` or `testnl`.

□

## ▷ Example 1

A simple macroeconomic model relates consumption (`consump`) to private and government wages paid (`wagepriv` and `wagegovt`). Simultaneously, private wages depend on consumption, total government expenditures (`govt`), and the lagged stock of capital in the economy (`capital1`). Although this is not a plausible model, it does meet the criterion of being simple. This model could be written as

$$\begin{aligned}\text{consump} &= \beta_0 + \beta_1 \text{wagepriv} + \beta_2 \text{wagegovt} + \epsilon_1 \\ \text{wagepriv} &= \beta_3 + \beta_4 \text{consump} + \beta_5 \text{govt} + \beta_6 \text{capital1} + \epsilon_2\end{aligned}$$

If we assume that this is the full system, `consump` and `wagepriv` will be endogenous variables, with `wagegovt`, `govt`, and `capital1` exogenous. Data for the U.S. economy on these variables are taken from [Klein \(1950\)](#). This model can be fit with `reg3` by typing

```
. use http://www.stata-press.com/data/r11/klein
. reg3 (consump wagepriv wagegovt) (wagepriv consump govt capital1)
```

Three-stage least-squares regression

| Equation | Obs | Parms | RMSE     | "R-sq" | chi2   | P      |
|----------|-----|-------|----------|--------|--------|--------|
| consump  | 22  | 2     | 1.776297 | 0.9388 | 208.02 | 0.0000 |
| wagepriv | 22  | 3     | 2.372443 | 0.8542 | 80.04  | 0.0000 |

|          | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|----------|-----------|-----------|-------|-------|----------------------|----------|
| consump  |           |           |       |       |                      |          |
| wagepriv | .8012754  | .1279329  | 6.26  | 0.000 | .5505314             | 1.052019 |
| wagegovt | 1.029531  | .3048424  | 3.38  | 0.001 | .432051              | 1.627011 |
| _cons    | 19.3559   | 3.583772  | 5.40  | 0.000 | 12.33184             | 26.37996 |
| wagepriv |           |           |       |       |                      |          |
| consump  | .4026076  | .2567312  | 1.57  | 0.117 | -.1005764            | .9057916 |
| govt     | 1.177792  | .5421253  | 2.17  | 0.030 | .1152461             | 2.240338 |
| capital1 | -.0281145 | .0572111  | -0.49 | 0.623 | -.1402462            | .0840173 |
| _cons    | 14.63026  | 10.26693  | 1.42  | 0.154 | -5.492552            | 34.75306 |

Endogenous variables: consump wagepriv

Exogenous variables: wagegovt govt capital1

Without showing the 2SLS results, we note that the consumption function in this system falls under the conditions noted earlier. That is, the 2SLS and 3SLS coefficients for the equation are identical.



## ► Example 2

Some of the most common simultaneous systems encountered are supply-and-demand models. A simple system could be specified as

$$q_{\text{Demand}} = \beta_0 + \beta_1 \text{price} + \beta_2 p_{\text{compete}} + \beta_3 \text{income} + \epsilon_1$$

$$q_{\text{Supply}} = \beta_4 + \beta_5 \text{price} + \beta_6 p_{\text{raw}} + \epsilon_2$$

$$\text{Equilibrium condition: } \text{quantity} = q_{\text{Demand}} = q_{\text{Supply}}$$

where

`quantity` is the quantity of a product produced and sold,  
`price` is the price of the product,  
`pcompete` is the price of a competing product,  
`income` is the average income level of consumers, and  
`praw` is the price of raw materials used to produce the product.

In this system, price is assumed to be determined simultaneously with demand. The important statistical implications are that price is not a predetermined variable and that it is correlated with the disturbances of both equations. The system is somewhat unusual: quantity is associated with two disturbances. This fact really poses no problem because the disturbances are specified on the behavioral demand and supply equations—two separate entities. Often one of the two equations is rewritten to place price on the left-hand side, making this endogeneity explicit in the specification.

To provide a concrete illustration of the effects of simultaneous equations, we can simulate data for the above system by using known coefficients and disturbance properties. Specifically, we will simulate the data as

$$q_{\text{Demand}} = 40 - 1.0 \text{price} + 0.25 p_{\text{compete}} + 0.5 \text{income} + \epsilon_1$$

$$q_{\text{Supply}} = 0.5 \text{price} - 0.75 p_{\text{raw}} + \epsilon_2$$

where

$$\epsilon_1 \sim N(0, 2.4)$$

$$\epsilon_2 \sim N(0, 3.8)$$

For comparison, we can estimate the supply and demand equations separately by OLS. The estimates for the demand equation are

```
. use http://www.stata-press.com/data/r11/supDem
```

```
. regress quantity price pcompete income
```

| Source   | SS         | df | MS         | Number of obs = 49     |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | 23.1579302 | 3  | 7.71931008 | F( 3, 45) = 1.00       |  |  |
| Residual | 346.459313 | 45 | 7.69909584 | Prob > F = 0.4004      |  |  |
|          |            |    |            | R-squared = 0.0627     |  |  |
|          |            |    |            | Adj R-squared = 0.0002 |  |  |
| Total    | 369.617243 | 48 | 7.70035923 | Root MSE = 2.7747      |  |  |

| quantity | Coef.    | Std. Err. | t    | P> t  | [95% Conf. Interval] |          |
|----------|----------|-----------|------|-------|----------------------|----------|
| price    | .1186265 | .1716014  | 0.69 | 0.493 | -.2269965            | .4642496 |
| pcompete | .0946416 | .1200815  | 0.79 | 0.435 | -.1472149            | .3364981 |
| income   | .0785339 | .1159867  | 0.68 | 0.502 | -.1550754            | .3121432 |
| _cons    | 7.563261 | 5.019479  | 1.51 | 0.139 | -2.54649             | 17.67301 |

The OLS estimates for the supply equation are

```
. regress quantity price praw
```

| Source   | SS         | df | MS         | Number of obs = 49     |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | 224.819549 | 2  | 112.409774 | F( 2, 46) = 35.71      |  |  |
| Residual | 144.797694 | 46 | 3.14777596 | Prob > F = 0.0000      |  |  |
|          |            |    |            | R-squared = 0.6082     |  |  |
|          |            |    |            | Adj R-squared = 0.5912 |  |  |
| Total    | 369.617243 | 48 | 7.70035923 | Root MSE = 1.7742      |  |  |

| quantity | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|----------|-----------|-----------|-------|-------|----------------------|----------|
| price    | .724675   | .1095657  | 6.61  | 0.000 | .5041307             | .9452192 |
| praw     | -.8674796 | .1066114  | -8.14 | 0.000 | -1.082077            | -.652882 |
| _cons    | -6.97291  | 3.323105  | -2.10 | 0.041 | -13.66197            | -.283847 |

Examining the coefficients from these regressions, we note that they are not close to the known parameters used to generate the simulated data. In particular, the positive coefficient on *price* in the demand equation stands out. We constructed our simulated data to be consistent with economic theory—people demand less of a product if its price rises and more if their personal income rises. Although the *price* coefficient is statistically insignificant, the positive value contrasts starkly with what is predicted from economic price theory and the  $-1.0$  value that we used in the simulation. Likewise, we are disappointed with the insignificance and level of the coefficient on average *income*. The supply equation has correct signs on the two main parameters, but their levels are different from the known values. In fact, the coefficient on *price* (0.724675) is different from the simulated parameter (0.5) at the 5% level of significance.

All these problems are to be expected. We explicitly constructed a simultaneous system of equations that violated one of the assumptions of least squares. Specifically, the disturbances were correlated with one of the regressors—*price*.

Two-stage least squares can be used to address the correlation between regressors and disturbances. Using instruments for the endogenous variable, *price*, 2SLS will produce consistent estimates of the parameters in the system. Let's use *ivregress* to see how our simulated system behaves when fit using 2SLS.

```
. ivregress 2sls quantity (price = praw) pcompete income
Instrumental variables (2SLS) regression Number of obs = 49
 Wald chi2(3) = 8.77
 Prob > chi2 = 0.0326
 R-squared = .
 Root MSE = 3.7333
```

| quantity | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|----------|-----------|-----------|-------|-------|----------------------|----------|
| price    | -1.015817 | .374209   | -2.71 | 0.007 | -1.749253            | -.282381 |
| pcompete | .3319504  | .172912   | 1.92  | 0.055 | -.0069508            | .6708517 |
| income   | .5090607  | .1919482  | 2.65  | 0.008 | .1328491             | .8852723 |
| _cons    | 39.89988  | 10.77378  | 3.70  | 0.000 | 18.78366             | 61.01611 |

```
Instrumented: price
Instruments: pcompete income praw
. ivregress 2sls quantity (price = pcompete income) praw
Instrumental variables (2SLS) regression Number of obs = 49
 Wald chi2(2) = 39.25
 Prob > chi2 = 0.0000
 R-squared = 0.5928
 Root MSE = 1.7525
```

| quantity | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| price    | .5773133  | .1749974  | 3.30  | 0.001 | .2343247             | .9203019  |
| praw     | -.7835496 | .1312414  | -5.97 | 0.000 | -1.040778            | -.5263213 |
| _cons    | -2.550694 | 5.273067  | -0.48 | 0.629 | -12.88571            | 7.784327  |

```
Instrumented: price
Instruments: praw pcompete income
```

We are now much happier with the estimation results. All the coefficients from both equations are close to the true parameter values for the system. In particular, the coefficients are all well within 95% confidence intervals for the parameters. The missing *R*-squared in the demand equation seems unusual; we will discuss that more later.

Finally, this system could be estimated using 3SLS. To demonstrate how large systems might be handled and to avoid multiline commands, we will use global macros (see [\[P\] macro](#)) to hold the specifications for our equations.

```
. global demand "(qDemand: quantity price pcompete income)"
. global supply "(qSupply: quantity price praw)"
. reg3 $demand $supply, endog(price)
```

We must specify `price` as endogenous because it does not appear as a dependent variable in either equation. Without this option, `reg3` would assume that there are no endogenous variables in the system and produce seemingly unrelated regression (`sureg`) estimates. The `reg3` output from our series of commands is



## Three-stage least-squares regression

| Equation | Obs | Parms | RMSE     | "R-sq"  | chi2  | P      |
|----------|-----|-------|----------|---------|-------|--------|
| qDemand  | 49  | 3     | 3.739686 | -0.8540 | 8.68  | 0.0338 |
| qSupply  | 49  | 2     | 1.752501 | 0.5928  | 39.25 | 0.0000 |

|          | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| qDemand  |           |           |       |       |                      |           |
| price    | -1.014345 | .3742036  | -2.71 | 0.007 | -1.74777             | -.2809194 |
| pcompete | .2647206  | .1464194  | 1.81  | 0.071 | -.0222561            | .5516973  |
| income   | .5299146  | .1898161  | 2.79  | 0.005 | .1578819             | .9019472  |
| _cons    | 40.08749  | 10.77072  | 3.72  | 0.000 | 18.97726             | 61.19772  |
| qSupply  |           |           |       |       |                      |           |
| price    | .5773133  | .1749974  | 3.30  | 0.001 | .2343247             | .9203019  |
| praw     | -.7835496 | .1312414  | -5.97 | 0.000 | -1.040778            | -.5263213 |
| _cons    | -2.550694 | 5.273067  | -0.48 | 0.629 | -12.88571            | 7.784327  |

Endogenous variables:   quantity price

Exogenous variables:   pcompete income praw

The use of 3SLS over 2SLS is essentially an efficiency issue. The coefficients of the demand equation from 3SLS are close to the coefficients from two-stage least squares, and those of the supply equation are identical. The latter case was mentioned earlier for systems with some exactly identified equations. However, even for the demand equation, we do not expect the coefficients to change systematically. What we do expect from three-stage least squares are more precise estimates of the parameters given the validity of our specification and `reg3`'s use of the covariances among the disturbances.

Let's summarize the results. With OLS, we got obviously biased estimates of the parameters. No amount of data would have improved the OLS estimates—they are inconsistent in the face of the violated OLS assumptions. With 2SLS, we obtained consistent estimates of the parameters, and these would have improved with more data. With 3SLS, we obtained consistent estimates of the parameters that are more efficient than those obtained by 2SLS.

◀

## □ Technical note

We noted earlier that the  $R$ -squared was missing from the two-stage estimates of the demand equation. Now we see that the  $R$ -squared is negative for the three-stage estimates of the same equation. How can we have a negative  $R$ -squared?

In most estimators, other than least squares, the  $R$ -squared is no more than a summary measure of the overall in-sample predictive power of the estimator. The computational formula for  $R$ -squared is  $R\text{-squared} = 1 - RSS/TSS$ , where  $RSS$  is the residual sum of squares (sum of squared residuals) and  $TSS$  is the total sum of squared deviations about the mean of the dependent variable. In a standard linear model with a constant, the model from which the  $TSS$  is computed is nested within the full model from which  $RSS$  is computed—they both have a constant term based on the same data. Thus it must be that  $TSS \geq RSS$  and  $R$ -squared is constrained between 0 and 1.

For 2SLS and 3SLS, some of the regressors enter the model as instruments when the parameters are estimated. However, because our goal is to fit the structural model, the actual values, not the instruments for the endogenous right-hand-side variables, are used to determine  $R$ -squared. The model residuals are computed over a different set of regressors from those used to fit the model. The two-

or three-stage estimates are no longer nested within a constant-only model of the dependent variable, and the residual sum of squares is no longer constrained to be smaller than the total sum of squares.

A negative  $R$ -squared in 3SLS should be taken for exactly what it is—an indication that the structural model predicts the dependent variable worse than a constant-only model. Is this a problem? It depends on the application. Three-stage least squares applied to our contrived supply-and-demand example produced good estimates of the known true parameters. Still, the demand equation produced an  $R$ -squared of  $-0.854$ . How do we feel about our parameter estimates? This should be determined by the estimates themselves, their associated standard errors, and the overall model significance. On this basis, negative  $R$ -squared and all, we feel pretty good about all the parameter estimates for both the supply and demand equations. Would we want to make predictions about equilibrium quantity by using the demand equation alone? Probably not. Would we want to make these quantity predictions by using the supply equation? Possibly, because based on in-sample predictions, they seem better than those from the demand equations. However, both the supply and demand estimates are based on limited information. If we are interested in predicting quantity, a reduced-form equation containing all our independent variables would usually be preferred.

□

□ Technical note

As a matter of syntax, we could have specified the supply-and-demand model on one line without using global macros.

```
. reg3 (quantity price pcompete income) (quantity price praw), endog(price)
Three-stage least-squares regression
```

| Equation  | Obs | Parms | RMSE     | "R-sq"  | chi2  | P      |
|-----------|-----|-------|----------|---------|-------|--------|
| quantity  | 49  | 3     | 3.739686 | -0.8540 | 8.68  | 0.0338 |
| 2quantity | 49  | 2     | 1.752501 | 0.5928  | 39.25 | 0.0000 |

|           | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-----------|-----------|-----------|-------|-------|----------------------|-----------|
| quantity  |           |           |       |       |                      |           |
| price     | -1.014345 | .3742036  | -2.71 | 0.007 | -1.74777             | -.2809194 |
| pcompete  | .2647206  | .1464194  | 1.81  | 0.071 | -.0222561            | .5516973  |
| income    | .5299146  | .1898161  | 2.79  | 0.005 | .1578819             | .9019472  |
| _cons     | 40.08749  | 10.77072  | 3.72  | 0.000 | 18.97726             | 61.19772  |
| 2quantity |           |           |       |       |                      |           |
| price     | .5773133  | .1749974  | 3.30  | 0.001 | .2343247             | .9203019  |
| praw      | -.7835496 | .1312414  | -5.97 | 0.000 | -1.040778            | -.5263213 |
| _cons     | -2.550694 | 5.273067  | -0.48 | 0.629 | -12.88571            | 7.784327  |

Endogenous variables: quantity price  
Exogenous variables: pcompete income praw

However, here `reg3` has been forced to create a unique equation name for the supply equation—`2quantity`. Both the supply and demand equations could not be designated as `quantity`, so a number was prefixed to the name for the supply equation.

We could have specified

```
. reg3 (qDemand: quantity price pcompete income) (qSupply: quantity price praw),
> endog(price)
```

and obtained the same results and equation labeling as when we used global macros to hold the equation specifications.

Without explicit equation names, `reg3` always assumes that the dependent variable should be used to name equations. When each equation has a different dependent variable, this rule causes no problems and produces easily interpreted result tables. If the same dependent variable appears in more than one equation, however, `reg3` will create a unique equation name based on the dependent variable name. Because equation names must be used for cross-equation tests, you have more control in this situation if explicit names are placed on the equations.

□

### ► Example 3: Using the full syntax of `reg3`

Klein's (1950) model of the U.S. economy is often used to demonstrate system estimators. It contains several common features that will serve to demonstrate the full syntax of `reg3`. The Klein model is defined by the following seven relationships:

$$c = \beta_0 + \beta_1 p + \beta_2 p1 + \beta_3 w + \epsilon_1 \quad (1)$$

$$i = \beta_4 + \beta_5 p + \beta_6 p1 + \beta_7 k1 + \epsilon_2 \quad (2)$$

$$wp = \beta_8 + \beta_9 y + \beta_{10} y1 + \beta_{11} yr + \epsilon_3 \quad (3)$$

$$y = c + i + g \quad (4)$$

$$p = y - t - wp \quad (5)$$

$$k = k1 + i \quad (6)$$

$$w = wg + wp \quad (7)$$

The variables in the model are listed below. Two sets of variable names are shown. The concise first name uses traditional economics mnemonics, whereas the second name provides more guidance for everyone else. The concise names serve to keep the specification of the model small (and quite understandable to economists).

| Short name | Long name | Variable definition                  | Type       |
|------------|-----------|--------------------------------------|------------|
| c          | consump   | Consumption                          | endogenous |
| p          | profits   | Private industry profits             | endogenous |
| p1         | profits1  | Last year's private industry profits | exogenous  |
| wp         | wagepriv  | Private wage bill                    | endogenous |
| wg         | wagegovt  | Government wage bill                 | exogenous  |
| w          | wagetot   | Total wage bill                      | endogenous |
| i          | invest    | Investment                           | endogenous |
| k1         | capital1  | Last year's level of capital stock   | exogenous  |
| y          | totinc    | Total income/demand                  | endogenous |
| y1         | totinc1   | Last year's total income             | exogenous  |
| g          | govt      | Government spending                  | exogenous  |
| t          | taxnetx   | Indirect bus. taxes + net exports    | exogenous  |
| yr         | year      | Year—1931                            | exogenous  |

Equations (1)–(3) are behavioral and contain explicit disturbances ( $\epsilon_1$ ,  $\epsilon_2$ , and  $\epsilon_3$ ). The remaining equations are identities that specify additional variables in the system and their accounting relationships with the variables in the behavioral equations. Some variables are explicitly endogenous by appearing as dependent variables in (1)–(3). Others are implicitly endogenous as linear combinations that contain other endogenous variables (e.g.,  $w$  and  $p$ ). Still other variables are implicitly exogenous by appearing in the identities but not in the behavioral equations (e.g.,  $wg$  and  $g$ ).

Using the concise names, we can fit Klein’s model with the following command:

```
. use http://www.stata-press.com/data/r11/kleinAbr
. reg3 (c p p1 w) (i p p1 k1) (wp y y1 yr), endog(w p y) exog(t wg g)
Three-stage least-squares regression
```

| Equation | Obs | Parms | RMSE     | "R-sq" | chi2    | P      |
|----------|-----|-------|----------|--------|---------|--------|
| c        | 21  | 3     | .9443305 | 0.9801 | 864.59  | 0.0000 |
| i        | 21  | 3     | 1.446736 | 0.8258 | 162.98  | 0.0000 |
| wp       | 21  | 3     | .7211282 | 0.9863 | 1594.75 | 0.0000 |

|    | Coef. | Std. Err. | z        | P> z  | [95% Conf. Interval] |                     |
|----|-------|-----------|----------|-------|----------------------|---------------------|
| c  |       |           |          |       |                      |                     |
|    | p     | .1248904  | .1081291 | 1.16  | 0.248                | -.0870387 .3368194  |
|    | p1    | .1631439  | .1004382 | 1.62  | 0.104                | -.0337113 .3599992  |
|    | w     | .790081   | .0379379 | 20.83 | 0.000                | .715724 .8644379    |
|    | _cons | 16.44079  | 1.304549 | 12.60 | 0.000                | 13.88392 18.99766   |
| i  |       |           |          |       |                      |                     |
|    | p     | -.0130791 | .1618962 | -0.08 | 0.936                | -.3303898 .3042316  |
|    | p1    | .7557238  | .1529331 | 4.94  | 0.000                | .4559805 1.055467   |
|    | k1    | -.1948482 | .0325307 | -5.99 | 0.000                | -.2586072 -.1310893 |
|    | _cons | 28.17785  | 6.793768 | 4.15  | 0.000                | 14.86231 41.49339   |
| wp |       |           |          |       |                      |                     |
|    | y     | .4004919  | .0318134 | 12.59 | 0.000                | .3381388 .462845    |
|    | y1    | .181291   | .0341588 | 5.31  | 0.000                | .1143411 .2482409   |
|    | yr    | .149674   | .0279352 | 5.36  | 0.000                | .094922 .2044261    |
|    | _cons | 1.797216  | 1.115854 | 1.61  | 0.107                | -.3898181 3.984251  |

```
Endogenous variables: c i wp w p y
Exogenous variables: p1 k1 y1 yr t wg g
```

We used the `exog()` option to identify `t`, `wg`, and `g` as exogenous variables in the system. These variables must be identified because they are part of the system but appear directly in none of the behavioral equations. Without this option, `reg3` would not know they were part of the system. The `endog()` option specifying `w`, `p`, and `y` is also required. Without this information, `reg3` would be unaware that these variables are linear combinations that include endogenous variables.

□ Technical note

Rather than listing additional endogenous and exogenous variables, we could specify the full list of exogenous variables in an `inst()` option,

```
. reg3 (c p p1 w) (i p p1 k1) (wp y y1 yr), inst(g t wg yr p1 k1 y1)
```

or equivalently,

```
. global consequ "(c p p1 w)"
. global inveqn "(i p p1 k1)"
. global wageqn "(wp y y1 yr)"
. global inlist "g t wg yr p1 k1 y1"
. reg3 $consequ $inveqn $wageqn, inst($inlist)
```

Macros and explicit equations can also be mixed in the specification

```
. reg3 $consequ (i p p1 k1) $wageqn, endog(w p y) exog(t wg g)
```

or

```
. reg3 (c p p1 w) $inveqn (wp y y1 yr), endog(w p y) exog(t wg g)
```

Placing the equation-binding parentheses in the global macros was also arbitrary. We could have used

```
. global consump "c p p1 w"
. global invest "i p p1 k1"
. global wagepriv "wp y y1 yr"
. reg3 ($consump) ($invest) ($wagepriv), endog(w p y) exog(t wg g)
```

reg3 is tolerant of all combinations, and these commands will produce identical output.

□

Switching to the full variable names, we can fit Klein's model with the commands below. We will use global macros to store the lists of endogenous and exogenous variables. Again this is not necessary: these lists could have been typed directly on the command line. However, assigning the lists to local macros makes additional processing easier if alternative models are to be fit. We will also use the ireg3 option to produce the iterated estimates.

```
. use http://www.stata-press.com/data/r11/klein
. global consequ "(consump profits profits1 wagetot)"
. global inveqn "(invest profits profits1 capital1)"
. global wageqn "(wagepriv totinc totinc1 year)"
. global enlist "wagetot profits totinc"
. global exlist "taxnetx wagegovt govt"
. reg3 $consequ $inveqn $wageqn, endog($enlist) exog($exlist) ireg3
Iteration 1: tolerance = .3712549
Iteration 2: tolerance = .1894712
Iteration 3: tolerance = .1076401
(output omitted)
Iteration 24: tolerance = 7.049e-07
Three-stage least-squares regression, iterated
```

| Equation | Obs | Parms | RMSE     | "R-sq" | chi2    | P      |
|----------|-----|-------|----------|--------|---------|--------|
| consump  | 21  | 3     | .9565088 | 0.9796 | 970.31  | 0.0000 |
| invest   | 21  | 3     | 2.134327 | 0.6209 | 56.78   | 0.0000 |
| wagepriv | 21  | 3     | .7782334 | 0.9840 | 1312.19 | 0.0000 |

|                                                                            | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------------------------------------------------------------------------|-----------|-----------|-------|-------|----------------------|-----------|
| consump                                                                    |           |           |       |       |                      |           |
| profits                                                                    | .1645096  | .0961979  | 1.71  | 0.087 | -.0240348            | .3530539  |
| profits1                                                                   | .1765639  | .0901001  | 1.96  | 0.050 | -.0000291            | .3531569  |
| wagetot                                                                    | .7658011  | .0347599  | 22.03 | 0.000 | .6976729             | .8339294  |
| _cons                                                                      | 16.55899  | 1.224401  | 13.52 | 0.000 | 14.15921             | 18.95877  |
| invest                                                                     |           |           |       |       |                      |           |
| profits                                                                    | -.3565316 | .2601568  | -1.37 | 0.171 | -.8664296            | .1533664  |
| profits1                                                                   | 1.011299  | .2487745  | 4.07  | 0.000 | .5237098             | 1.498888  |
| capital1                                                                   | -.2602    | .0508694  | -5.12 | 0.000 | -.3599022            | -.1604978 |
| _cons                                                                      | 42.89629  | 10.59386  | 4.05  | 0.000 | 22.13271             | 63.65987  |
| wagepriv                                                                   |           |           |       |       |                      |           |
| totinc                                                                     | .3747792  | .0311027  | 12.05 | 0.000 | .3138191             | .4357394  |
| totinc1                                                                    | .1936506  | .0324018  | 5.98  | 0.000 | .1301443             | .257157   |
| year                                                                       | .1679262  | .0289291  | 5.80  | 0.000 | .1112263             | .2246261  |
| _cons                                                                      | 2.624766  | 1.195559  | 2.20  | 0.028 | .2815124             | 4.968019  |
| Endogenous variables:  consump invest wagepriv wagetot profits totinc      |           |           |       |       |                      |           |
| Exogenous variables:  profits1 capital1 totinc1 year taxnetx wagegovt govt |           |           |       |       |                      |           |

◀

► Example 4: Constraints with reg3

As a simple example of constraints, (1) above may be rewritten with both wages explicitly appearing (rather than as a variable containing the sum). Using the longer variable names, we have

$$\text{consump} = \beta_0 + \beta_1 \text{profits} + \beta_2 \text{profits1} + \beta_3 \text{wagepriv} + \beta_{12} \text{wagegovt} + \epsilon_1$$

To retain the effect of the identity in (7), we need  $\beta_3 = \beta_{12}$  as a constraint on the system. We obtain this result by defining the constraint in the usual way and then specifying its use in `reg3`. Because `reg3` is a system estimator, we will need to use the full equation syntax of `constraint`. The assumption that the following commands are entered after the model above has been estimated. We are simply changing the definition of the consumption equation (`consump`) and adding a constraint on two of its parameters. The rest of the model definition is carried forward.

```
. global conseqn "(consump profits profits1 wagepriv wagegovt)"
. constraint 1 [consump]wagepriv = [consump]wagegovt
. reg3 $conseqn $inveqn $wageqn, endog($enlist) exog($exlist) constr(1) ireg3
note: additional endogenous variables not in the system have no effect
 and are ignored: wagetot
Iteration 1: tolerance = .3712547
Iteration 2: tolerance = .189471
Iteration 3: tolerance = .10764
(output omitted)
Iteration 24: tolerance = 7.049e-07
```

Three-stage least-squares regression, iterated

| Equation | Obs | Parms | RMSE     | "R-sq" | chi2    | P      |
|----------|-----|-------|----------|--------|---------|--------|
| consump  | 21  | 3     | .9565086 | 0.9796 | 970.31  | 0.0000 |
| invest   | 21  | 3     | 2.134326 | 0.6209 | 56.78   | 0.0000 |
| wagepriv | 21  | 3     | .7782334 | 0.9840 | 1312.19 | 0.0000 |

( 1) [consump]wagepriv - [consump]wagegovt = 0

|          | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| consump  |           |           |       |       |                      |           |
| profits  | .1645097  | .0961978  | 1.71  | 0.087 | -.0240346            | .353054   |
| profits1 | .1765639  | .0901001  | 1.96  | 0.050 | -.0000291            | .3531568  |
| wagepriv | .7658012  | .0347599  | 22.03 | 0.000 | .6976729             | .8339294  |
| wagegovt | .7658012  | .0347599  | 22.03 | 0.000 | .6976729             | .8339294  |
| _cons    | 16.55899  | 1.224401  | 13.52 | 0.000 | 14.1592              | 18.95877  |
| invest   |           |           |       |       |                      |           |
| profits  | -.3565311 | .2601567  | -1.37 | 0.171 | -.8664288            | .1533666  |
| profits1 | 1.011298  | .2487744  | 4.07  | 0.000 | .5237096             | 1.498887  |
| capital1 | -.2601999 | .0508694  | -5.12 | 0.000 | -.359902             | -.1604977 |
| _cons    | 42.89626  | 10.59386  | 4.05  | 0.000 | 22.13269             | 63.65984  |
| wagepriv |           |           |       |       |                      |           |
| totinc   | .3747792  | .0311027  | 12.05 | 0.000 | .313819              | .4357394  |
| totinc1  | .1936506  | .0324018  | 5.98  | 0.000 | .1301443             | .257157   |
| year     | .1679262  | .0289291  | 5.80  | 0.000 | .1112263             | .2246261  |
| _cons    | 2.624766  | 1.195559  | 2.20  | 0.028 | .281512              | 4.968019  |

Endogenous variables: consump invest wagepriv wagetot profits totinc

Exogenous variables: profits1 wagegovt capital1 totinc1 year taxnetx govt

As expected, none of the parameter or standard error estimates has changed from the previous estimates (before the seventh significant digit). We have simply decomposed the total wage variable into its two parts and constrained the coefficients on these parts. The warning about additional endogenous variables was just `reg3`'s way of letting us know that we had specified some information that was irrelevant to the estimation of the system. We had left the `wagetot` variable in our `endog` macro. It does not mean anything to the system to specify `wagetot` as endogenous because it is no longer in the system. That's fine with `reg3` and fine for our current purposes.

We can also impose constraints across the equations. For example, the admittedly meaningless constraint of requiring `profits` to have the same effect in both the consumption and investment equations could be imposed. Retaining the constraint on the wage coefficients, we would estimate this constrained system.

```
. constraint 2 [consump]profits = [invest]profits
. reg3 $conseqn $inveqn $wageqn, endog($enlist) exog($exlist) constr(1 2) ireg3
note: additional endogenous variables not in the system have no effect
 and are ignored: wagetot

Iteration 1: tolerance = .1427927
Iteration 2: tolerance = .032539
Iteration 3: tolerance = .00307811
Iteration 4: tolerance = .00016903
Iteration 5: tolerance = .00003409
Iteration 6: tolerance = 7.763e-06
Iteration 7: tolerance = 9.240e-07
```

Three-stage least-squares regression, iterated

| Equation | Obs | Parms | RMSE     | "R-sq" | chi2    | P      |
|----------|-----|-------|----------|--------|---------|--------|
| consump  | 21  | 3     | .9504669 | 0.9798 | 1019.54 | 0.0000 |
| invest   | 21  | 3     | 1.247066 | 0.8706 | 144.57  | 0.0000 |
| wagepriv | 21  | 3     | .7225276 | 0.9862 | 1537.45 | 0.0000 |

( 1) [consump]wagepriv - [consump]wagegovt = 0

( 2) [consump]profits - [invest]profits = 0

|          | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| consump  |           |           |       |       |                      |           |
| profits  | .1075413  | .0957767  | 1.12  | 0.262 | -.0801777            | .2952602  |
| profits1 | .1712756  | .0912613  | 1.88  | 0.061 | -.0075932            | .3501444  |
| wagepriv | .798484   | .0340876  | 23.42 | 0.000 | .7316734             | .8652946  |
| wagegovt | .798484   | .0340876  | 23.42 | 0.000 | .7316734             | .8652946  |
| _cons    | 16.2521   | 1.212157  | 13.41 | 0.000 | 13.87631             | 18.62788  |
| invest   |           |           |       |       |                      |           |
| profits  | .1075413  | .0957767  | 1.12  | 0.262 | -.0801777            | .2952602  |
| profits1 | .6443378  | .1058682  | 6.09  | 0.000 | .43684               | .8518356  |
| capital1 | -.1766669 | .0261889  | -6.75 | 0.000 | -.2279962            | -.1253375 |
| _cons    | 24.31931  | 5.284325  | 4.60  | 0.000 | 13.96222             | 34.6764   |
| wagepriv |           |           |       |       |                      |           |
| totinc   | .4014106  | .0300552  | 13.36 | 0.000 | .3425035             | .4603177  |
| totinc1  | .1775359  | .0321583  | 5.52  | 0.000 | .1145068             | .240565   |
| year     | .1549211  | .0282291  | 5.49  | 0.000 | .099593              | .2102492  |
| _cons    | 1.959788  | 1.14467   | 1.71  | 0.087 | -.2837242            | 4.203299  |

Endogenous variables: consump invest wagepriv wagetot profits totinc

Exogenous variables: profits1 wagegovt capital1 totinc1 year taxnetx govt

□ Technical note

Identification in a system of simultaneous equations involves the notion that there is enough information to estimate the parameters of the model given the specified functional form. Under-identification usually manifests itself as one matrix in the 3SLS computations. The most commonly violated order condition for 2SLS or 3SLS involves the number of endogenous and exogenous variables. There must be at least as many noncollinear exogenous variables in the remaining system as there are endogenous right-hand-side variables in an equation. This condition must hold for each structural equation in the system.

- Put as a set of rules the following:
1. Count the number of right-hand-side endogenous variables in an equation and call this  $m_i$ .
  2. Count the number of exogenous variables in the same equation and call this  $k_i$ .
  3. Count the total number of exogenous variables in all the structural equations plus any additional variables specified in an `exog()` or `inst()` option and call this  $K$ .
  4. If  $m_i > (K - k_i)$  for any structural equation ( $i$ ), then the system is underidentified and cannot be estimated by 3SLS.

We are also possibly in trouble if any of the exogenous variables are linearly dependent. We must have  $m_i$  linearly independent variables among the exogenous variables represented by  $(K - k_i)$ .



The complete conditions for identification involve rank-order conditions on several matrices. For a full treatment, see [Theil \(1971\)](#) or [Greene \(2008, 381–383\)](#).



## Saved results

`reg3` saves the following in `e()`:

### Scalars

|                           |                                                                       |
|---------------------------|-----------------------------------------------------------------------|
| <code>e(N)</code>         | number of observations                                                |
| <code>e(k)</code>         | number of parameters                                                  |
| <code>e(k_eq)</code>      | number of equations                                                   |
| <code>e(k_autoCns)</code> | number of base, empty, and omitted constraints                        |
| <code>e(mss_#)</code>     | model sum of squares for equation #                                   |
| <code>e(df_m#)</code>     | model degrees of freedom for equation #                               |
| <code>e(rss_#)</code>     | residual sum of squares for equation #                                |
| <code>e(df_r)</code>      | residual degrees of freedom (small)                                   |
| <code>e(r2_#)</code>      | $R$ -squared for equation #                                           |
| <code>e(F_#)</code>       | $F$ statistic for equation # (small)                                  |
| <code>e(rmse_#)</code>    | root mean squared error for equation #                                |
| <code>e(dfk2_adj)</code>  | divisor used with VCE when <code>dfk2</code> specified                |
| <code>e(ll)</code>        | log likelihood                                                        |
| <code>e(chi2_#)</code>    | $\chi^2$ for equation #                                               |
| <code>e(p_#)</code>       | significance for equation #                                           |
| <code>e(k_eform)</code>   | number of leading equations appropriate for <code>eform</code> output |
| <code>e(cons_#)</code>    | 1 when equation # has a constant, 0 otherwise                         |
| <code>e(rank)</code>      | rank of <code>e(V)</code>                                             |
| <code>e(ic)</code>        | number of iterations                                                  |

### Macros

|                              |                                                                |
|------------------------------|----------------------------------------------------------------|
| <code>e(cmd)</code>          | <code>reg3</code>                                              |
| <code>e(cmdline)</code>      | command as typed                                               |
| <code>e(depvar)</code>       | names of dependent variables                                   |
| <code>e(exog)</code>         | names of exogenous variables                                   |
| <code>e(endog)</code>        | names of endogenous variables                                  |
| <code>e(eqnames)</code>      | names of equations                                             |
| <code>e(corr)</code>         | correlation structure                                          |
| <code>e(wtype)</code>        | weight type                                                    |
| <code>e(wexp)</code>         | weight expression                                              |
| <code>e(method)</code>       | 3sls, 2sls, ols, sure, or mvreg                                |
| <code>e(small)</code>        | small                                                          |
| <code>e(properties)</code>   | b V                                                            |
| <code>e(predict)</code>      | program used to implement <code>predict</code>                 |
| <code>e(marginsok)</code>    | predictions allowed by <code>margins</code>                    |
| <code>e(marginsnotok)</code> | predictions disallowed by <code>margins</code>                 |
| <code>e(asbalanced)</code>   | factor variables <code>fvset</code> as <code>asbalanced</code> |
| <code>e(asobserved)</code>   | factor variables <code>fvset</code> as <code>asobserved</code> |

## Matrices

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>e(b)</code>     | coefficient vector                           |
| <code>e(Cns)</code>   | constraints matrix                           |
| <code>e(Sigma)</code> | $\widehat{\Sigma}$ matrix                    |
| <code>e(V)</code>     | variance–covariance matrix of the estimators |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`reg3` is implemented as an ado-file.

The most concise way to represent a system of equations for 3SLS requires thinking of the individual equations and their associated data as being stacked. `reg3` does not expect the data in this format, but it is a convenient shorthand. The system could then be formulated as

$$\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_M \end{bmatrix} = \begin{bmatrix} \mathbf{Z}_1 & 0 & \dots & 0 \\ 0 & \mathbf{Z}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbf{Z}_M \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_M \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_M \end{bmatrix}$$

In full matrix notation, this is just

$$\mathbf{y} = \mathbf{Z}\mathbf{B} + \boldsymbol{\epsilon}$$

The  $\mathbf{Z}$  elements in these matrices represent both the endogenous and the exogenous right-hand-side variables in the equations.

Also assume that there will be correlation between the disturbances of the equations so that

$$E(\boldsymbol{\epsilon}\boldsymbol{\epsilon}') = \boldsymbol{\Sigma}$$

where the disturbances are further assumed to have an expected value of 0;  $E(\boldsymbol{\epsilon}) = 0$ .

The first stage of 3SLS regression requires developing instrumented values for the endogenous variables in the system. These values can be derived as the predictions from a linear regression of each endogenous regressor on all exogenous variables in the system or, more succinctly, as the projection of each regressor through the projection matrix of all exogenous variables onto the regressors. Designating the set of all exogenous variables as  $\mathbf{X}$  results in

$$\widehat{\mathbf{z}}_i = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{z}_i \quad \text{for each } i$$

Taken collectively, these  $\widehat{\mathbf{Z}}$  contain the instrumented values for all the regressors. They take on the actual values for the exogenous variables and first-stage predictions for the endogenous variables. Given these instrumented variables, a generalized least squares (GLS) or [Aitken \(1935\)](#) estimator can be formed for the parameters of the system

$$\widehat{\mathbf{B}} = \left\{ \widehat{\mathbf{Z}}'(\boldsymbol{\Sigma}^{-1} \otimes \mathbf{I})\widehat{\mathbf{Z}} \right\}^{-1} \widehat{\mathbf{Z}}'(\boldsymbol{\Sigma}^{-1} \otimes \mathbf{I})\mathbf{y}$$

All that remains is to obtain a consistent estimator for  $\Sigma$ . This estimate can be formed from the residuals of 2SLS estimates of each equation in the system. Alternately, and identically, the residuals can be computed from the estimates formed by taking  $\Sigma$  to be an identity matrix. This maintains the full system of coefficients and allows constraints to be applied when the residuals are computed.

If we take  $\mathbf{E}$  to be the matrix of residuals from these estimates, a consistent estimate of  $\Sigma$  is

$$\hat{\Sigma} = \frac{\mathbf{E}'\mathbf{E}}{n}$$

where  $n$  is the number of observations in the sample. An alternative divisor for this estimate can be obtained with the `dfk` option as outlined under options.

With the estimate of  $\hat{\Sigma}$  placed into the GLS estimating equation,

$$\hat{\mathbf{B}} = \left\{ \hat{\mathbf{Z}}'(\hat{\Sigma}^{-1} \otimes \mathbf{I})\hat{\mathbf{Z}} \right\}^{-1} \hat{\mathbf{Z}}'(\hat{\Sigma}^{-1} \otimes \mathbf{I})\mathbf{y}$$

is the 3SLS estimates of the system parameters.

The asymptotic variance–covariance matrix of the estimator is just the standard formulation for a GLS estimator

$$\mathbf{V}_{\hat{\mathbf{B}}} = \left\{ \hat{\mathbf{Z}}'(\hat{\Sigma}^{-1} \otimes \mathbf{I})\hat{\mathbf{Z}} \right\}^{-1}$$

Iterated 3SLS estimates can be obtained by computing the residuals from the three-stage parameter estimates, using these to formulate a new  $\hat{\Sigma}$ , and recomputing the parameter estimates. This process is repeated until the estimates  $\hat{\mathbf{B}}$  converge—if they converge. Convergence is not guaranteed. When estimating a system by SURE, these iterated estimates will be the maximum likelihood estimates for the system. The iterated solution can also be used to produce estimates that are invariant to choice of system and restriction parameterization for many linear systems under full 3SLS.

The exposition above follows the parallel developments in [Greene \(2008\)](#) and [Davidson and MacKinnon \(1993\)](#).

Henri Theil (1924–2000) was born in Amsterdam and awarded a PhD in 1951 by the University of Amsterdam. He researched and taught econometric theory, statistics, microeconomics, macroeconomic modeling, and economic forecasting, and policy at what is now Erasmus University in Rotterdam, the University of Chicago, and the University of Florida. Theil's many specific contributions include work on 2SLS and 3SLS, inequality and concentration, and consumer demand.

## References

- Aitken, A. C. 1935. On least squares and linear combination of observations. *Proceedings of the Royal Society of Edinburgh* 55: 42–48.
- Bewley, R. 2000. Mr. Henri Theil: An interview with the International Journal of Forecasting. *International Journal of Forecasting* 16: 1–16.
- Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.

Klein, L. R. 1950. *Economic Fluctuations in the United States 1921-1941*. New York: Wiley.

Nichols, A. 2007. [Causal inference with observational data](#). *Stata Journal* 7: 507–541.

Poi, B. P. 2006. [Jackknife instrumental variables estimation in Stata](#). *Stata Journal* 6: 364–376.

Theil, H. 1971. *Principles of Econometrics*. New York: Wiley.

Weesie, J. 1999. [sg121: Seemingly unrelated estimation and the cluster-adjusted sandwich estimator](#). *Stata Technical Bulletin* 52: 34–47. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 231–248. College Station, TX: Stata Press.

Zellner, A., and H. Theil. 1962. Three stage least squares: Simultaneous estimate of simultaneous equations. *Econometrica* 29: 54–78.

## Also see

[R] [reg3 postestimation](#) — Postestimation tools for reg3

[R] [ivregress](#) — Single-equation instrumental-variables regression

[R] [mvreg](#) — Multivariate regression

[R] [nlsur](#) — Estimation of nonlinear systems of equations

[R] [regress](#) — Linear regression

[R] [sureg](#) — Zellner’s seemingly unrelated regression

[U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `reg3`:

| command                   | description                                                                                         |
|---------------------------|-----------------------------------------------------------------------------------------------------|
| * <a href="#">estat</a>   | AIC, BIC, VCE, and estimation sample summary                                                        |
| <a href="#">estimates</a> | cataloging estimation results                                                                       |
| <a href="#">hausman</a>   | Hausman's specification test                                                                        |
| <a href="#">lincom</a>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <a href="#">margins</a>   | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <a href="#">nlcom</a>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <a href="#">predict</a>   | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <a href="#">predictnl</a> | point estimates, standard errors, testing, and inference for generalized predictions                |
| <a href="#">test</a>      | Wald tests of simple and composite linear hypotheses                                                |
| <a href="#">testnl</a>    | Wald tests of nonlinear hypotheses                                                                  |

\*`estat ic` is not appropriate after `reg3`, `2sls`.

See the corresponding entries in the *Base Reference Manual* for details.

Syntax for predict

```
predict [type] newvar [if] [in] [, equation(eqno[,eqno]) statistic]
```

| statistic                      | description                                                |
|--------------------------------|------------------------------------------------------------|
| Main                           |                                                            |
| <code>xb</code>                | linear prediction; the default                             |
| <code>stdp</code>              | standard error of the linear prediction                    |
| <code><u>r</u>esiduals</code>  | residuals                                                  |
| <code><u>d</u>ifference</code> | difference between the linear predictions of two equations |
| <code>std<u>d</u>p</code>      | standard error of the difference in linear predictions     |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

`equation(eqno [, eqno])` specifies to which equation you are referring.

`equation()` is filled in with one *eqno* for the `xb`, `stdp`, and `residuals` options. `equation(#1)` would mean the calculation is to be made for the first equation, `equation(#2)` would mean the second, and so on. You could also refer to the equations by their names. `equation(income)` would refer to the equation named `income` and `equation(hours)` to the equation named `hours`.

If you do not specify `equation()`, results are the same as if you specified `equation(#1)`.

`difference` and `stdp` refer to between-equation concepts. To use these options, you must specify two equations, e.g., `equation(#1,#2)` or `equation(income, hours)`. When two equations must be specified, `equation()` is required.

`xb`, the default, calculates the linear prediction (fitted values)—the prediction of  $\mathbf{x}_j\mathbf{b}$  for the specified equation.

`stdp` calculates the standard error of the prediction for the specified equation. It can be thought of as the standard error of the predicted expected value or mean for the observation's covariate pattern. The standard error of the prediction is also referred to as the standard error of the fitted value.

`residuals` calculates the residuals.

`difference` calculates the difference between the linear predictions of two equations in the system. With `equation(#1,#2)`, `difference` computes the prediction of `equation(#1)` minus the prediction of `equation(#2)`.

`stdp` is allowed only after you have previously fit a multiple-equation model. The standard error of the difference in linear predictions ( $\mathbf{x}_{1j}\mathbf{b} - \mathbf{x}_{2j}\mathbf{b}$ ) between equations 1 and 2 is calculated.

For more information on using `predict` after multiple-equation estimation commands, see [\[R\] predict](#).

## Remarks

### ► Example 1

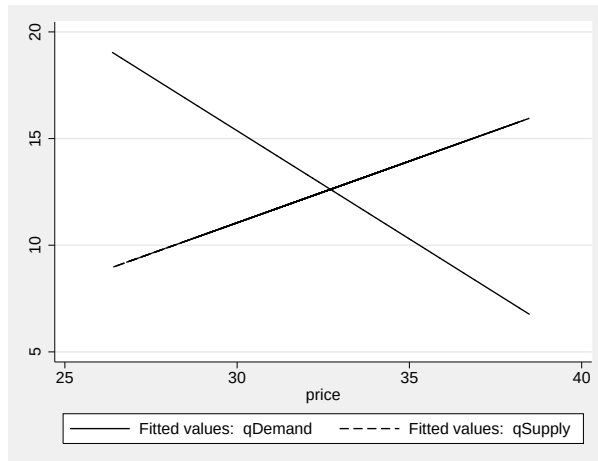
In [example 2](#) of [\[R\] reg3](#), we fit a simple supply-and-demand model. Here we obtain the fitted supply and demand curves assuming that the exogenous regressors equal their sample means. We first replace each of the three exogenous regressors with their sample means, then we call `predict` to obtain the predictions.

```
. use http://www.stata-press.com/data/r11/supDem
. global demand "(qDemand: quantity price pcompete income)"
. global supply "(qSupply: quantity price praw)"
. reg3 $demand $supply, endog(price)
 (output omitted)
. summarize pcompete, meanonly
. replace pcompete = r(mean)
(49 real changes made)
```

```

. summarize income, meanonly
. replace income = r(mean)
(49 real changes made)
. summarize praw, meanonly
. replace praw = r(mean)
(49 real changes made)
. predict demand, equation(qDemand)
(option xb assumed; fitted values)
. predict supply, equation(qSupply)
(option xb assumed; fitted values)
. graph twoway line demand price, sort || line supply price

```



As we would expect based on economic theory, the demand curve slopes downward while the supply curve slopes upward. With the exogenous variables at their mean levels, the equilibrium price and quantity are slightly less than 33 and 13, respectively.

◀

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

The computational formulas for the statistics produced by `predict` can be found in [\[R\] predict](#) and [\[R\] regress postestimation](#).

## Also see

[\[R\] reg3](#) — Three-stage estimation for systems of simultaneous equations

[\[U\] 20 Estimation and postestimation commands](#)

Syntax

```
regress depvar [indepvars] [if] [in] [weight] [, options]
```

| <i>options</i>                       | description                                                                                                                                                                                    |
|--------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                |                                                                                                                                                                                                |
| <code>noconstant</code>              | suppress constant term                                                                                                                                                                         |
| <code>hascons</code>                 | has user-supplied constant                                                                                                                                                                     |
| <code>tsscons</code>                 | compute total sum of squares with constant; seldom used                                                                                                                                        |
| SE/Robust                            |                                                                                                                                                                                                |
| <code>vce(<i>vcetype</i>)</code>     | <i>vcetype</i> may be <code>ols</code> , <code>robust</code> , <code>cluster <i>clustvar</i></code> , <code>bootstrap</code> , <code>jackknife</code> , <code>hc2</code> , or <code>hc3</code> |
| Reporting                            |                                                                                                                                                                                                |
| <code>level(#)</code>                | set confidence level; default is <code>level(95)</code>                                                                                                                                        |
| <code>beta</code>                    | report standardized beta coefficients                                                                                                                                                          |
| <code>eform(<i>string</i>)</code>    | report exponentiated coefficients and label as <i>string</i>                                                                                                                                   |
| <code>depname(<i>varname</i>)</code> | substitute dependent variable name; programmer's option                                                                                                                                        |
| <code>display_options</code>         | control spacing and display of omitted variables and base and empty cells                                                                                                                      |
| <sup>†</sup> <code>noheader</code>   | suppress table header                                                                                                                                                                          |
| <sup>†</sup> <code>notable</code>    | suppress table header                                                                                                                                                                          |
| <sup>†</sup> <code>plus</code>       | make table extendable                                                                                                                                                                          |
| <sup>†</sup> <code>mse1</code>       | force mean squared error to 1                                                                                                                                                                  |
| <sup>†</sup> <code>coeflegend</code> | display coefficients' legend instead of coefficient table                                                                                                                                      |

<sup>†</sup>`noheader`, `notable`, `plus`, `mse1`, and `coeflegend` do not appear in the dialog box.

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

`bootstrap`, `by`, `fracpoly`, `jackknife`, `mfp`, `mi estimate`, `nestreg`, `rolling`, `statsby`, `stepwise`, and `svy` are allowed; see [U] 11.1.10 Prefix commands.

`vce(bootstrap)` and `vce(jackknife)` are not allowed with the `mi estimate` prefix.

Weights are not allowed with the `bootstrap` prefix.

`aweight`s are not allowed with the `jackknife` prefix.

`hascons`, `tsscons`, `vce()`, `beta`, `noheader`, `notable`, `plus`, `depname()`, `mse1`, and `weights` are not allowed with the `svy` prefix.

`aweight`s, `fweight`s, `ilweight`s, and `pweight`s are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.



# Menu

Statistics > Linear models and related > Linear regression

## Description

`regress` fits a model of *depvar* on *indepvars* using linear regression.

Here is a short list of other regression commands that may be of interest. See [\[1\] estimation commands](#) for a complete list.

| command                 | entry                           | description                                                       |
|-------------------------|---------------------------------|-------------------------------------------------------------------|
| <code>areg</code>       | <a href="#">[R] areg</a>        | an easier way to fit regressions with many dummy variables        |
| <code>arch</code>       | <a href="#">[TS] arch</a>       | regression models with ARCH errors                                |
| <code>arima</code>      | <a href="#">[TS] arima</a>      | ARIMA models                                                      |
| <code>boxcox</code>     | <a href="#">[R] boxcox</a>      | Box–Cox regression models                                         |
| <code>cnsreg</code>     | <a href="#">[R] cnsreg</a>      | constrained linear regression                                     |
| <code>eivreg</code>     | <a href="#">[R] eivreg</a>      | errors-in-variables regression                                    |
| <code>frontier</code>   | <a href="#">[R] frontier</a>    | stochastic frontier models                                        |
| <code>gmm</code>        | <a href="#">[R] gmm</a>         | generalized method of moments estimation                          |
| <code>heckman</code>    | <a href="#">[R] heckman</a>     | Heckman selection model                                           |
| <code>intreg</code>     | <a href="#">[R] intreg</a>      | interval regression                                               |
| <code>ivregress</code>  | <a href="#">[R] ivregress</a>   | single-equation instrumental-variables regression                 |
| <code>ivtobit</code>    | <a href="#">[R] ivtobit</a>     | tobit regression with endogenous variables                        |
| <code>newey</code>      | <a href="#">[TS] newey</a>      | regression with Newey–West standard errors                        |
| <code>nl</code>         | <a href="#">[R] nl</a>          | nonlinear least-squares estimation                                |
| <code>nlstur</code>     | <a href="#">[R] nlstur</a>      | estimation of nonlinear systems of equations                      |
| <code>qreg</code>       | <a href="#">[R] qreg</a>        | quantile (including median) regression                            |
| <code>reg3</code>       | <a href="#">[R] reg3</a>        | three-stage least-squares (3SLS) regression                       |
| <code>rreg</code>       | <a href="#">[R] rreg</a>        | a type of robust regression                                       |
| <code>sureg</code>      | <a href="#">[R] sureg</a>       | seemingly unrelated regression                                    |
| <code>tobit</code>      | <a href="#">[R] tobit</a>       | tobit regression                                                  |
| <code>treatreg</code>   | <a href="#">[R] treatreg</a>    | treatment-effects model                                           |
| <code>truncreg</code>   | <a href="#">[R] truncreg</a>    | truncated regression                                              |
| <code>xtabond</code>    | <a href="#">[XT] xtabond</a>    | Arellano–Bond linear dynamic panel-data estimation                |
| <code>xtdpd</code>      | <a href="#">[XT] xtdpd</a>      | linear dynamic panel-data estimation                              |
| <code>xtfrontier</code> | <a href="#">[XT] xtfrontier</a> | panel-data stochastic frontier models                             |
| <code>xtgls</code>      | <a href="#">[XT] xtgls</a>      | panel-data GLS models                                             |
| <code>xthtaylor</code>  | <a href="#">[XT] xthtaylor</a>  | Hausman–Taylor estimator for error-components models              |
| <code>xtintreg</code>   | <a href="#">[XT] xtintreg</a>   | panel-data interval regression models                             |
| <code>xtivreg</code>    | <a href="#">[XT] xtivreg</a>    | panel-data instrumental-variables (2SLS) regression               |
| <code>xtpcse</code>     | <a href="#">[XT] xtpcse</a>     | linear regression with panel-corrected standard errors            |
| <code>xtreg</code>      | <a href="#">[XT] xtreg</a>      | fixed- and random-effects linear models                           |
| <code>xtregar</code>    | <a href="#">[XT] xtregar</a>    | fixed- and random-effects linear models with an AR(1) disturbance |
| <code>xttobit</code>    | <a href="#">[XT] xttobit</a>    | panel-data tobit models                                           |

## Options

## Model

`noconstant`; see [R] [estimation options](#).

`hascons` indicates that a user-defined constant or its equivalent is specified among the independent variables in *indepvars*. Some caution is recommended when specifying this option, as resulting estimates may not be as accurate as they otherwise would be. Use of this option requires “sweeping” the constant last, so the moment matrix must be accumulated in absolute rather than deviation form. This option may be safely specified when the means of the dependent and independent variables are all reasonable and there is not much collinearity between the independent variables. The best procedure is to view `hascons` as a reporting option—estimate with and without `hascons` and verify that the coefficients and standard errors of the variables not affected by the identity of the constant are unchanged.

`tsscons` forces the total sum of squares to be computed as though the model has a constant, that is, as deviations from the mean of the dependent variable. This is a rarely used option that has an effect only when specified with `noconstant`. It affects the total sum of squares and all results derived from the total sum of squares.

## SE/Robust

`vce(vctype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

`vce(ols)`, the default, uses the standard variance estimator for ordinary least-squares regression.

`regress` also allows the following:

`vce(hc2)` and `vce(hc3)` specify an alternative bias correction for the robust variance calculation. `vce(hc2)` and `vce(hc3)` may not be specified with `svy` prefix. In the unclustered case, `vce(robust)` uses  $\hat{\sigma}_j^2 = \{n/(n-k)\}u_j^2$  as an estimate of the variance of the  $j$ th observation, where  $u_j$  is the calculated residual and  $n/(n-k)$  is included to improve the overall estimate’s small-sample properties.

`vce(hc2)` instead uses  $u_j^2/(1 - h_{jj})$  as the observation’s variance estimate, where  $h_{jj}$  is the diagonal element of the hat (projection) matrix. This estimate is unbiased if the model really is homoskedastic. `vce(hc2)` tends to produce slightly more conservative confidence intervals.

`vce(hc3)` uses  $u_j^2/(1 - h_{jj})^2$  as suggested by [Davidson and MacKinnon \(1993\)](#), who report that this method tends to produce better results when the model really is heteroskedastic. `vce(hc3)` produces confidence intervals that tend to be even more conservative.

See [Davidson and MacKinnon \(1993, 554–556\)](#) and [Angrist and Pischke \(2009, 294–308\)](#) for more discussion on these two bias corrections.

## Reporting

`level(#)`; see [R] [estimation options](#).

`beta` asks that standardized beta coefficients be reported instead of confidence intervals. The beta coefficients are the regression coefficients obtained by first standardizing all variables to have a mean of 0 and a standard deviation of 1. `beta` may not be specified with `vce(cluster clustvar)` or the `svy` prefix.

`eform(string)` is used only in programs and ado-files that use `regress` to fit models other than linear regression. `eform()` specifies that the coefficient table be displayed in exponentiated form as defined in [R] [maximize](#) and that *string* be used to label the exponentiated coefficients in the table.

`depname(varname)` is used only in programs and ado-files that use `regress` to fit models other than linear regression. `depname()` may be specified only at estimation time. *varname* is recorded as the identity of the dependent variable, even though the estimates are calculated using *depvar*. This method affects the labeling of the output—not the results calculated—but could affect subsequent calculations made by `predict`, where the residual would be calculated as deviations from *varname* rather than *depvar*. `depname()` is most typically used when *depvar* is a temporary variable (see [P] [macro](#)) used as a proxy for *varname*.

*display\_options*: [noomitted](#), [vsquish](#), [noemptycells](#), [baselevels](#), [allbaselevels](#); see [R] [estimation options](#).

The following options are available with `regress` but are not shown in the dialog box:

`noheader` suppresses the display of the ANOVA table and summary statistics at the top of the output; only the coefficient table is displayed. This option is often used in programs and ado-files.

`notable` suppresses display of the coefficient table.

`plus` specifies that the output table be made extendable. This option is often used in programs and ado-files.

`mse1` is used only in programs and ado-files that use `regress` to fit models other than linear regression and is not allowed with the `svy` prefix. `mse1` sets the mean squared error to 1, forcing the variance–covariance matrix of the estimators to be  $(\mathbf{X}'\mathbf{D}\mathbf{X})^{-1}$  (see [Methods and formulas](#) below) and affecting calculated standard errors. Degrees of freedom for *t* statistics are calculated as *n* rather than *n* − *k*.

`coeflegend`; see [R] [estimation options](#).

## Remarks

Remarks are presented under the following headings:

*Ordinary least squares*

*Treatment of the constant*

*Robust standard errors*

*Weighted regression*

*Instrumental variables and two-stage least-squares regression*

`regress` performs linear regression, including ordinary least squares and weighted least squares. For a general discussion of linear regression, see [Draper and Smith \(1998\)](#), [Greene \(2008\)](#), or [Kmenta \(1997\)](#).

See [Wooldridge \(2009\)](#) for an excellent treatment of estimation, inference, interpretation, and specification testing in linear regression models. This presentation stands out for its clarification of the statistical issues, as opposed to the algebraic issues. See [Wooldridge \(2002, chap. 4\)](#) for a more advanced discussion along the same lines.

See [Hamilton \(2009, chap. 6\)](#) and [Cameron and Trivedi \(2009, chap. 3\)](#) for an introduction to linear regression using Stata. [Dohoo, Martin, and Stryhn \(2003\)](#) discuss linear regression using examples from epidemiology, and Stata datasets and do-files used in the text are available. [Cameron and Trivedi \(2009\)](#) discuss linear regression using econometric examples with Stata.

Chatterjee and Hadi (2006) explain regression analysis by using examples containing typical problems that you might encounter when performing exploratory data analysis. We also recommend Weisberg (2005), who emphasizes the importance of the assumptions of linear regression and problems resulting from these assumptions. Angrist and Pischke (2009) approach regression as a tool for exploring relationships, estimating treatment effects, and providing answers to public policy questions. For a discussion of model-selection techniques and exploratory data analysis, see Mosteller and Tukey (1977). For a mathematically rigorous treatment, see Peracchi (2001, chap. 6). Finally, see Plackett (1972) if you are interested in the history of regression. Least squares, which dates back to the 1790s, was discovered independently by Legendre and Gauss.

Ordinary least squares

Example 1

Suppose that we have data on the mileage rating and weight of 74 automobiles. The variables in our data are mpg, weight, and foreign. The last variable assumes the value 1 for foreign and 0 for domestic automobiles. We wish to fit the model

$$\text{mpg} = \beta_0 + \beta_1\text{weight} + \beta_2\text{weight}^2 + \beta_3\text{foreign} + \epsilon$$

We include c.weight#c.weight in our model for the weight-squared term (see [U] 11.4.3 Factor variables):

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. regress mpg weight c.weight#c.weight foreign
```

| Source   | SS         | df | MS         |                 |        |  |
|----------|------------|----|------------|-----------------|--------|--|
| Model    | 1689.15372 | 3  | 563.05124  | Number of obs = | 74     |  |
| Residual | 754.30574  | 70 | 10.7757963 | F( 3, 70) =     | 52.25  |  |
|          |            |    |            | Prob > F =      | 0.0000 |  |
|          |            |    |            | R-squared =     | 0.6913 |  |
|          |            |    |            | Adj R-squared = | 0.6781 |  |
| Total    | 2443.45946 | 73 | 33.4720474 | Root MSE =      | 3.2827 |  |

| mpg               | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|-------------------|-----------|-----------|-------|-------|----------------------|-----------|
| weight            | -.0165729 | .0039692  | -4.18 | 0.000 | -.0244892            | -.0086567 |
| c.weight#c.weight | 1.59e-06  | 6.25e-07  | 2.55  | 0.013 | 3.45e-07             | 2.84e-06  |
| foreign           | -2.2035   | 1.059246  | -2.08 | 0.041 | -4.3161              | -.0909002 |
| _cons             | 56.53884  | 6.197383  | 9.12  | 0.000 | 44.17855             | 68.89913  |



regress produces a variety of summary statistics along with the table of regression coefficients. At the upper left, regress reports an analysis-of-variance (ANOVA) table. The column headings SS, df, and MS stand for “sum of squares”, “degrees of freedom”, and “mean square”, respectively. In the previous example, the total sum of squares is 2,443.5: 1,689.2 accounted for by the model and 754.3 left unexplained. Because the regression included a constant, the total sum reflects the sum after removal of means, as does the sum of squares due to the model. The table also reveals that there are 73 total degrees of freedom (counted as 74 observations less 1 for the mean removal), of which 3 are consumed by the model, leaving 70 for the residual.

To the right of the ANOVA table are presented other summary statistics. The  $F$  statistic associated with the ANOVA table is 52.25. The statistic has 3 numerator and 70 denominator degrees of freedom. The  $F$  statistic tests the hypothesis that all coefficients *excluding the constant* are zero. The chance of observing an  $F$  statistic that large or larger is reported as 0.0000, which is Stata's way of indicating a number smaller than 0.00005. The  $R$ -squared ( $R^2$ ) for the regression is 0.6913, and the  $R$ -squared adjusted for degrees of freedom ( $R_a^2$ ) is 0.6781. The root mean squared error, labeled Root MSE, is 3.2827. It is the square root of the mean squared error reported for the residual in the ANOVA table.

Finally, Stata produces a table of the estimated coefficients. The first line of the table indicates that the left-hand-side variable is `mpg`. Thereafter follow the four estimated coefficients. Our fitted model is

$$\text{mpg\_hat} = 56.54 - 0.0166\text{weight} + 1.59 \times 10^{-6} \text{c.weight\#c.weight} - 2.20\text{foreign}$$

Reported to the right of the coefficients in the output are the standard errors. For instance, the standard error for the coefficient on `weight` is 0.0039692. The corresponding  $t$  statistic is  $-4.18$ , which has a two-sided significance level of 0.000. This number indicates that the significance is less than 0.0005. The 95% confidence interval for the coefficient is  $[-0.024, -0.009]$ .

## ► Example 2

`regress` shares the features of all estimation commands. Among other things, this means that after running a regression, we can use `test` to test hypotheses about the coefficients, `estat vce` to examine the covariance matrix of the estimators, and `predict` to obtain predicted values, residuals, and influence statistics. See [\[U\] 20 Estimation and postestimation commands](#). Options that affect how estimates are displayed, such as `beta` or `level()`, can be used when replaying results.

Suppose that we meant to specify the `beta` option to obtain beta coefficients (regression coefficients normalized by the ratio of the standard deviation of the regressor to the standard deviation of the dependent variable). Even though we forgot, we can specify the option now:

| . regress, beta   |            |           |            |                 |           |
|-------------------|------------|-----------|------------|-----------------|-----------|
| Source            | SS         | df        | MS         |                 |           |
| Model             | 1689.15372 | 3         | 563.05124  | Number of obs = | 74        |
| Residual          | 754.30574  | 70        | 10.7757963 | F( 3, 70) =     | 52.25     |
|                   |            |           |            | Prob > F =      | 0.0000    |
|                   |            |           |            | R-squared =     | 0.6913    |
|                   |            |           |            | Adj R-squared = | 0.6781    |
| Total             | 2443.45946 | 73        | 33.4720474 | Root MSE =      | 3.2827    |
| mpg               | Coef.      | Std. Err. | t          | P> t            | Beta      |
| weight            | -.0165729  | .0039692  | -4.18      | 0.000           | -2.226321 |
| c.weight#c.weight | 1.59e-06   | 6.25e-07  | 2.55       | 0.013           | 1.32654   |
| foreign           | -2.2035    | 1.059246  | -2.08      | 0.041           | -.17527   |
| _cons             | 56.53884   | 6.197383  | 9.12       | 0.000           | .         |

Treatment of the constant

By default, `regress` includes an intercept (constant) term in the model. The `noconstant` option suppresses it, and the `hascons` option tells `regress` that the model already has one.

➤ Example 3

We wish to fit a regression of the `weight` of an automobile against its `length`, and we wish to impose the constraint that the weight is zero when the length is zero.

If we simply type `regress weight length`, we are fitting the model

$$\text{weight} = \beta_0 + \beta_1 \text{length} + \epsilon$$

Here a `length` of zero corresponds to a `weight` of  $\beta_0$ . We want to force  $\beta_0$  to be zero or, equivalently, estimate an equation that does not include an intercept:

$$\text{weight} = \beta_1 \text{length} + \epsilon$$

We do this by specifying the `noconstant` option:

```
. regress weight length, noconstant
```

| Source   | SS         | df | MS         |
|----------|------------|----|------------|
| Model    | 703869302  | 1  | 703869302  |
| Residual | 14892897.8 | 73 | 204012.299 |
| Total    | 718762200  | 74 | 9713002.7  |

Number of obs = 74

F( 1, 73) = 3450.13

Prob > F = 0.0000

R-squared = 0.9793

Adj R-squared = 0.9790

Root MSE = 451.68

| weight | Coef.    | Std. Err. | t     | P> t  | [95% Conf. Interval] |         |
|--------|----------|-----------|-------|-------|----------------------|---------|
| length | 16.29829 | .2774752  | 58.74 | 0.000 | 15.74528             | 16.8513 |

In our data, `length` is measured in inches and `weight` in pounds. We discover that each inch of `length` adds 16 pounds to the weight.



Sometimes there is no need for Stata to include a constant term in the model. Most commonly, this occurs when the model contains a set of mutually exclusive indicator variables. `hascons` is a variation of the `noconstant` option—it tells Stata not to add a constant to the regression because the regression specification already has one, either directly or indirectly.

For instance, we now refit our model of `weight` as a function of `length` and include separate constants for foreign and domestic cars by specifying `bn.foreign`. `bn.foreign` is factor-variable notation for “no base for foreign” or “include all levels of variable `foreign` in the model”; see [U] 11.4.3 Factor variables.

```
. regress weight length bn.foreign, hascons
```

| Source   | SS         | df | MS         | Number of obs = 74     |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | 39647744.7 | 2  | 19823872.3 | F( 2, 71) = 316.54     |  |  |
| Residual | 4446433.7  | 71 | 62625.8268 | Prob > F = 0.0000      |  |  |
|          |            |    |            | R-squared = 0.8992     |  |  |
|          |            |    |            | Adj R-squared = 0.8963 |  |  |
| Total    | 44094178.4 | 73 | 604029.841 | Root MSE = 250.25      |  |  |

| weight  | Coef.     | Std. Err. | t      | P> t  | [95% Conf. Interval] |           |
|---------|-----------|-----------|--------|-------|----------------------|-----------|
| length  | 31.44455  | 1.601234  | 19.64  | 0.000 | 28.25178             | 34.63732  |
| foreign |           |           |        |       |                      |           |
| 0       | -2850.25  | 315.9691  | -9.02  | 0.000 | -3480.274            | -2220.225 |
| 1       | -2983.927 | 275.1041  | -10.85 | 0.000 | -3532.469            | -2435.385 |

## □ Technical note

There is a subtle distinction between the `hascons` and `noconstant` options. We can most easily reveal it by refitting the last regression, specifying `noconstant` rather than `hascons`:

```
. regress weight length bn.foreign, noconstant
```

| Source   | SS        | df | MS         | Number of obs = 74     |  |  |
|----------|-----------|----|------------|------------------------|--|--|
| Model    | 714315766 | 3  | 238105255  | F( 3, 71) = 3802.03    |  |  |
| Residual | 4446433.7 | 71 | 62625.8268 | Prob > F = 0.0000      |  |  |
|          |           |    |            | R-squared = 0.9938     |  |  |
|          |           |    |            | Adj R-squared = 0.9936 |  |  |
| Total    | 718762200 | 74 | 9713002.7  | Root MSE = 250.25      |  |  |

| weight  | Coef.     | Std. Err. | t      | P> t  | [95% Conf. Interval] |           |
|---------|-----------|-----------|--------|-------|----------------------|-----------|
| length  | 31.44455  | 1.601234  | 19.64  | 0.000 | 28.25178             | 34.63732  |
| foreign |           |           |        |       |                      |           |
| 0       | -2850.25  | 315.9691  | -9.02  | 0.000 | -3480.274            | -2220.225 |
| 1       | -2983.927 | 275.1041  | -10.85 | 0.000 | -3532.469            | -2435.385 |

Comparing this output with that produced by the previous `regress` command, we see that they are almost, but not quite, identical. The parameter estimates and their associated statistics—the second half of the output—are identical. The overall summary statistics and the ANOVA table—the first half of the output—are different, however.

In the first case, the  $R^2$  is shown as 0.8992; here it is shown as 0.9938. In the first case, the  $F$  statistic is 316.54; now it is 3,802.03. The numerator degrees of freedom are different as well. In the first case, the numerator degrees of freedom are 2; now they are 3. Which is correct?

Both are. Specifying the `hascons` option causes `regress` to adjust the ANOVA table and its associated statistics for the explanatory power of the constant. The regression in effect has a constant; it is just written in such a way that a separate constant is unnecessary. No such adjustment is made with the `noconstant` option.

□

□ Technical note

When the `hascons` option is specified, `regress` checks to make sure that the model does in fact have a constant term. If `regress` cannot find a constant term, it automatically adds one. Fitting a model of `weight` on `length` and specifying the `hascons` option, we obtain

```
. regress weight length, hascons
(note: hascons false)
```

| Source   | SS         | df | MS         | Number of obs = 74     |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | 39461306.8 | 1  | 39461306.8 | F( 1, 72) = 613.27     |  |  |
| Residual | 4632871.55 | 72 | 64345.4382 | Prob > F = 0.0000      |  |  |
|          |            |    |            | R-squared = 0.8949     |  |  |
|          |            |    |            | Adj R-squared = 0.8935 |  |  |
| Total    | 44094178.4 | 73 | 604029.841 | Root MSE = 253.66      |  |  |

| weight | Coef.     | Std. Err. | t      | P> t  | [95% Conf. Interval] |           |
|--------|-----------|-----------|--------|-------|----------------------|-----------|
| length | 33.01988  | 1.333364  | 24.76  | 0.000 | 30.36187             | 35.67789  |
| _cons  | -3186.047 | 252.3113  | -12.63 | 0.000 | -3689.02             | -2683.073 |

Even though we specified `hascons`, `regress` included a constant, anyway. It also added a note to our output: “note: hascons false”.



□ Technical note

Even if the model specification effectively includes a constant term, we need not specify the `hascons` option. `regress` is always on the lookout for collinear variables and omits them from the model. For instance,

```
. regress weight length bn.foreign
note: 1.foreign omitted because of collinearity
```

| Source   | SS         | df | MS         | Number of obs = 74     |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | 39647744.7 | 2  | 19823872.3 | F( 2, 71) = 316.54     |  |  |
| Residual | 4446433.7  | 71 | 62625.8268 | Prob > F = 0.0000      |  |  |
|          |            |    |            | R-squared = 0.8992     |  |  |
|          |            |    |            | Adj R-squared = 0.8963 |  |  |
| Total    | 44094178.4 | 73 | 604029.841 | Root MSE = 250.25      |  |  |

| weight  | Coef.     | Std. Err. | t      | P> t  | [95% Conf. Interval] |           |
|---------|-----------|-----------|--------|-------|----------------------|-----------|
| length  | 31.44455  | 1.601234  | 19.64  | 0.000 | 28.25178             | 34.63732  |
| foreign |           |           |        |       |                      |           |
| 0       | 133.6775  | 77.47615  | 1.73   | 0.089 | -20.80555            | 288.1605  |
| 1       | (omitted) |           |        |       |                      |           |
| _cons   | -2983.927 | 275.1041  | -10.85 | 0.000 | -3532.469            | -2435.385 |





## Robust standard errors

`regress` with the `vce(robust)` option substitutes a robust variance matrix calculation for the conventional calculation, or if `vce(cluster clustvar)` is specified, allows relaxing the assumption of independence within groups. How this method works is explained in [U] 20.16 **Obtaining robust variance estimates**. Below we show how well this approach works.

### ► Example 4

Specifying the `vce(robust)` option is equivalent to requesting White-corrected standard errors in the presence of heteroskedasticity. We use the automobile data and, in the process of looking at the energy efficiency of cars, analyze a variable with considerable heteroskedasticity.

We will examine the amount of energy—measured in gallons of gasoline—that the cars in the data need to move 1,000 pounds of their weight 100 miles. We are going to examine the relative efficiency of foreign and domestic cars.

```
. gen gpmw = ((1/mpg)/weight)*100*1000
. summarize gpmw
```

| Variable | Obs | Mean     | Std. Dev. | Min     | Max     |
|----------|-----|----------|-----------|---------|---------|
| gpmw     | 74  | 1.682184 | .2426311  | 1.09553 | 2.30521 |

In these data, the engines consume between 1.10 and 2.31 gallons of gas to move 1,000 pounds of the car's weight 100 miles. If we ran a regression with conventional standard errors of `gpmw` on `foreign`, we would obtain

```
. regress gpmw foreign
```

| Source   | SS         | df        | MS         | Number of obs = 74     |                      |          |
|----------|------------|-----------|------------|------------------------|----------------------|----------|
| Model    | .936705572 | 1         | .936705572 | F( 1, 72) = 20.07      |                      |          |
| Residual | 3.36079459 | 72        | .046677703 | Prob > F = 0.0000      |                      |          |
|          |            |           |            | R-squared = 0.2180     |                      |          |
|          |            |           |            | Adj R-squared = 0.2071 |                      |          |
| Total    | 4.29750017 | 73        | .058869865 | Root MSE = .21605      |                      |          |
| gpmw     | Coef.      | Std. Err. | t          | P> t                   | [95% Conf. Interval] |          |
| foreign  | .2461526   | .0549487  | 4.48       | 0.000                  | .1366143             | .3556909 |
| _cons    | 1.609004   | .0299608  | 53.70      | 0.000                  | 1.549278             | 1.66873  |

`regress` with the `vce(robust)` option, on the other hand, reports

```
. regress gpmw foreign, vce(robust)
```

| Linear regression |          |                  |       | Number of obs = 74 |                      |          |
|-------------------|----------|------------------|-------|--------------------|----------------------|----------|
|                   |          |                  |       | F( 1, 72) = 13.13  |                      |          |
|                   |          |                  |       | Prob > F = 0.0005  |                      |          |
|                   |          |                  |       | R-squared = 0.2180 |                      |          |
|                   |          |                  |       | Root MSE = .21605  |                      |          |
| gpmw              | Coef.    | Robust Std. Err. | t     | P> t               | [95% Conf. Interval] |          |
| foreign           | .2461526 | .0679238         | 3.62  | 0.001              | .1107489             | .3815563 |
| _cons             | 1.609004 | .0234535         | 68.60 | 0.000              | 1.56225              | 1.655758 |

The point estimates are the same (foreign cars need one-quarter gallon more gas), but the standard errors differ by roughly 20%. Conventional regression reports the 95% confidence interval as [0.14, 0.36], whereas the robust standard errors make the interval [0.11, 0.38].

Which is right? Notice that `gpmw` is a variable with considerable heteroskedasticity:

```
. tabulate foreign, summarize(gpmw)
```

| Car type | Summary of gpmw |           | Freq. |
|----------|-----------------|-----------|-------|
|          | Mean            | Std. Dev. |       |
| Domestic | 1.6090039       | .16845182 | 52    |
| Foreign  | 1.8551565       | .30186861 | 22    |
| Total    | 1.6821844       | .24263113 | 74    |

Thus here we favor the robust standard errors. In [U] 20.16 Obtaining robust variance estimates, we show another example using linear regression where it makes little difference whether we specify `vce(robust)`. The linear-regression assumptions were true, and we obtained nearly linear-regression results. The advantage of the robust estimate is that in neither case did we have to check assumptions.

□ Technical note

`regress` purposefully suppresses displaying the ANOVA table when `vce(robust)` is specified, as it is no longer appropriate in a statistical sense, even though, mechanically, the numbers would be unchanged. That is, sums of squares remain unchanged, but the meaning of those sums is no longer relevant. The  $F$  statistic, for instance, is no longer based on sums of squares; it becomes a Wald test based on the robustly estimated variance matrix. Nevertheless, `regress` continues to report the  $R^2$  and the root MSE even though both numbers are based on sums of squares and are, strictly speaking, irrelevant. In this, the root MSE is more in violation of the spirit of the robust estimator than is  $R^2$ . As a goodness-of-fit statistic,  $R^2$  is still fine; just do not use it in formulas to obtain  $F$  statistics because those formulas no longer apply. The root MSE is valid in a literal sense—it is the square root of the mean squared error, but it is no longer an estimate of  $\sigma$  because there is no single  $\sigma$ ; the variance of the residual varies observation by observation.

▷ Example 5

The `vce(hc2)` and `vce(hc3)` options modify the robust variance calculation. In the context of linear regression without clustering, the idea behind the robust calculation is somehow to measure  $\sigma_j^2$ , the variance of the residual associated with the  $j$ th observation, and then to use that estimate to improve the estimated variance of  $\hat{\beta}$ . Because residuals have (theoretically and practically) mean 0, one estimate of  $\sigma_j^2$  is the observation's squared residual itself— $u_j^2$ . A finite-sample correction could improve that by multiplying  $u_j^2$  by  $n/(n - k)$ , and, as a matter of fact, `vce(robust)` uses  $\{n/(n - k)\}u_j^2$  as its estimate of the residual's variance.

`vce(hc2)` and `vce(hc3)` use alternative estimators of the observation-specific variances. For instance, if the residuals are homoskedastic, we can show that the expected value of  $u_j^2$  is  $\sigma^2(1 - h_{jj})$ , where  $h_{jj}$  is the  $j$ th diagonal element of the projection (hat) matrix.  $h_{jj}$  has average value  $k/n$ , so  $1 - h_{jj}$  has average value  $1 - k/n = (n - k)/n$ . Thus the default robust estimator  $\hat{\sigma}_j = \{n/(n - k)\}u_j^2$  amounts to dividing  $u_j^2$  by the average of the expectation.

`vce(hc2)` divides  $u_j^2$  by  $1 - h_{jj}$  itself, so it should yield better estimates if the residuals really are homoskedastic. `vce(hc3)` divides  $u_j^2$  by  $(1 - h_{jj})^2$  and has no such clean interpretation. Davidson and MacKinnon (1993) show that  $u_j^2/(1 - h_{jj})^2$  approximates a more complicated estimator that they obtain by jackknifing (MacKinnon and White 1985). Angrist and Pischke (2009) also illustrate the relative merits of these adjustments.

Here are the results of refitting our efficiency model using `vce(hc2)` and `vce(hc3)`:

```
. regress gpmw foreign, vce(hc2)
```

Linear regression

```
Number of obs = 74
F(1, 72) = 12.93
Prob > F = 0.0006
R-squared = 0.2180
Root MSE = .21605
```

| gpmw    | Robust HC2 |           | t     | P> t  | [95% Conf. Interval] |          |
|---------|------------|-----------|-------|-------|----------------------|----------|
|         | Coef.      | Std. Err. |       |       |                      |          |
| foreign | .2461526   | .0684669  | 3.60  | 0.001 | .1096662             | .3826389 |
| _cons   | 1.609004   | .0233601  | 68.88 | 0.000 | 1.562437             | 1.655571 |

```
. regress gpmw foreign, vce(hc3)
```

Linear regression

```
Number of obs = 74
F(1, 72) = 12.38
Prob > F = 0.0008
R-squared = 0.2180
Root MSE = .21605
```

| gpmw    | Robust HC3 |           | t     | P> t  | [95% Conf. Interval] |          |
|---------|------------|-----------|-------|-------|----------------------|----------|
|         | Coef.      | Std. Err. |       |       |                      |          |
| foreign | .2461526   | .069969   | 3.52  | 0.001 | .1066719             | .3856332 |
| _cons   | 1.609004   | .023588   | 68.21 | 0.000 | 1.561982             | 1.656026 |

◀

## ► Example 6

The `vce(cluster clustvar)` option relaxes the assumption of independence. Below we have 28,534 observations on 4,711 women aged 14–46 years. Data were collected on these women between 1968 and 1988. We are going to fit a classic earnings model, and we begin by ignoring that each woman appears an average of 6.056 times in the data.

(Continued on next page)

```
. use http://www.stata-press.com/data/r11/regsmpl
(NLS Women 14-26 in 1968)

. regress ln_wage age c.age#c.age tenure
```

| Source   | SS         | df    | MS         |
|----------|------------|-------|------------|
| Model    | 1054.52501 | 3     | 351.508335 |
| Residual | 5360.43962 | 28097 | .190783344 |
| Total    | 6414.96462 | 28100 | .228290556 |

Number of obs = 28101  
F( 3, 28097) = 1842.45  
Prob > F = 0.0000  
R-squared = 0.1644  
Adj R-squared = 0.1643  
Root MSE = .43679

| ln_wage     | Coef.     | Std. Err. | t      | P> t  | [95% Conf. Interval] |           |
|-------------|-----------|-----------|--------|-------|----------------------|-----------|
| age         | .0752172  | .0034736  | 21.65  | 0.000 | .0684088             | .0820257  |
| c.age#c.age | -.0010851 | .0000575  | -18.86 | 0.000 | -.0011979            | -.0009724 |
| tenure      | .0390877  | .0007743  | 50.48  | 0.000 | .0375699             | .0406054  |
| _cons       | .3339821  | .0504413  | 6.62   | 0.000 | .2351148             | .4328495  |

The number of observations in our model is 28,101 because Stata drops observations that have a missing value for one or more of the variables in the model. We can be reasonably certain that the standard errors reported above are meaningless. Without a doubt, a woman with higher-than-average wages in one year typically has higher-than-average wages in other years, and so the residuals are not independent. One way to deal with this would be to fit a random-effects model—and we are going to do that—but first we fit the model using `regress` specifying `vce(cluster id)`, which treats only observations with different person ids as truly independent:

```
. regress ln_wage age c.age#c.age tenure, vce(cluster id)
```

Linear regression

Number of obs = 28101  
F( 3, 4698) = 748.82  
Prob > F = 0.0000  
R-squared = 0.1644  
Root MSE = .43679

(Std. Err. adjusted for 4699 clusters in idcode)

| ln_wage     | Coef.     | Robust Std. Err. | t      | P> t  | [95% Conf. Interval] |           |
|-------------|-----------|------------------|--------|-------|----------------------|-----------|
| age         | .0752172  | .0045711         | 16.45  | 0.000 | .0662557             | .0841788  |
| c.age#c.age | -.0010851 | .0000778         | -13.94 | 0.000 | -.0012377            | -.0009325 |
| tenure      | .0390877  | .0014425         | 27.10  | 0.000 | .0362596             | .0419157  |
| _cons       | .3339821  | .0641918         | 5.20   | 0.000 | .208136              | .4598282  |

For comparison, we focus on the tenure coefficient, which in economics jargon can be interpreted as the rate of return for keeping your job. The 95% confidence interval we previously estimated—an interval we do not believe—is [0.038,0.041]. The robust interval is twice as wide, being [0.036,0.042].

As we said, one correct way to fit this model is by random-effects regression. Here is the random-effects result:

```
. xtreg ln_wage age c.age#c.age tenure, re
```

```
Random-effects GLS regression Number of obs = 28101
Group variable: idcode Number of groups = 4699
R-sq: within = 0.1370 Obs per group: min = 1
 between = 0.2154 avg = 6.0
 overall = 0.1608 max = 15
Random effects u_i ~ Gaussian Wald chi2(3) = 4717.05
corr(u_i, X) = 0 (assumed) Prob > chi2 = 0.0000
```

| ln_wage     | Coef.     | Std. Err.                         | z      | P> z  | [95% Conf. Interval] |          |
|-------------|-----------|-----------------------------------|--------|-------|----------------------|----------|
| age         | .0568296  | .0026958                          | 21.08  | 0.000 | .0515459             | .0621132 |
| c.age#c.age | -.0007566 | .0000447                          | -16.93 | 0.000 | -.0008441            | -.000669 |
| tenure      | .0260135  | .0007477                          | 34.79  | 0.000 | .0245481             | .0274789 |
| _cons       | .6136792  | .0394611                          | 15.55  | 0.000 | .5363368             | .6910216 |
| sigma_u     | .33542449 |                                   |        |       |                      |          |
| sigma_e     | .29674679 |                                   |        |       |                      |          |
| rho         | .56095413 | (fraction of variance due to u_i) |        |       |                      |          |

Robust regression estimated the 95% interval [0.036, 0.042], and xtreg estimates [0.025, 0.027]. Which is better? The random-effects regression estimator assumes a lot. We can check some of these assumptions by performing a Hausman test. Using `estimates`, we save the random-effects estimation results, and then we run the required fixed-effects regression to perform the test.

```
. estimates store random
```

```
. xtreg ln_wage age c.age#c.age tenure, fe
```

```
Fixed-effects (within) regression Number of obs = 28101
Group variable: idcode Number of groups = 4699
R-sq: within = 0.1375 Obs per group: min = 1
 between = 0.2066 avg = 6.0
 overall = 0.1568 max = 15
 F(3,23399) = 1243.00
corr(u_i, Xb) = 0.1380 Prob > F = 0.0000
```

| ln_wage     | Coef.     | Std. Err.                         | t      | P> t  | [95% Conf. Interval] |           |
|-------------|-----------|-----------------------------------|--------|-------|----------------------|-----------|
| age         | .0522751  | .002783                           | 18.78  | 0.000 | .0468202             | .05773    |
| c.age#c.age | -.0006717 | .0000461                          | -14.56 | 0.000 | -.0007621            | -.0005813 |
| tenure      | .021738   | .000799                           | 27.21  | 0.000 | .020172              | .023304   |
| _cons       | .687178   | .0405944                          | 16.93  | 0.000 | .6076103             | .7667456  |
| sigma_u     | .38743138 |                                   |        |       |                      |           |
| sigma_e     | .29674679 |                                   |        |       |                      |           |
| rho         | .6302569  | (fraction of variance due to u_i) |        |       |                      |           |

```
F test that all u_i=0: F(4698, 23399) = 7.98 Prob > F = 0.0000
```

```
. hausman . random
```

|             | Coefficients |           |            |                     |
|-------------|--------------|-----------|------------|---------------------|
|             | (b)          | (B)       | (b-B)      | sqrt(diag(V_b-V_B)) |
|             | .            | random    | Difference | S.E.                |
| age         | .0522751     | .0568296  | -.0045545  | .0006913            |
| c.age#c.age | -.0006717    | -.0007566 | .0000849   | .0000115            |
| tenure      | .021738      | .0260135  | -.0042756  | .0002816            |

b = consistent under Ho and Ha; obtained from xtreg  
B = inconsistent under Ha, efficient under Ho; obtained from xtreg  
Test: Ho: difference in coefficients not systematic  
chi2(3) = (b-B)'[(V\_b-V\_B)^(-1)](b-B)  
= 336.62  
Prob>chi2 = 0.0000

The Hausman test casts grave suspicions on the random-effects model we just fit, so we should be careful in interpreting those results.

Meanwhile, our robust regression results still stand, as long as we are careful about the interpretation. The correct interpretation is that, if the data collection were repeated (on women sampled the same way as in the original sample), and if we were to refit the model, 95% of the time we would expect the estimated coefficient on tenure to be in the range [0.036,0.042].

Even with robust regression, we must be careful about going beyond that statement. Here the Hausman test is probably picking up something that differs within and between person, which would cast doubt on our robust regression model in terms of interpreting [0.036,0.042] to contain the rate of return for keeping a job, economywide, for all women, without exception.

Weighted regression

regress can perform weighted and unweighted regression. We indicate the weight by specifying the [weight] qualifier. By default, regress assumes analytic weights; see the technical note below.

Example 7

We have census data recording the death rate (drate) and median age (medage) for each state. The data also record the region of the country in which each state is located and the overall population of the state:

```
. use http://www.stata-press.com/data/r11/census9
(1980 Census data by state)
. describe
Contains data from http://www.stata-press.com/data/r11/census9.dta
obs: 50 1980 Census data by state
vars: 6 6 Apr 2009 15:43
size: 1,650 (99.9% of memory free)
```

| variable name | storage type | display format | value label | variable label                |
|---------------|--------------|----------------|-------------|-------------------------------|
| state         | str14        | %-14s          |             | State                         |
| state2        | str2         | %-2s           |             | Two-letter state abbreviation |
| drate         | float        | %9.0g          |             | Death Rate                    |
| pop           | long         | %12.0gc        |             | Population                    |
| medage        | float        | %9.2f          |             | Median age                    |
| region        | byte         | %-8.0g         | cenreg      | Census region                 |

Sorted by:

We can use factor variables to include dummy variables for region. Because the variables in the regression reflect means rather than individual observations, the appropriate method of estimation is analytically weighted least squares (Davidson and MacKinnon 2004, 261–262), where the weight is total population:

```
. regress drate medage i.region [w=pop]
(analytic weights assumed)
(sum of wgt is 2.2591e+08)
```

| Source   | SS         | df | MS         | Number of obs = | 50     |
|----------|------------|----|------------|-----------------|--------|
| Model    | 4096.6093  | 4  | 1024.15232 | F( 4, 45) =     | 37.21  |
| Residual | 1238.40987 | 45 | 27.5202192 | Prob > F =      | 0.0000 |
| Total    | 5335.01916 | 49 | 108.877942 | R-squared =     | 0.7679 |
|          |            |    |            | Adj R-squared = | 0.7472 |
|          |            |    |            | Root MSE =      | 5.246  |

| drate  | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |
|--------|-----------|-----------|-------|-------|----------------------|
| medage | 4.283183  | .5393329  | 7.94  | 0.000 | 3.196911 5.369455    |
| region |           |           |       |       |                      |
| 2      | .3138738  | 2.456431  | 0.13  | 0.899 | -4.633632 5.26138    |
| 3      | -1.438452 | 2.320244  | -0.62 | 0.538 | -6.111663 3.234758   |
| 4      | -10.90629 | 2.681349  | -4.07 | 0.000 | -16.30681 -5.505777  |
| _cons  | -39.14727 | 17.23613  | -2.27 | 0.028 | -73.86262 -4.431915  |

To weight the regression by population, we added the qualifier `[w=pop]` to the end of the `regress` command. Our qualifier was vague (we did not say `[aweight=pop]`), but unless told otherwise, Stata assumes analytic weights for `regress`. Stata informed us that the sum of the weight is  $2.2591 \times 10^8$ ; there were approximately 226 million people residing in the United States according to our 1980 data. ◀

## □ Technical note

Once we fit a weighted regression, we can obtain the appropriately weighted variance–covariance matrix of the estimators using `estat vce` and perform appropriately weighted hypothesis tests using `test`.

In the weighted regression in the previous example, we see that `4.region` is statistically significant but that `2.region` and `3.region` are not. We use `test` to test the joint significance of the region variables:

```
. test 2.region 3.region 4.region
(1) 2.region = 0
(2) 3.region = 0
(3) 4.region = 0
 F(3, 45) = 9.84
 Prob > F = 0.0000
```

The results indicate that the region variables are jointly significant. □

`regress` also accepts frequency weights (`fweights`). Frequency weights are appropriate when the data do not reflect cell means, but instead represent replicated observations. Specifying `aweight`s or `fweight`s will not change the parameter estimates, but it will change the corresponding significance levels.

For instance, if we specified `[fweight=pop]` in the weighted regression example above—which would be statistically incorrect—Stata would treat the data as if the data represented 226 million independent observations on death rates and median age. The data most certainly do not represent that—they represent 50 observations on state averages.

With `aweight`s, Stata treats the number of observations on the process as the number of observations in the data. When we specify `fweights`, Stata treats the number of observations as if it were equal to the sum of the weights; see [Methods and formulas](#) below.

## □ Technical note

A popular request on the help line is to describe the effect of specifying `[aweight=exp]` with `regress` in terms of transformation of the dependent and independent variables. The mechanical answer is that typing

```
. regress y x1 x2 [aweight=n]
```

is equivalent to fitting the model

$$y_j \sqrt{n_j} = \beta_0 \sqrt{n_j} + \beta_1 x_{1j} \sqrt{n_j} + \beta_2 x_{2j} \sqrt{n_j} + u_j \sqrt{n_j}$$

This regression will reproduce the coefficients and covariance matrix produced by the `aweight`d regression. The mean squared errors (estimates of the variance of the residuals) will, however, be different. The transformed regression reports  $s_t^2$ , an estimate of  $\text{Var}(u_j \sqrt{n_j})$ . The `aweight`d regression reports  $s_a^2$ , an estimate of  $\text{Var}(u_j \sqrt{n_j} \sqrt{N / \sum_k n_k})$ , where  $N$  is the number of observations. Thus

$$s_a^2 = \frac{N}{\sum_k n_k} s_t^2 = \frac{s_t^2}{\bar{n}} \quad (1)$$

The logic for this adjustment is as follows: Consider the model

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + u$$

Assume that, were this model fit on individuals,  $\text{Var}(u) = \sigma_u^2$ , a constant. Assume that individual data are not available; what is available are averages  $(\bar{y}_j, \bar{x}_{1j}, \bar{x}_{2j})$  for  $j = 1, \dots, N$ , and each average is calculated over  $n_j$  observations. Then it is still true that

$$\bar{y}_j = \beta_0 + \beta_1 \bar{x}_{1j} + \beta_2 \bar{x}_{2j} + \bar{u}_j$$

where  $\bar{u}_j$  is the average of  $n_j$  mean 0, variance  $\sigma_u^2$  deviates and has variance  $\sigma_u^2 = \sigma_u^2/n_j$ . Thus multiplying through by  $\sqrt{n_j}$  produces

$$\bar{y}_j \sqrt{n_j} = \beta_0 \sqrt{n_j} + \beta_1 \bar{x}_{1j} \sqrt{n_j} + \beta_2 \bar{x}_{2j} \sqrt{n_j} + \bar{u}_j \sqrt{n_j}$$

and  $\text{Var}(\bar{u}_j \sqrt{n_j}) = \sigma_u^2$ . The mean squared error,  $s_t^2$ , reported by fitting this transformed regression is an estimate of  $\sigma_u^2$ . The coefficients and covariance matrix could also be obtained by `aweight`d `regress`. The only difference would be in the reported mean squared error, which from (1) is  $\sigma_u^2/\bar{n}$ . On average, each observation in the data reflects the averages calculated over  $\bar{n} = \sum_k n_k/N$  individuals, and thus this reported mean squared error is the average variance of an observation in the dataset. We can retrieve the estimate of  $\sigma_u^2$  by multiplying the reported mean squared error by  $\bar{n}$ .

More generally, `aweight`s are used to solve general heteroskedasticity problems. In these cases, we have the model

$$y_j = \beta_0 + \beta_1 x_{1j} + \beta_2 x_{2j} + u_j$$



and the variance of  $u_j$  is thought to be proportional to  $a_j$ . If the variance is proportional to  $a_j$ , it is also proportional to  $\alpha a_j$ , where  $\alpha$  is any positive constant. Not quite arbitrarily, but with no loss of generality, we could choose  $\alpha = \sum_k (1/a_k)/N$ , the average value of the inverse of  $a_j$ . We can then write  $\text{Var}(u_j) = k\alpha a_j \sigma^2$ , where  $k$  is the constant of proportionality that is no longer a function of the scale of the weights.

Dividing this regression through by the  $\sqrt{a_j}$ ,

$$y_j/\sqrt{a_j} = \beta_0/\sqrt{a_j} + \beta_1 x_{1j}/\sqrt{a_j} + \beta_2 x_{2j}/\sqrt{a_j} + u_j/\sqrt{a_j}$$

produces a model with  $\text{Var}(u_j/\sqrt{a_j}) = k\alpha\sigma^2$ , which is the constant part of  $\text{Var}(u_j)$ . This variance is a function of  $\alpha$ , the average of the reciprocal weights; if the weights are scaled arbitrarily, then so is this variance.

We can also fit this model by typing

```
. regress y x1 x2 [aweight=1/a]
```

This input will produce the same estimates of the coefficients and covariance matrix; the reported mean squared error is, from (1),  $\{N/\sum_k (1/a_k)\}k\alpha\sigma^2 = k\sigma^2$ . This variance is independent of the scale of  $a_j$ .

□

## Instrumental variables and two-stage least-squares regression

An alternate syntax for `regress` can be used to produce instrumental-variables (two-stage least squares) estimates.

```
regress depvar [varlist1 [(varlist2)] [if] [in] [weight] [, regress_options]
```

This syntax is used mainly by programmers developing estimators using the instrumental-variables estimates as intermediate results. `ivregress` is normally used to directly fit these models; see [R] `ivregress`.

With this syntax, `regress` fits a structural equation of *depvar* on *varlist*<sub>1</sub> using instrumental variables regression; (*varlist*<sub>2</sub>) indicates the list of instrumental variables. With the exception of `vce(hc2)` and `vce(hc3)`, all standard `regress` options are allowed.

(Continued on next page)

## Saved results

`regress` saves the following in `e()`:

### Scalars

|                         |                                                                    |
|-------------------------|--------------------------------------------------------------------|
| <code>e(N)</code>       | number of observations                                             |
| <code>e(mss)</code>     | model sum of squares                                               |
| <code>e(df_m)</code>    | model degrees of freedom                                           |
| <code>e(rss)</code>     | residual sum of squares                                            |
| <code>e(df_r)</code>    | residual degrees of freedom                                        |
| <code>e(r2)</code>      | <i>R</i> -squared                                                  |
| <code>e(r2_a)</code>    | adjusted <i>R</i> -squared                                         |
| <code>e(F)</code>       | <i>F</i> statistic                                                 |
| <code>e(rmse)</code>    | root mean squared error                                            |
| <code>e(ll)</code>      | log likelihood under additional assumption of i.i.d. normal errors |
| <code>e(ll_0)</code>    | log likelihood, constant-only model                                |
| <code>e(N_clust)</code> | number of clusters                                                 |
| <code>e(rank)</code>    | rank of <code>e(V)</code>                                          |

### Macros

|                            |                                                                            |
|----------------------------|----------------------------------------------------------------------------|
| <code>e(cmd)</code>        | <code>regress</code>                                                       |
| <code>e(cmdline)</code>    | command as typed                                                           |
| <code>e(depvar)</code>     | name of dependent variable                                                 |
| <code>e(model)</code>      | ols or iv                                                                  |
| <code>e(wtype)</code>      | weight type                                                                |
| <code>e(wexp)</code>       | weight expression                                                          |
| <code>e(title)</code>      | title in estimation output when <code>vce()</code> is not <code>ols</code> |
| <code>e(clustvar)</code>   | name of cluster variable                                                   |
| <code>e(vce)</code>        | <i>vcetype</i> specified in <code>vce()</code>                             |
| <code>e(vcetype)</code>    | title used to label Std. Err.                                              |
| <code>e(properties)</code> | <code>b V</code>                                                           |
| <code>e(estat_cmd)</code>  | program used to implement <code>estat</code>                               |
| <code>e(predict)</code>    | program used to implement <code>predict</code>                             |
| <code>e(marginsok)</code>  | predictions allowed by <code>margins</code>                                |
| <code>e(asbalanced)</code> | factor variables <code>fvset</code> as <code>asbalanced</code>             |
| <code>e(asobserved)</code> | factor variables <code>fvset</code> as <code>asobserved</code>             |

### Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

### Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`regress` is implemented as an ado-file.

Methods and formulas are presented under the following headings:

*Coefficient estimation and ANOVA table*  
*A general notation for the robust variance calculation*  
*Robust calculation for regress*

## Coefficient estimation and ANOVA table

Variables printed in lowercase and not boldfaced (e.g.,  $x$ ) are scalars. Variables printed in lowercase and boldfaced (e.g.,  $\mathbf{x}$ ) are column vectors. Variables printed in uppercase and boldfaced (e.g.,  $\mathbf{X}$ ) are matrices.

Let  $\mathbf{v}$  be a column vector of weights specified by the user. If no weights are specified,  $\mathbf{v} = \mathbf{1}$ . Let  $\mathbf{w}$  be a column vector of normalized weights. If no weights are specified or if the user specified `fweights` or `iweights`,  $\mathbf{w} = \mathbf{v}$ . Otherwise,  $\mathbf{w} = \{\mathbf{v}/(\mathbf{1}'\mathbf{v})\}(\mathbf{1}'\mathbf{1})$ .

The *number of observations*,  $n$ , is defined as  $\mathbf{1}'\mathbf{w}$ . For `iweights`, this is truncated to an integer. The *sum of the weights* is  $\mathbf{1}'\mathbf{v}$ . Define  $c = 1$  if there is a constant in the regression and zero otherwise. Define  $k$  as the number of right-hand-side variables (including the constant).

Let  $\mathbf{X}$  denote the matrix of observations on the right-hand-side variables,  $\mathbf{y}$  the vector of observations on the left-hand-side variable, and  $\mathbf{Z}$  the matrix of observations on the instruments. If the user specifies no instruments, then  $\mathbf{Z} = \mathbf{X}$ . In the following formulas, if the user specifies weights, then  $\mathbf{X}'\mathbf{X}$ ,  $\mathbf{X}'\mathbf{y}$ ,  $\mathbf{y}'\mathbf{y}$ ,  $\mathbf{Z}'\mathbf{Z}$ ,  $\mathbf{Z}'\mathbf{X}$ , and  $\mathbf{Z}'\mathbf{y}$  are replaced by  $\mathbf{X}'\mathbf{D}\mathbf{X}$ ,  $\mathbf{X}'\mathbf{D}\mathbf{y}$ ,  $\mathbf{y}'\mathbf{D}\mathbf{y}$ ,  $\mathbf{Z}'\mathbf{D}\mathbf{Z}$ ,  $\mathbf{Z}'\mathbf{D}\mathbf{X}$ , and  $\mathbf{Z}'\mathbf{D}\mathbf{y}$ , respectively, where  $\mathbf{D}$  is a diagonal matrix whose diagonal elements are the elements of  $\mathbf{w}$ . We suppress the  $\mathbf{D}$  below to simplify the notation.

If no instruments are specified, define  $\mathbf{A}$  as  $\mathbf{X}'\mathbf{X}$  and  $\mathbf{a}$  as  $\mathbf{X}'\mathbf{y}$ . Otherwise, define  $\mathbf{A}$  as  $\mathbf{X}'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}(\mathbf{X}'\mathbf{Z})'$  and  $\mathbf{a}$  as  $\mathbf{X}'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{y}$ .

The coefficient vector  $\mathbf{b}$  is defined as  $\mathbf{A}^{-1}\mathbf{a}$ . Although not shown in the notation, unless `hascons` is specified,  $\mathbf{A}$  and  $\mathbf{a}$  are accumulated in deviation form and the constant is calculated separately. This comment applies to all statistics listed below.

The *total sum of squares*, TSS, equals  $\mathbf{y}'\mathbf{y}$  if there is no intercept and  $\mathbf{y}'\mathbf{y} - \{(\mathbf{1}'\mathbf{y})^2/n\}$  otherwise. The *degrees of freedom* is  $n - c$ .

The *error sum of squares*, ESS, is defined as  $\mathbf{y}'\mathbf{y} - 2\mathbf{bX}'\mathbf{y} + \mathbf{b}'\mathbf{X}'\mathbf{X}\mathbf{b}$  if there are instruments and as  $\mathbf{y}'\mathbf{y} - \mathbf{b}'\mathbf{X}'\mathbf{y}$  otherwise. The *degrees of freedom* is  $n - k$ .

The *model sum of squares*, MSS, equals  $\text{TSS} - \text{ESS}$ . The *degrees of freedom* is  $k - c$ .

The *mean squared error*,  $s^2$ , is defined as  $\text{ESS}/(n - k)$ . The *root mean squared error* is  $s$ , its square root.

The  $F$  statistic with  $k - c$  and  $n - k$  degrees of freedom is defined as

$$F = \frac{\text{MSS}}{(k - c)s^2}$$

if no instruments are specified. If instruments are specified and  $c = 1$ , then  $F$  is defined as

$$F = \frac{(\mathbf{b} - \mathbf{c})'\mathbf{A}(\mathbf{b} - \mathbf{c})}{(k - 1)s^2}$$

where  $\mathbf{c}$  is a vector of  $k - 1$  zeros and  $k$ th element  $\mathbf{1}'\mathbf{y}/n$ . Otherwise,  $F$  is defined as *missing*. (Here you may use the `test` command to construct any  $F$  test that you wish.)

The *R-squared*,  $R^2$ , is defined as  $R^2 = 1 - \text{ESS}/\text{TSS}$ .

The *adjusted R-squared*,  $R_a^2$ , is  $1 - (1 - R^2)(n - c)/(n - k)$ .

If `vce(robust)` is not specified, the conventional estimate of variance is  $s^2\mathbf{A}^{-1}$ . The handling of `vce(robust)` is described below.

## A general notation for the robust variance calculation

Put aside all context of linear regression and the notation that goes with it—we will return to it. First, we are going to establish a notation for describing robust variance calculations.

The calculation formula for the robust variance calculation is

$$\hat{\mathbf{V}} = q_c \hat{\mathbf{V}} \left( \sum_{k=1}^M \mathbf{u}_k^{(G)'} \mathbf{u}_k^{(G)} \right) \hat{\mathbf{V}}$$

where

$$\mathbf{u}_k^{(G)} = \sum_{j \in G_k} w_j \mathbf{u}_j$$

$G_1, G_2, \dots, G_M$  are the clusters specified by `vce(cluster clustvar)`, and  $w_j$  are the user-specified weights, normalized if `aweight` or `pweight` are specified and equal to 1 if no weights are specified.

For `fweights` without clusters, the variance formula is

$$\hat{\mathbf{V}} = q_c \hat{\mathbf{V}} \left( \sum_{j=1}^N w_j \mathbf{u}_j' \mathbf{u}_j \right) \hat{\mathbf{V}}$$

which is the same as expanding the dataset and making the calculation on the unweighted data.

If `vce(cluster clustvar)` is not specified,  $M = N$ , and each cluster contains 1 observation. The inputs into this calculation are

- $\hat{\mathbf{V}}$ , which is typically a conventionally calculated variance matrix;
- $\mathbf{u}_j$ ,  $j = 1, \dots, N$ , a row vector of scores; and
- $q_c$ , a constant finite-sample adjustment.

Thus we can now describe how estimators apply the robust calculation formula by defining  $\hat{\mathbf{V}}$ ,  $\mathbf{u}_j$ , and  $q_c$ .

Two definitions are popular enough for  $q_c$  to deserve a name. The regression-like formula for  $q_c$  (Fuller et al. 1986) is

$$q_c = \frac{N-1}{N-k} \frac{M}{M-1}$$

where  $M$  is the number of clusters and  $N$  is the number of observations. For weights,  $N$  refers to the sum of the weights if weights are frequency weights and the number of observations in the dataset (ignoring weights) in all other cases. Also note that, weighted or not,  $M = N$  when `vce(cluster clustvar)` is not specified, and then  $q_c = N/(N-k)$ .

The asymptotic-like formula for  $q_c$  is

$$q_c = \frac{M}{M-1}$$

where  $M = N$  if `vce(cluster clustvar)` is not specified.

See [U] [20.16 Obtaining robust variance estimates](#) and [P] [\\_robust](#) for a discussion of the robust variance estimator and a development of these formulas.

## Robust calculation for regress

For `regress`,  $\widehat{\mathbf{V}} = \mathbf{A}^{-1}$ . The other terms are

No instruments, `vce(robust)`, but not `vce(hc2)` or `vce(hc3)`,

$$\mathbf{u}_j = (y_j - \mathbf{x}_j \mathbf{b}) \mathbf{x}_j$$

and  $q_c$  is given by its regression-like definition.

No instruments, `vce(hc2)`,

$$\mathbf{u}_j = \frac{1}{\sqrt{1 - h_{jj}}} (y_j - \mathbf{x}_j \mathbf{b}) \mathbf{x}_j$$

where  $q_c = 1$  and  $h_{jj} = \mathbf{x}_j (\mathbf{X}' \mathbf{X})^{-1} \mathbf{x}_j'$ .

No instruments, `vce(hc3)`,

$$\mathbf{u}_j = \frac{1}{1 - h_{jj}} (y_j - \mathbf{x}_j \mathbf{b}) \mathbf{x}_j$$

where  $q_c = 1$  and  $h_{jj} = \mathbf{x}_j (\mathbf{X}' \mathbf{X})^{-1} \mathbf{x}_j'$ .

Instrumental variables,

$$\mathbf{u}_j = (y_j - \mathbf{x}_j \mathbf{b}) \widehat{\mathbf{x}}_j$$

where  $q_c$  is given by its regression-like definition, and

$$\widehat{\mathbf{x}}_j' = \mathbf{P} \mathbf{z}_j'$$

where  $\mathbf{P} = (\mathbf{X}' \mathbf{Z})(\mathbf{Z}' \mathbf{Z})^{-1}$ .

## Acknowledgments

The robust estimate of variance was first implemented in Stata by Mead Over, World Bank; Dean Jolliffe, U.S. Department of Agriculture; and Andrew Foster, Department of Economics, Brown University ([Over, Jolliffe, and Foster 1996](#)).

*(Continued on next page)*

The history of regression is long and complicated: the books by [Stigler \(1986\)](#) and [Hald \(1998\)](#) are devoted largely to the story. Legendre published first on least squares in 1805. Gauss published later in 1809, but he had the idea earlier. Gauss, and especially Laplace, tied least squares to a normal errors assumption. The idea of the normal distribution can itself be traced back to De Moivre in 1733. Laplace discussed a variety of other estimation methods and error assumptions over his long career, while linear models long predate either innovation. Most of this work was linked to problems in astronomy and geodesy.

A second wave of ideas started when Galton used graphical and descriptive methods on data bearing on heredity to develop what he called regression. His term reflects the common phenomenon that characteristics of offspring are positively correlated with those of parents but with regression slope such that offspring “regress toward the mean”. Galton’s work was rather intuitive: contributions from Pearson, Edgeworth, Yule, and others introduced more formal machinery, developed related ideas on correlation, and extended application into the biological and social sciences. So most of the elements of regression as we know it were in place by 1900.

Pierre-Simon Laplace (1749–1827) was born in Normandy and was early recognized as a remarkable mathematician. He weathered a changing political climate well enough to rise to Minister of the Interior under Napoleon in 1799 (although only for 6 weeks) and to be made a Marquis by Louis XVIII in 1817. He made many contributions to mathematics and physics, his two main interests being theoretical astronomy and probability theory (including statistics). Laplace transforms are named for him.

Adrien-Marie Legendre (1752–1833) was born in Paris (or possibly in Toulouse) and educated in mathematics and physics. He worked in number theory, geometry, differential equations, calculus, function theory, applied mathematics, and geodesy. The Legendre polynomials are named for him. His main contribution to statistics is as one of the discoverers of least squares. He died in poverty, having refused to bow to political pressures.

Johann Carl Friedrich Gauss (1777–1855) was born in Braunschweig (Brunswick), now in Germany. He studied there and at Göttingen. His doctoral dissertation at the University of Helmstedt was a discussion of the fundamental theorem of algebra. He made many fundamental contributions to geometry, number theory, algebra, real analysis, differential equations, numerical analysis, statistics, astronomy, optics, geodesy, mechanics, and magnetism. An outstanding genius, Gauss worked mostly in isolation in Göttingen.

Francis Galton (1822–1911) was born in Birmingham, England, into a well-to-do family with many connections: he and Charles Darwin were first cousins. After an unsuccessful foray into medicine, he became independently wealthy at the death of his father. Galton traveled widely in Europe, the Middle East, and Africa, and became celebrated as an explorer and geographer. His pioneering work on weather maps helped in the identification of anticyclones, which he named. From about 1865, most of his work was centered on quantitative problems in biology, anthropology, and psychology. In a sense, Galton (re)invented regression, and he certainly named it. Galton also promoted the normal distribution, correlation approaches, and the use of median and selected quantiles as descriptive statistics. He was knighted in 1909.

## References

- Adkins, L. C., and R. C. Hill. 2008. *Using Stata for Principles of Econometrics*. 3rd ed. Hoboken, NJ: Wiley.
- Alexandersson, A. 1998. [gr32: Confidence ellipses](#). *Stata Technical Bulletin* 46: 10–13. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 54–57. College Station, TX: Stata Press.

- Angrist, J. D., and J.-S. Pischke. 2009. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton, NJ: Princeton University Press.
- Cameron, A. C., and P. K. Trivedi. 2009. *Microeconometrics Using Stata*. College Station, TX: Stata Press.
- Chatterjee, S., and A. S. Hadi. 2006. *Regression Analysis by Example*. 4th ed. New York: Wiley.
- Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.
- . 2004. *Econometric Theory and Methods*. New York: Oxford University Press.
- Dohoo, I., W. Martin, and H. Stryhn. 2003. *Veterinary Epidemiologic Research*. Charlottetown, Prince Edward Island: AVC.
- Draper, N., and H. Smith. 1998. *Applied Regression Analysis*. 3rd ed. New York: Wiley.
- Dunnington, G. W. 1955. *Gauss: Titan of Science*. New York: Hafner Publishing.
- Fuller, W. A., W. J. Kennedy Jr., D. Schnell, G. Sullivan, and H. J. Park. 1986. *PC CARP*. Software package. Ames, IA: Statistical Laboratory, Iowa State University.
- Gillham, N. W. 2001. *A Life of Sir Francis Galton: From African Exploration to the Birth of Eugenics*. New York: Oxford University Press.
- Gillispie, C. C. 1997. *Pierre-Simon Laplace, 1749–1827: A Life in Exact Science*. Princeton: Princeton University Press.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- Hald, A. 1998. *A History of Mathematical Statistics from 1750 to 1930*. New York: Wiley.
- Hamilton, L. C. 2009. *Statistics with Stata (Updated for Version 10)*. Belmont, CA: Brooks/Cole.
- Hill, R. C., W. E. Griffiths, and G. C. Lim. 2008. *Principles of Econometrics*. 3rd ed. Hoboken, NJ: Wiley.
- Kmenta, J. 1997. *Elements of Econometrics*. 2nd ed. Ann Arbor: University of Michigan Press.
- Kohler, U., and F. Kreuter. 2009. *Data Analysis Using Stata*. 2nd ed. College Station, TX: Stata Press.
- Long, J. S., and J. Freese. 2000. [sg152: Listing and interpreting transformed coefficients from certain regression models](#). *Stata Technical Bulletin* 57: 27–34. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 231–240. College Station, TX: Stata Press.
- MacKinnon, J. G., and H. White. 1985. Some heteroskedasticity-consistent covariance matrix estimators with improved finite sample properties. *Journal of Econometrics* 29: 305–325.
- Mosteller, F., and J. W. Tukey. 1977. *Data Analysis and Regression: A Second Course in Statistics*. Reading, MA: Addison–Wesley.
- Over, M., D. Jolliffe, and A. Foster. 1996. [sg46: Huber correction for two-stage least squares estimates](#). *Stata Technical Bulletin* 29: 24–25. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 140–142. College Station, TX: Stata Press.
- Peracchi, F. 2001. *Econometrics*. Chichester, UK: Wiley.
- Plackett, R. L. 1972. Studies in the history of probability and statistics: XXIX. The discovery of the method of least squares. *Biometrika* 59: 239–251.
- Rogers, W. H. 1991. [smv2: Analyzing repeated measurements—some practical alternatives](#). *Stata Technical Bulletin* 4: 10–16. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 123–131. College Station, TX: Stata Press.
- Royston, P., and G. Ambler. 1998. [sg79: Generalized additive models](#). *Stata Technical Bulletin* 42: 38–43. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 217–224. College Station, TX: Stata Press.
- Schonlau, M. 2005. [Boosted regression \(boosting\): An introductory tutorial and a Stata plugin](#). *Stata Journal* 5: 330–354.
- Stigler, S. M. 1986. *The History of Statistics: The Measurement of Uncertainty before 1900*. Cambridge, MA: Belknap Press.
- Tyler, J. H. 1997. [sg73: Table making programs](#). *Stata Technical Bulletin* 40: 18–23. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 186–192. College Station, TX: Stata Press.
- Weesie, J. 1998. [sg77: Regression analysis with multiplicative heteroscedasticity](#). *Stata Technical Bulletin* 42: 28–32. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 204–210. College Station, TX: Stata Press.
- Weisberg, S. 2005. *Applied Linear Regression*. 3rd ed. New York: Wiley.

- Wooldridge, J. M. 2002. *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA: MIT Press.
- . 2009. *Introductory Econometrics: A Modern Approach*. 4th ed. Cincinnati, OH: South-Western.
- Zimmerman, F. 1998. sg93: Switching regressions. *Stata Technical Bulletin* 45: 30–33. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 183–186. College Station, TX: Stata Press.

## Also see

- [R] **regress postestimation** — Postestimation tools for regress
- [R] **regress postestimation time series** — Postestimation tools for regress with time series
- [R] **anova** — Analysis of variance and covariance
- [TS] **dfactor** — Dynamic-factor models
- [TS] **dvech** — Diagonal vech multivariate GARCH models
- [U] **20 Estimation and postestimation commands**



Description

The following postestimation commands are of special interest after `regress`:

| command                     | description                                                      |
|-----------------------------|------------------------------------------------------------------|
| <code>dfbeta</code>         | DFBETA influence statistics                                      |
| <code>estat hettest</code>  | tests for heteroskedasticity                                     |
| <code>estat imtest</code>   | information matrix test                                          |
| <code>estat ovtest</code>   | Ramsey regression specification-error test for omitted variables |
| <code>estat szroeter</code> | Szroeter's rank test for heteroskedasticity                      |
| <code>estat vif</code>      | variance inflation factors for the independent variables         |
| <code>acprplot</code>       | augmented component-plus-residual plot                           |
| <code>avplot</code>         | added-variable plot                                              |
| <code>avplots</code>        | all added-variables plots in one image                           |
| <code>cprplot</code>        | component-plus-residual plot                                     |
| <code>lvr2plot</code>       | leverage-versus-squared-residual plot                            |
| <code>rvfplot</code>        | residual-versus-fitted plot                                      |
| <code>rvpplot</code>        | residual-versus-predictor plot                                   |

These commands are not appropriate after the `svy` prefix.

For information about these commands, see below.

The following standard postestimation commands are also available:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>hausman</code>             | Hausman’s specification test                                                                        |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>            | link test for model specification                                                                   |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

For postestimation tests specific to time series, see [\[R\] regress postestimation time series](#).

### Special-interest postestimation commands

These commands provide tools for diagnosing sensitivity to individual observations, analyzing residuals, and assessing specification.

`dfbeta` will calculate one, more than one, or all the DFBETAs after `regress`. Although `predict` will also calculate DFBETAs, `predict` can do this for only one variable at a time. `dfbeta` is a convenience tool for those who want to calculate DFBETAs for multiple variables. The names for the new variables created are chosen automatically and begin with the letters `_dfbeta_`.

`estat hettest` performs three versions of the Breusch–Pagan (1979) and Cook–Weisberg (1983) test for heteroskedasticity. All three versions of this test present evidence against the null hypothesis that  $t = \mathbf{0}$  in  $\text{Var}(e) = \sigma^2 \exp(\mathbf{z}t)$ . In the `normal` version, performed by default, the null hypothesis also includes the assumption that the regression disturbances are independent-normal draws with variance  $\sigma^2$ . The normality assumption is dropped from the null hypothesis in the `iid` and `fstat` versions, which respectively produce the score and  $F$  tests discussed in [Methods and formulas](#). If `varlist` is not specified, the fitted values are used for `z`. If `varlist` or the `rhs` option is specified, the variables specified are used for `z`.

`estat imtest` performs an information matrix test for the regression model and an orthogonal decomposition into tests for heteroskedasticity, skewness, and kurtosis due to [Cameron and Trivedi \(1990\)](#); White’s test for homoskedasticity against unrestricted forms of heteroskedasticity (1980) is available as an option. White’s test is usually similar to the first term of the Cameron–Trivedi decomposition.

`estat ovtest` performs two versions of the Ramsey (1969) regression specification-error test (RESET) for omitted variables. This test amounts to fitting  $y = \mathbf{x}\mathbf{b} + \mathbf{z}\mathbf{t} + u$  and then testing  $\mathbf{t} = \mathbf{0}$ . If the `rhs` option is not specified, powers of the fitted values are used for  $\mathbf{z}$ . If `rhs` is specified, powers of the individual elements of  $\mathbf{x}$  are used.

`estat szroeter` performs Szroeter's rank test for heteroskedasticity for each of the variables in *varlist* or for the explanatory variables of the regression if `rhs` is specified.

`estat vif` calculates the centered or uncentered variance inflation factors (VIFs) for the independent variables specified in a linear regression model.

`acprplot` graphs an augmented component-plus-residual plot (a.k.a. augmented partial residual plot) as described by Mallows (1986). This seems to work better than the component-plus-residual plot for identifying nonlinearities in the data.

`avplot` graphs an added-variable plot (a.k.a. partial-regression leverage plot, partial regression plot, or adjusted partial residual plot) after `regress`. *indepvar* may be an independent variable (a.k.a. predictor, carrier, or covariate) that is currently in the model or not.

`avplots` graphs all the added-variable plots in one image.

`cprplot` graphs a component-plus-residual plot (a.k.a. partial residual plot) after `regress`. *indepvar* must be an independent variable that is currently in the model.

`lvr2plot` graphs a leverage-versus-squared-residual plot (a.k.a. L-R plot).

`rvfplot` graphs a residual-versus-fitted plot, a graph of the residuals against the fitted values.

`rvpplot` graphs a residual-versus-predictor plot (a.k.a. independent variable plot or carrier plot), a graph of the residuals against the specified predictor.

(Continued on next page)

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic]
```

| statistic        | description                                 |
|------------------|---------------------------------------------|
| xb               | linear prediction; the default              |
| residuals        | residuals                                   |
| score            | score; equivalent to residuals              |
| rstandard        | standardized residuals                      |
| rstudent         | studentized (jackknifed) residuals          |
| cooksd           | Cook's distance                             |
| leverage   hat   | leverage (diagonal elements of hat matrix)  |
| pr(a,b)          | $\Pr(y_j \mid a < y_j < b)$                 |
| e(a,b)           | $E(y_j \mid a < y_j < b)$                   |
| ystar(a,b)       | $E(y_j^*), y_j^* = \max\{a, \min(y_j, b)\}$ |
| *dfbeta(varname) | DFBETA for varname                          |
| stdp             | standard error of the linear prediction     |
| stdf             | standard error of the forecast              |
| stdr             | standard error of the residual              |
| *covratio        | COVRATIO                                    |
| *dfits           | DFITS                                       |
| *welsch          | Welsch distance                             |

Unstarred statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample. Starred statistics are calculated only for the estimation sample, even when `if e(sample)` is not specified.

`rstandard`, `rstudent`, `cooksd`, `leverage`, `dfbeta()`, `stdf`, `stdr`, `covratio`, `dfits`, and `welsch` are not available if any `vce()` other than `vce(ols)` was specified with `regress`.

`xb`, `residuals`, `score`, and `stdp` are the only options allowed with `svy` estimation results.

where *a* and *b* may be numbers or variables; *a* missing (*a* ≥ .) means  $-\infty$ , and *b* missing (*b* ≥ .) means  $+\infty$ ; see [U] 12.2.1 Missing values.

Menu

Statistics > Postestimation > Predictions, residuals, etc.

Options for predict

- `xb`, the default, calculates the linear prediction.
- `residuals` calculates the residuals.
- `score` is equivalent to `residuals` in linear regression.
- `rstandard` calculates the standardized residuals.
- `rstudent` calculates the studentized (jackknifed) residuals.
- `cooksd` calculates the Cook's *D* influence statistic (Cook 1977).
- `leverage` or `hat` calculates the diagonal elements of the projection hat matrix.

**pr**( $a, b$ ) calculates  $\Pr(a < \mathbf{x}_j \mathbf{b} + u_j < b)$ , the probability that  $y_j | \mathbf{x}_j$  would be observed in the interval  $(a, b)$ .

$a$  and  $b$  may be specified as numbers or variable names;  $lb$  and  $ub$  are variable names;

**pr**(20, 30) calculates  $\Pr(20 < \mathbf{x}_j \mathbf{b} + u_j < 30)$ ;

**pr**( $lb, ub$ ) calculates  $\Pr(lb < \mathbf{x}_j \mathbf{b} + u_j < ub)$ ;

and **pr**(20,  $ub$ ) calculates  $\Pr(20 < \mathbf{x}_j \mathbf{b} + u_j < ub)$ .

$a$  missing ( $a \geq .$ ) means  $-\infty$ ; **pr**(., 30) calculates  $\Pr(-\infty < \mathbf{x}_j \mathbf{b} + u_j < 30)$ ;

**pr**( $lb, 30$ ) calculates  $\Pr(-\infty < \mathbf{x}_j \mathbf{b} + u_j < 30)$  in observations for which  $lb \geq .$

and calculates  $\Pr(lb < \mathbf{x}_j \mathbf{b} + u_j < 30)$  elsewhere.

$b$  missing ( $b \geq .$ ) means  $+\infty$ ; **pr**(20, .) calculates  $\Pr(+\infty > \mathbf{x}_j \mathbf{b} + u_j > 20)$ ;

**pr**(20,  $ub$ ) calculates  $\Pr(+\infty > \mathbf{x}_j \mathbf{b} + u_j > 20)$  in observations for which  $ub \geq .$

and calculates  $\Pr(20 < \mathbf{x}_j \mathbf{b} + u_j < ub)$  elsewhere.

**e**( $a, b$ ) calculates  $E(\mathbf{x}_j \mathbf{b} + u_j \mid a < \mathbf{x}_j \mathbf{b} + u_j < b)$ , the expected value of  $y_j | \mathbf{x}_j$  conditional on  $y_j | \mathbf{x}_j$  being in the interval  $(a, b)$ , meaning that  $y_j | \mathbf{x}_j$  is censored.

$a$  and  $b$  are specified as they are for **pr**(.).

**ystar**( $a, b$ ) calculates  $E(y_j^*)$ , where  $y_j^* = a$  if  $\mathbf{x}_j \mathbf{b} + u_j \leq a$ ,  $y_j^* = b$  if  $\mathbf{x}_j \mathbf{b} + u_j \geq b$ , and  $y_j^* = \mathbf{x}_j \mathbf{b} + u_j$  otherwise, meaning that  $y_j^*$  is truncated.  $a$  and  $b$  are specified as they are for **pr**(.).

**dfbeta**(*varname*) calculates the DFBETA for *varname*, the difference between the regression coefficient when the  $j$ th observation is included and excluded, said difference being scaled by the estimated standard error of the coefficient. *varname* must have been included among the regressors in the previously fitted model. The calculation is automatically restricted to the estimation subsample.

**stdp** calculates the standard error of the prediction, which can be thought of as the standard error of the predicted expected value or mean for the observation's covariate pattern. The standard error of the prediction is also referred to as the standard error of the fitted value.

**stdf** calculates the standard error of the forecast, which is the standard error of the point prediction for 1 observation. It is commonly referred to as the standard error of the future or forecast value. By construction, the standard errors produced by **stdf** are always larger than those produced by **stdp**; see [Methods and formulas](#).

**stdr** calculates the standard error of the residuals.

**covratio** calculates COVRATIO (Belsley, Kuh, and Welsch 1980), a measure of the influence of the  $j$ th observation based on considering the effect on the variance–covariance matrix of the estimates. The calculation is automatically restricted to the estimation subsample.

**dfits** calculates DFITS (Welsch and Kuh 1977) and attempts to summarize the information in the leverage versus residual-squared plot into one statistic. The calculation is automatically restricted to the estimation subsample.

**welsch** calculates Welsch distance (Welsch 1982) and is a variation on **dfits**. The calculation is automatically restricted to the estimation subsample.

## Syntax for dfbeta

```
dfbeta [indepvar [indepvar [...]]] [, stub(name)]
```

## Menu

Statistics > Linear models and related > Regression diagnostics > Residual-versus-fitted plot

## Option for dfbeta

`stub(name)` specifies the leading characters `dfbeta` uses to name the new variables to be generated. The default is `stub(_dfbeta_)`.

## Syntax for estat hettest

```
estat hettest [varlist] [, rhs [normal | iid | fstat] mtest[(spec)]
```

## Menu

Statistics > Postestimation > Reports and statistics

## Options for estat hettest

**rhs** specifies that tests for heteroskedasticity be performed for the right-hand-side (explanatory) variables of the fitted regression model. The **rhs** option may be combined with a *varlist*.

**normal**, the default, causes `estat hettest` to compute the original Breusch–Pagan/Cook–Weisberg test, which assumes that the regression disturbances are normally distributed.

**iid** causes `estat hettest` to compute the  $N * R^2$  version of the score test that drops the normality assumption.

**fstat** causes `estat hettest` to compute the  $F$  statistic version that drops the normality assumption.

**mtest**[(*spec*)] specifies that multiple testing be performed. The argument specifies how  $p$ -values are adjusted. The following specifications, *spec*, are supported:

|                   |                                            |
|-------------------|--------------------------------------------|
| <u>bonferroni</u> | Bonferroni's multiple testing adjustment   |
| <u>holm</u>       | Holm's multiple testing adjustment         |
| <u>sidak</u>      | Šidák's multiple testing adjustment        |
| <u>noadjust</u>   | no adjustment is made for multiple testing |

**mtest** may be specified without an argument. This is equivalent to specifying `mtest(noadjust)`, that is, tests for the individual variables should be performed with unadjusted  $p$ -values. By default, `estat hettest` does not perform multiple testing. `mtest`[(*spec*)] may not be specified with **iid** or **fstat**.

## Syntax for estat imtest

```
estat imtest [, preserve white]
```

## Menu

Statistics > Postestimation > Reports and statistics

## Options for estat imtest

`preserve` specifies that the data in memory be preserved, all variables and cases that are not needed in the calculations be dropped, and at the conclusion the original data be restored. This option is costly for large datasets. However, because `estat imtest` has to perform an auxiliary regression on  $k(k+1)/2$  temporary variables, where  $k$  is the number of regressors, it may not be able to perform the test otherwise.

`white` specifies that White's original heteroskedasticity test also be performed.

## Syntax for estat ovtest

```
estat ovtest [, rhs]
```

## Menu

Statistics > Postestimation > Reports and statistics

## Option for estat ovtest

`rhs` specifies that powers of the right-hand-side (explanatory) variables be used in the test rather than powers of the fitted values.

## Syntax for estat szroeter

```
estat szroeter [varlist] [, rhs mtest(spec)]
```

Either *varlist* or `rhs` must be specified.

## Menu

Statistics > Postestimation > Reports and statistics

## Options for estat szroeter

`rhs` specifies that tests for heteroskedasticity be performed for the right-hand-side (explanatory) variables of the fitted regression model. Option `rhs` may be combined with a *varlist*.

`mtest(spec)` specifies that multiple testing be performed. The argument specifies how  $p$ -values are adjusted. The following specifications *spec* are supported:

|                   |                                            |
|-------------------|--------------------------------------------|
| <u>bonferroni</u> | Bonferroni's multiple testing adjustment   |
| <u>holm</u>       | Holm's multiple testing adjustment         |
| <u>sidak</u>      | Šidák's multiple testing adjustment        |
| <u>noadjust</u>   | no adjustment is made for multiple testing |

`estat szroeter` always performs multiple testing. By default, it does not adjust the  $p$ -values.

## Syntax for estat vif

```
estat vif [, uncentered]
```

## Menu

Statistics > Postestimation > Reports and statistics

## Option for estat vif

`uncentered` requests that the computation of the uncentered variance inflation factors. This option is often used to detect the collinearity of the regressors with the constant. `estat vif, uncentered` may be used after regression models fit without the constant term.

## Syntax for acprplot

```
acprplot indepvar [, acprplot_options]
```

| <i>acprplot_options</i>                  | description                                                                      |
|------------------------------------------|----------------------------------------------------------------------------------|
| Plot                                     |                                                                                  |
| <i>marker_options</i>                    | change look of markers (color, size, etc.)                                       |
| <i>marker_label_options</i>              | add marker labels; change look or position                                       |
| Reference line                           |                                                                                  |
| <u>rlopts</u> ( <i>cline_options</i> )   | affect rendition of the reference line                                           |
| Options                                  |                                                                                  |
| <u>lowess</u>                            | add a lowess smooth of the plotted points                                        |
| <u>lsopts</u> ( <i>lowess_options</i> )  | affect rendition of the lowess smooth                                            |
| <u>mspline</u>                           | add median spline of the plotted points                                          |
| <u>msopts</u> ( <i>mspline_options</i> ) | affect rendition of the spline                                                   |
| Add plots                                |                                                                                  |
| <u>addplot</u> ( <i>plot</i> )           | add other plots to the generated graph                                           |
| Y axis, X axis, Titles, Legend, Overall  |                                                                                  |
| <i>twoway_options</i>                    | any options other than <code>by()</code> documented in [G] <i>twoway_options</i> |

## Menu

Statistics > Linear models and related > Regression diagnostics > Augmented component-plus-residual plot



## Options for acprplot

Plot

*marker\_options* affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [G] [marker\\_options](#).

*marker\_label\_options* specify if and how the markers are to be labeled; see [G] [marker\\_label\\_options](#).

Reference line

`rlopts(cline_options)` affects the rendition of the reference line. See [G] [cline\\_options](#).

Options

`lowess` adds a lowess smooth of the plotted points to assist in detecting nonlinearities.

`lsopts(lowess_options)` affects the rendition of the lowess smooth. For an explanation of these options, especially the `bwidth()` option, see [R] [lowess](#). Specifying `lsopts()` implies the `lowess` option.

`mspline` adds a median spline of the plotted points to assist in detecting nonlinearities.

`msopts(mspline_options)` affects the rendition of the spline. For an explanation of these options, especially the `bands()` option, see [G] [graph twoway mspline](#). Specifying `msopts()` implies the `mspline` option.

Add plots

`addplot(plot)` provides a way to add other plots to the generated graph. See [G] [addplot\\_option](#).

Y axis, X axis, Titles, Legend, Overall

*twoway\_options* are any of the options documented in [G] [twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

## Syntax for avplot

`avplot indepvar [ , avplot_options ]`

| <i>avplot_options</i>                     | description                                                                               |
|-------------------------------------------|-------------------------------------------------------------------------------------------|
| Plot                                      |                                                                                           |
| <i>marker_options</i>                     | change look of markers (color, size, etc.)                                                |
| <i>marker_label_options</i>               | add marker labels; change look or position                                                |
| Reference line                            |                                                                                           |
| <code>rlopts(<i>cline_options</i>)</code> | affect rendition of the reference line                                                    |
| Add plots                                 |                                                                                           |
| <code>addplot(<i>plot</i>)</code>         | add other plots to the generated graph                                                    |
| Y axis, X axis, Titles, Legend, Overall   |                                                                                           |
| <i>twoway_options</i>                     | any options other than <code>by()</code> documented in [G] <a href="#">twoway_options</a> |

Menu

Statistics > Linear models and related > Regression diagnostics > Added-variable plot

Options for avplot

Plot

*marker\_options* affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [G] [marker\\_options](#).  
*marker\_label\_options* specify if and how the markers are to be labeled; see [G] [marker\\_label\\_options](#).

Reference line

`rlopts(cline_options)` affects the rendition of the reference line. See [G] [cline\\_options](#).

Add plots

`addplot(plot)` provides a way to add other plots to the generated graph. See [G] [addplot\\_option](#).

Y axis, X axis, Titles, Legend, Overall

*twoway\_options* are any of the options documented in [G] [twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

Syntax for avplots

```
avplots [, avplots_options]
```

| <i>avplots_options</i>                           | description                                                                               |
|--------------------------------------------------|-------------------------------------------------------------------------------------------|
| Plot                                             |                                                                                           |
| <i>marker_options</i>                            | change look of markers (color, size, etc.)                                                |
| <i>marker_label_options</i>                      | add marker labels; change look or position                                                |
| <i>combine_options</i>                           | any of the options documented in [G] <a href="#">graph combine</a>                        |
| Reference line                                   |                                                                                           |
| <code><u>rlopts</u>(<i>cline_options</i>)</code> | affect rendition of the reference line                                                    |
| Y axis, X axis, Titles, Legend, Overall          |                                                                                           |
| <i>twoway_options</i>                            | any options other than <code>by()</code> documented in [G] <a href="#">twoway_options</a> |

Menu

Statistics > Linear models and related > Regression diagnostics > Added-variable plot

## Options for avplots

Plot

*marker\_options* affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [G] [marker\\_options](#).

*marker\_label\_options* specify if and how the markers are to be labeled; see [G] [marker\\_label\\_options](#).

*combine\_options* are any of the options documented in [G] [graph combine](#). These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

Reference line

*rlopts*(*cline\_options*) affects the rendition of the reference line. See [G] [cline\\_options](#).

Y axis, X axis, Titles, Legend, Overall

*twoway\_options* are any of the options documented in [G] [twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

## Syntax for cprplot

`cprplot indepvar [ , cprplot_options ]`

| <i>cprplot_options</i> | description |
|------------------------|-------------|
|------------------------|-------------|

Plot

|                             |                                            |
|-----------------------------|--------------------------------------------|
| <i>marker_options</i>       | change look of markers (color, size, etc.) |
| <i>marker_label_options</i> | add marker labels; change look or position |

Reference line

|                                        |                                        |
|----------------------------------------|----------------------------------------|
| <i>rlopts</i> ( <i>cline_options</i> ) | affect rendition of the reference line |
|----------------------------------------|----------------------------------------|

Options

|                                          |                                           |
|------------------------------------------|-------------------------------------------|
| <i>lowess</i>                            | add a lowess smooth of the plotted points |
| <i>lsopts</i> ( <i>lowess_options</i> )  | affect rendition of the lowess smooth     |
| <i>mspline</i>                           | add median spline of the plotted points   |
| <i>msopts</i> ( <i>mspline_options</i> ) | affect rendition of the spline            |

Add plots

|                                |                                        |
|--------------------------------|----------------------------------------|
| <i>addplot</i> ( <i>plot</i> ) | add other plots to the generated graph |
|--------------------------------|----------------------------------------|

Y axis, X axis, Titles, Legend, Overall

|                       |                                                                                           |
|-----------------------|-------------------------------------------------------------------------------------------|
| <i>twoway_options</i> | any options other than <code>by()</code> documented in [G] <a href="#">twoway_options</a> |
|-----------------------|-------------------------------------------------------------------------------------------|

## Menu

Statistics > Linear models and related > Regression diagnostics > Component-plus-residual plot

# Options for cprplot

Plot

*marker\_options* affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [G] [marker\\_options](#).

*marker\_label\_options* specify if and how the markers are to be labeled; see [G] [marker\\_label\\_options](#).

Reference line

*rlopts(cline\_options)* affects the rendition of the reference line. See [G] [cline\\_options](#).

Options

*lowess* adds a lowess smooth of the plotted points to assist in detecting nonlinearities.

*lsopts(lowess\_options)* affects the rendition of the lowess smooth. For an explanation of these options, especially the *bwidth()* option, see [R] [lowess](#). Specifying *lsopts()* implies the *lowess* option.

*mspline* adds a median spline of the plotted points to assist in detecting nonlinearities.

*msopts(mspline\_options)* affects the rendition of the spline. For an explanation of these options, especially the *bands()* option, see [G] [graph twoway mspline](#). Specifying *msopts()* implies the *mspline* option.

Add plots

*addplot(plot)* provides a way to add other plots to the generated graph. See [G] [addplot\\_option](#).

Y axis, X axis, Titles, Legend, Overall

*twoway\_options* are any of the options documented in [G] [twoway\\_options](#), excluding *by()*. These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

## Syntax for lvr2plot

`lvr2plot [ , lvr2plot_options ]`

| <i>lvr2plot_options</i> | description |
|-------------------------|-------------|
|-------------------------|-------------|

Plot

|                             |                                            |
|-----------------------------|--------------------------------------------|
| <i>marker_options</i>       | change look of markers (color, size, etc.) |
| <i>marker_label_options</i> | add marker labels; change look or position |

Add plots

|                      |                                        |
|----------------------|----------------------------------------|
| <i>addplot(plot)</i> | add other plots to the generated graph |
|----------------------|----------------------------------------|

Y axis, X axis, Titles, Legend, Overall

|                       |                                                                                     |
|-----------------------|-------------------------------------------------------------------------------------|
| <i>twoway_options</i> | any options other than <i>by()</i> documented in [G] <a href="#">twoway_options</a> |
|-----------------------|-------------------------------------------------------------------------------------|

# Menu

Statistics > Linear models and related > Regression diagnostics > Leverage-versus-squared-residual plot

## Options for lvr2plot

Plot

*marker\_options* affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [G] [marker\\_options](#).  
*marker\_label\_options* specify if and how the markers are to be labeled; see [G] [marker\\_label\\_options](#).

Add plots

`addplot(plot)` provides a way to add other plots to the generated graph; see [G] [addplot\\_option](#).

Y axis, X axis, Titles, Legend, Overall

*twoway\_options* are any of the options documented in [G] [twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

## Syntax for rvfplot

```
rvfplot [, rvfplot_options]
```

| <i>rvfplot_options</i>                  | description                                                                               |
|-----------------------------------------|-------------------------------------------------------------------------------------------|
| Plot                                    |                                                                                           |
| <i>marker_options</i>                   | change look of markers (color, size, etc.)                                                |
| <i>marker_label_options</i>             | add marker labels; change look or position                                                |
| Add plots                               |                                                                                           |
| <code>addplot(plot)</code>              | add plots to the generated graph                                                          |
| Y axis, X axis, Titles, Legend, Overall |                                                                                           |
| <i>twoway_options</i>                   | any options other than <code>by()</code> documented in [G] <a href="#">twoway_options</a> |

# Menu

Statistics > Linear models and related > Regression diagnostics > Residual-versus-fitted plot

## Options for rvfplot

Plot

*marker\_options* affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [G] [marker\\_options](#).  
*marker\_label\_options* specify if and how the markers are to be labeled; see [G] [marker\\_label\\_options](#).

Add plots

`addplot(plot)` provides a way to add plots to the generated graph. See [G] [addplot\\_option](#).

Y axis, X axis, Titles, Legend, Overall

*twoway\_options* are any of the options documented in [G] [twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

Syntax for `rvpplot`

```
rvpplot indepvar [, rvpplot_options]
```

| <i>rvpplot_options</i>                  | description                                                                               |
|-----------------------------------------|-------------------------------------------------------------------------------------------|
| Plot                                    |                                                                                           |
| <i>marker_options</i>                   | change look of markers (color, size, etc.)                                                |
| <i>marker_label_options</i>             | add marker labels; change look or position                                                |
| Add plots                               |                                                                                           |
| <code>addplot(plot)</code>              | add other plots to the generated graph                                                    |
| Y axis, X axis, Titles, Legend, Overall |                                                                                           |
| <i>twoway_options</i>                   | any options other than <code>by()</code> documented in [G] <a href="#">twoway_options</a> |

Menu

Statistics > Linear models and related > Regression diagnostics > Residual-versus-predictor plot

Options for `rvpplot`

Plot

*marker\_options* affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [G] [marker\\_options](#).  
*marker\_label\_options* specify if and how the markers are to be labeled; see [G] [marker\\_label\\_options](#).

Add plots

`addplot(plot)` provides a way to add other plots to the generated graph; see [G] [addplot\\_option](#).

Y axis, X axis, Titles, Legend, Overall

*twoway\_options* are any of the options documented in [G] [twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

## Remarks

Remarks are presented under the following headings:

*Fitted values and residuals*  
*Prediction standard errors*  
*Prediction with weighted data*  
*Residual-versus-fitted plots*  
*Added-variable plots*  
*Component-plus-residual plots*  
*Residual-versus-predictor plots*  
*Leverage statistics*  
*L-R plots*  
*Standardized and studentized residuals*  
*DFITS, Cook's Distance, and Welsch Distance*  
*COVRATIO*  
*DFBETAs*  
*Formal tests for violations of assumptions*  
*Variance inflation factors*

Many of these commands concern identifying influential data in linear regression. This is, unfortunately, a field that is dominated by jargon, codified and partially begun by [Belsley, Kuh, and Welsch \(1980\)](#). In the words of [Chatterjee and Hadi \(1986, 416\)](#), “Belsley, Kuh, and Welsch’s book, *Regression Diagnostics*, was a very valuable contribution to the statistical literature, but it unleashed on an unsuspecting statistical community a computer speak (à la Orwell), the likes of which we have never seen.” Things have only gotten worse since then. Chatterjee and Hadi’s (1986, 1988) own attempts to clean up the jargon did not improve matters (see [Hoaglin and Kempthorne \[1986\]](#), [Velleman \[1986\]](#), and [Welsch \[1986\]](#)). We apologize for the jargon, and for our contribution to the jargon in the form of inelegant command names, we apologize most of all.

Model sensitivity refers to how estimates are affected by subsets of our data. Imagine data on  $y$  and  $x$ , and assume that the data are to be fit by the regression  $y_i = \alpha + \beta x_i + \epsilon_i$ . The regression estimates of  $\alpha$  and  $\beta$  are  $a$  and  $b$ , respectively. Now imagine that the estimated  $a$  and  $b$  would be different if a small portion of the dataset, perhaps even one observation, were deleted. As a data analyst, you would like to think that you are summarizing tendencies that apply to all the data, but you have just been told that the model you fit is unduly influenced by one point or just a few points and that, as a matter of fact, there is another model that applies to the rest of the data—a model that you have ignored. The search for subsets of the data that, if deleted, would change the results markedly is a predominant theme of this entry.

There are three key issues in identifying model sensitivity to individual observations, which go by the names *residuals*, *leverage*, and *influence*. In our  $y_i = a + bx_i + e_i$  regression, the residuals are, of course,  $e_i$ —they reveal how much our fitted value  $\hat{y}_i = a + bx_i$  differs from the observed  $y_i$ . A point  $(x_i, y_i)$  with a corresponding large residual is called an outlier. Say that you are interested in outliers because you somehow think that such points will exert undue influence on your estimates. Your feelings are generally right, but there are exceptions. A point might have a huge residual and yet not affect the estimated  $b$  at all. Nevertheless, studying observations with large residuals almost always pays off.

$(x_i, y_i)$  can be an outlier in another way—just as  $y_i$  can be far from  $\hat{y}_i$ ,  $x_i$  can be far from the center of mass of the other  $x$ ’s. Such an “outlier” should interest you just as much as the more traditional outliers. Picture a scatterplot of  $y$  against  $x$  with thousands of points in some sort of mass at the lower left of the graph and one point at the upper right of the graph. Now run a regression line through the points—the regression line will come close to the point at the upper right of the graph and may in fact, go through it. That is, this isolated point will not appear as an outlier as measured by residuals because its residual will be small. Yet this point might have a dramatic effect on our resulting estimates in the sense that, were you to delete the point, the estimates would change

markedly. Such a point is said to have high leverage. Just as with traditional outliers, a high leverage point does not necessarily have an undue effect on regression estimates, but if it does not, it is more the exception than the rule.

Now all this is a most unsatisfactory state of affairs. Points with large residuals may, but need not, have a large effect on our results, and points with small residuals may still have a large effect. Points with high leverage may, but need not, have a large effect on our results, and points with low leverage may still have a large effect. Can you not identify the influential points and simply have the computer list them for you? You can, but you will have to define what you mean by “influential”.

“Influential” is defined with respect to some statistic. For instance, you might ask which points in your data have a large effect on your estimated  $a$ , which points have a large effect on your estimated  $b$ , which points have a large effect on your estimated standard error of  $b$ , and so on, but do not be surprised when the answers to these questions are different. In any case, obtaining such measures is not difficult—all you have to do is fit the regression excluding each observation one at a time and record the statistic of interest which, in the day of the modern computer, is not too onerous. Moreover, you can save considerable computer time by doing algebra ahead of time and working out formulas that will calculate the same answers as if you ran each of the regressions. (Ignore the question of pairs of observations that, together, exert undue influence, and triples, and so on, which remains largely unsolved and for which the brute force fit-every-possible-regression procedure is not a viable alternative.)

Fitted values and residuals

Typing `predict newvar` with no options creates *newvar* containing the fitted values. Typing `predict newvar, resid` creates *newvar* containing the residuals.

Example 1

Continuing with [example 1](#) from [\[R\] regress](#), we wish to fit the following model:

$$mpg = \beta_0 + \beta_1 weight + \beta_2 weight^2 + \beta_3 foreign + \epsilon$$

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. regress mpg weight c.weight#c.weight foreign
```

|          |            |    |            |                        |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Source   | SS         | df | MS         | Number of obs = 74     |  |  |
| Model    | 1689.15372 | 3  | 563.05124  | F( 3, 70) = 52.25      |  |  |
| Residual | 754.30574  | 70 | 10.7757963 | Prob > F = 0.0000      |  |  |
| Total    | 2443.45946 | 73 | 33.4720474 | R-squared = 0.6913     |  |  |
|          |            |    |            | Adj R-squared = 0.6781 |  |  |
|          |            |    |            | Root MSE = 3.2827      |  |  |

|                   |           |           |       |       |                      |           |
|-------------------|-----------|-----------|-------|-------|----------------------|-----------|
| mpg               | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
| weight            | -.0165729 | .0039692  | -4.18 | 0.000 | -.0244892            | -.0086567 |
| c.weight#c.weight | 1.59e-06  | 6.25e-07  | 2.55  | 0.013 | 3.45e-07             | 2.84e-06  |
| foreign           | -2.2035   | 1.059246  | -2.08 | 0.041 | -4.3161              | -.0909002 |
| _cons             | 56.53884  | 6.197383  | 9.12  | 0.000 | 44.17855             | 68.89913  |



That done, we can now obtain the predicted values from the regression. We will store them in a new variable called `pmpg` by typing `predict pmpg`. Because `predict` produces no output, we will follow that by summarizing our predicted and observed values.

```
. predict pmpg
(option xb assumed; fitted values)
. summarize pmpg mpg
```

| Variable | Obs | Mean    | Std. Dev. | Min      | Max      |
|----------|-----|---------|-----------|----------|----------|
| pmpg     | 74  | 21.2973 | 4.810311  | 13.59953 | 31.86288 |
| mpg      | 74  | 21.2973 | 5.785503  | 12       | 41       |

◀

## ► Example 2: Out-of-sample predictions

We can just as easily obtain predicted values from the model by using a wholly different dataset from the one on which the model was fit. The only requirement is that the data have the necessary variables, which here are `weight` and `foreign`.

Using the data on two new cars (the Pontiac Sunbird and the Volvo 260) from the `newautos.dta` dataset, we can obtain out-of-sample predictions (or forecasts) by typing

```
. use http://www.stata-press.com/data/r11/newautos, clear
(New Automobile Models)
. predict pmpg
(option xb assumed; fitted values)
. list, divider
```

|    | make          | weight | foreign  | pmpg     |
|----|---------------|--------|----------|----------|
| 1. | Pont. Sunbird | 2690   | Domestic | 23.47137 |
| 2. | Volvo 260     | 3170   | Foreign  | 17.78846 |

The Pontiac Sunbird has a predicted mileage rating of 23.5 mpg, whereas the Volvo 260 has a predicted rating of 17.8 mpg. In comparison, the actual mileage ratings are 24 for the Pontiac and 17 for the Volvo.

◀

## Prediction standard errors

`predict` can calculate the standard error of the forecast (`stdf` option), the standard error of the prediction (`stdp` option), and the standard error of the residual (`stdr` option). It is easy to confuse `stdf` and `stdp` because both are often called the prediction error. Consider the prediction  $\hat{y}_j = \mathbf{x}_j \mathbf{b}$ , where  $\mathbf{b}$  is the estimated coefficient (column) vector and  $\mathbf{x}_j$  is a (row) vector of independent variables for which you want the prediction. First,  $\hat{y}_j$  has a variance due to the variance of the estimated coefficient vector  $\mathbf{b}$ ,

$$\text{Var}(\hat{y}_j) = \text{Var}(\mathbf{x}_j \mathbf{b}) = s^2 h_j$$

where  $h_j = \mathbf{x}_j (\mathbf{X}' \mathbf{X})^{-1} \mathbf{x}_j'$  and  $s^2$  is the mean squared error of the regression. Do not panic over the algebra—just remember that  $\text{Var}(\hat{y}_j) = s^2 h_j$ , whatever  $s^2$  and  $h_j$  are. `stdp` calculates this quantity. This is the error in the prediction due to the uncertainty about  $\mathbf{b}$ .

If you are about to hand this number out as your forecast, however, there is another error. According to your model, the true value of  $y_j$  is given by

$$y_j = \mathbf{x}_j \mathbf{b} + \epsilon_j = \hat{y}_j + \epsilon_j$$

and thus the  $\text{Var}(y_j) = \text{Var}(\hat{y}_j) + \text{Var}(\epsilon_j) = s^2 h_j + s^2$ , which is the square of `stdf. stdf`, then, is the sum of the error in the prediction plus the residual error.

`stdr` has to do with an analysis-of-variance decomposition of  $s^2$ , the estimated variance of  $y$ . The standard error of the prediction is  $s^2 h_j$ , and therefore  $s^2 h_j + s^2(1 - h_j) = s^2$  decomposes  $s^2$  into the prediction and residual variances.

► Example 3: standard error of the forecast

Returning to our model of `mpg` on `weight`, `weight2`, and `foreign`, we previously predicted the mileage rating for the Pontiac Sunbird and Volvo 260 as 23.5 and 17.8 mpg, respectively. We now want to put a standard error around our forecast. Remember, the data for these two cars were in `newautos.dta`:

```
. use http://www.stata-press.com/data/r11/newautos, clear
(New Automobile Models)
. predict pmpg
(option xb assumed; fitted values)
. predict se_pmpg, stdf
. list, divider
```

|    | make          | weight | foreign  | pmpg     | se_pmpg  |
|----|---------------|--------|----------|----------|----------|
| 1. | Pont. Sunbird | 2690   | Domestic | 23.47137 | 3.341823 |
| 2. | Volvo 260     | 3170   | Foreign  | 17.78846 | 3.438714 |

Thus an approximate 95% confidence interval for the mileage rating of the Volvo 260 is  $17.8 \pm 2 \cdot 3.44 = [10.92, 24.68]$ .



Prediction with weighted data

`predict` can be used after frequency-weighted (`fweight`) estimation, just as it is used after unweighted estimation. The technical note below concerns the use of `predict` after analytically weighted (`aweight`) estimation.

□ Technical note

After analytically weighted estimation, `predict` is willing to calculate only the prediction (no options), residual (`residual` option), standard error of the prediction (`stdp` option), and diagonal elements of the projection matrix (`hat` option). Moreover, the results produced by `hat` need to be adjusted, as will be described. For analytically weighted estimation, the standard error of the forecast and residuals, the standardized and studentized residuals, and Cook's  $D$  are not statistically well-defined concepts.

To obtain the correct values of the diagonal elements of the hat matrix, you can use `predict` with the `hat` option to make a first, partially adjusted calculation, and then follow that by completing the adjustment. Assume that you are fitting a linear regression model weighting the data with the variable `w` (`[aweight=w]`). Begin by creating a new variable, `w0`:

```
. predict resid if e(sample), resid
. summarize w if resid < . & e(sample)
. gen w0=w/r(mean)
```

Some caution is necessary at this step—the `summarize w` must be performed on the same sample that was used to fit the model, which means that you must include `if e(sample)` to restrict the prediction to the estimation sample. You created the residual and then included the modifier ‘`if resid < .`’ so that if the dependent variable or any of the independent variables is missing, the corresponding observations will be excluded from the calculation of the average value of the original weight.

To correct `predict`’s `hat` calculation, multiply the result by `w0`:

```
. predict myhat, hat
. replace myhat = w0 * myhat
```

□

## Residual-versus-fitted plots

### ► Example 4: `rvfplot`

Using the automobile dataset described in [U] 1.2.2 [Example datasets](#), we will use `regress` to fit a model of price on weight, mpg, foreign, and the interaction of foreign with mpg. We specify `foreign##c.mpg` to obtain the interaction of foreign with mpg; see [U] 11.4.3 [Factor variables](#).

```
. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)
```

```
. regress price weight foreign##c.mpg
```

| Source   | SS        | df | MS         |  |  |  |
|----------|-----------|----|------------|--|--|--|
| Model    | 350319665 | 4  | 87579916.3 |  |  |  |
| Residual | 284745731 | 69 | 4126749.72 |  |  |  |
| Total    | 635065396 | 73 | 8699525.97 |  |  |  |

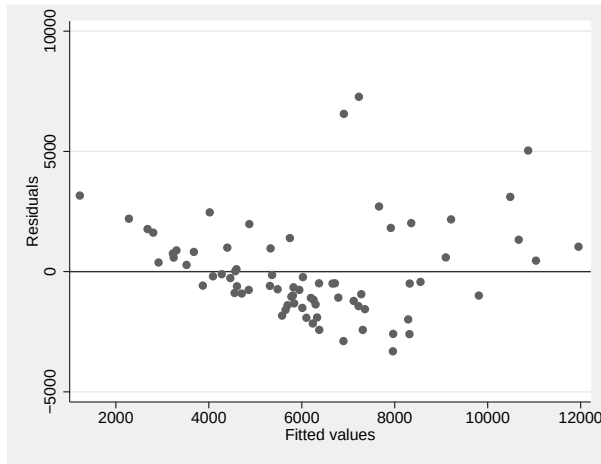
|                 |        |
|-----------------|--------|
| Number of obs = | 74     |
| F( 4, 69) =     | 21.22  |
| Prob > F =      | 0.0000 |
| R-squared =     | 0.5516 |
| Adj R-squared = | 0.5256 |
| Root MSE =      | 2031.4 |

| price             | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|-------------------|-----------|-----------|-------|-------|----------------------|-----------|
| weight            | 4.613589  | .7254961  | 6.36  | 0.000 | 3.166263             | 6.060914  |
| 1.foreign         | 11240.33  | 2751.681  | 4.08  | 0.000 | 5750.878             | 16729.78  |
| mpg               | 263.1875  | 110.7961  | 2.38  | 0.020 | 42.15527             | 484.2197  |
| foreign#<br>c.mpg |           |           |       |       |                      |           |
| 1                 | -307.2166 | 108.5307  | -2.83 | 0.006 | -523.7294            | -90.70368 |
| _cons             | -14449.58 | 4425.72   | -3.26 | 0.002 | -23278.65            | -5620.51  |

Once we have fit a model, we may use any of the regression diagnostics commands. `rvfplot` (read residual-versus-fitted plot) graphs the residuals against the fitted values:

```
. rvfplot, yline(0)
```



All the diagnostic plot commands allow the `graph twoway` and `graph twoway scatter` options; we specified a `ylines(0)` to draw a line across the graph at  $y = 0$ ; see [G] [graph twoway scatter](#).

In a well-fitted model, there should be no pattern to the residuals plotted against the fitted values—something not true of our model. Ignoring the two outliers at the top center of the graph, we see curvature in the pattern of the residuals, suggesting a violation of the assumption that price is linear in our independent variables. We might also have seen increasing or decreasing variation in the residuals—heteroskedasticity. Any pattern whatsoever indicates a violation of the least-squares assumptions.

◀

## Added-variable plots

### ► Example 5: avplot

We continue with our price model, and another diagnostic graph is provided by `avplot` (read added-variable plot, also known as the partial-regression leverage plot).

One of the wonderful features of one-regressor regressions (regressions of  $y$  on one  $x$ ) is that we can graph the data and the regression line. There is no easier way to understand the regression than to examine such a graph. Unfortunately, we cannot do this when we have more than one regressor. With two regressors, it is still theoretically possible—the graph must be drawn in three dimensions, but with three or more regressors no graph is possible.

The added-variable plot is an attempt to project multidimensional data back to the two-dimensional world for each of the original regressors. This is, of course, impossible without making some concessions. Call the coordinates on an added-variable plot  $y$  and  $x$ . The added-variable plot has the following properties:

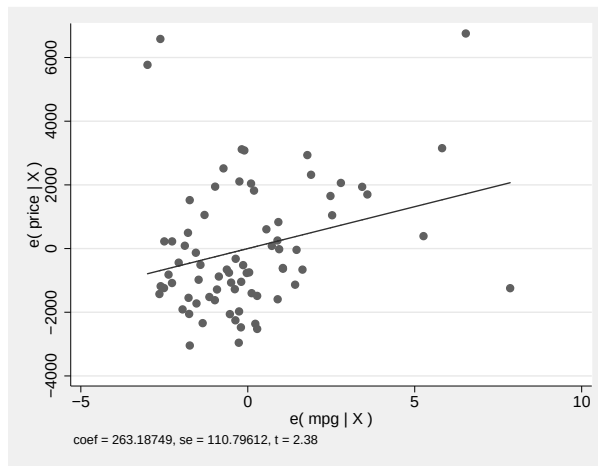
- There is a one-to-one correspondence between  $(x_i, y_i)$  and the  $i$ th observation used in the original regression.
- A regression of  $y$  on  $x$  has the same coefficient and standard error (up to a degree-of-freedom adjustment) as the estimated coefficient and standard error for the regressor in the original regression.

- The “outlierness” of each observation in determining the slope is in some sense preserved.

It is equally important to note the properties that are not listed. The  $y$  and  $x$  coordinates of the added-variable plot cannot be used to identify functional form, or, at least, not well (see [MalloWS \[1986\]](#)). In the construction of the added-variable plot, the relationship between  $y$  and  $x$  is forced to be linear.

Let’s examine the added-variable plot for mpg in our regression of price on weight and foreign##c.mpg:

```
. avplot mpg
```



This graph suggests a problem in determining the coefficient on mpg. Were this a one-regressor regression, the two points at the top-left corner and the one at the top right would cause us concern, and so it does in our more complicated multiple-regressor case. To identify the problem points, we retyped our command, modifying it to read `avplot mpg, mlabel(make)`, and discovered that the two cars at the top left are the Cadillac Eldorado and the Lincoln Versailles; the point at the top right is the Cadillac Seville. These three cars account for 100% of the luxury cars in our data, suggesting that our model is misspecified. By the way, the point at the lower right of the graph, also cause for concern, is the Plymouth Arrow, our data-entry error.

◀

## □ Technical note

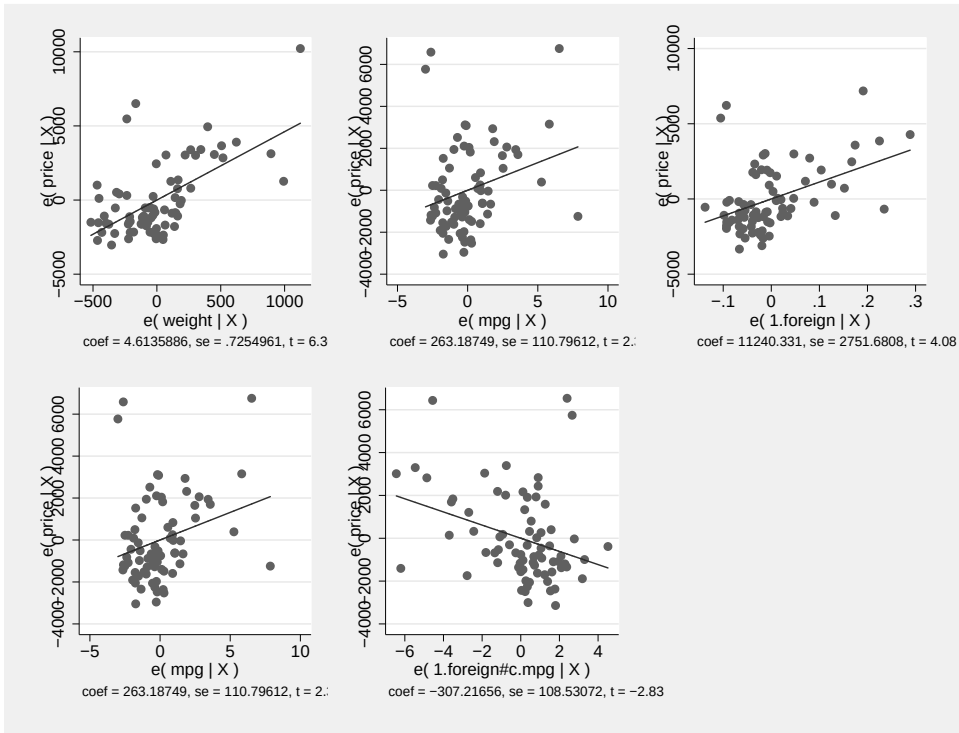
Stata’s `avplot` command can be used with regressors already in the model, as we just did, or with potential regressors not yet in the model. In either case, `avplot` will produce the correct graph. The name “added-variable plot” is unfortunate in the case when the variable is already among the list of regressors but is, we think, still preferable to the name “partial-regression leverage plot” assigned by [Belsley, Kuh, and Welsch \(1980, 30\)](#) and more in the spirit of the original use of such plots by [Mosteller and Tukey \(1977, 271–279\)](#). [Welsch \(1986, 403\)](#), however, disagrees: “I am sorry to see that [Chatterjee and Hadi \[1986\]](#) endorse the term ‘added-variable plot’ when  $X_j$  is part of the original model” and goes on to suggest the name “adjusted partial residual plot”.

□

### ► Example 6: avplots

Added-variable plots are so useful that we should look at them for every regressor in the data. `avplots` makes this easy:

```
. avplots
```



◀

### Component-plus-residual plots

Added-variable plots are successful at identifying outliers, but they cannot be used to identify functional form. The component-plus-residual plot (Ezekiel 1924; Larsen and McCleary 1972) is another attempt at projecting multidimensional data into a two-dimensional form, but with different properties. Although the added-variable plot can identify outliers, the component-plus-residual plot cannot. It can, however, be used to examine the functional-form assumptions of the model. Both plots have the property that a regression line through the coordinates has a slope equal to the estimated coefficient in the regression model.

### ► Example 7: cprplot and acprplot

To illustrate these plots, we begin with a different model:

```
. use http://www.stata-press.com/data/r11/auto1, clear
(Automobile Models)
```

```
. regress price mpg weight
```

| Source   | SS        | df | MS         | Number of obs = | 74     |
|----------|-----------|----|------------|-----------------|--------|
| Model    | 187716578 | 2  | 93858289   | F( 2, 71) =     | 14.90  |
| Residual | 447348818 | 71 | 6300687.58 | Prob > F =      | 0.0000 |
|          |           |    |            | R-squared =     | 0.2956 |
|          |           |    |            | Adj R-squared = | 0.2757 |
| Total    | 635065396 | 73 | 8699525.97 | Root MSE =      | 2510.1 |

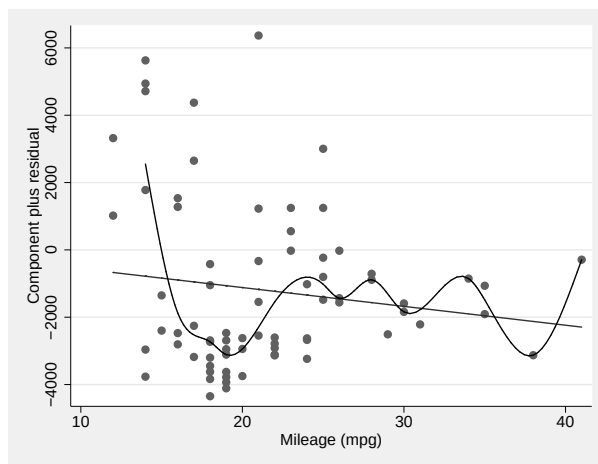
  

| price  | Coef.    | Std. Err. | t     | P> t  | [95% Conf. Interval] |
|--------|----------|-----------|-------|-------|----------------------|
| mpg    | -55.9393 | 75.24136  | -0.74 | 0.460 | -205.9663 94.08771   |
| weight | 1.710992 | .5861682  | 2.92  | 0.005 | .5422063 2.879779    |
| _cons  | 2197.9   | 3190.768  | 0.69  | 0.493 | -4164.311 8560.11    |

In fact, we know that the effects of `mpg` in this model are nonlinear—if we added `mpg squared` to the model, its coefficient would have a  $t$  statistic of 2.38, the  $t$  statistic on `mpg` would become  $-2.48$ , and `weight`'s effect would become about one-third of its current value and become statistically insignificant. Pretend that we do not know this.

The component-plus-residual plot for `mpg` is

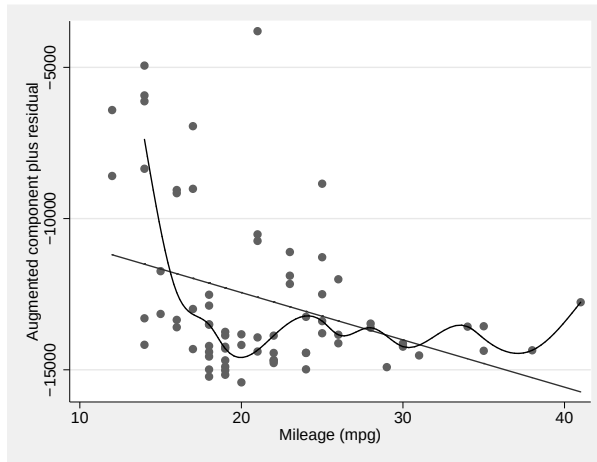
```
. cprplot mpg, mspline msopts(bands(13))
```



We are supposed to examine the above graph for nonlinearities or, equivalently, ask if the regression line, which has slope equal to the estimated effect of `mpg` in the original model, fits the data adequately. To assist our eyes, we added a median spline. Perhaps some people may detect nonlinearity from this graph, but we assert that if we had not previously revealed the nonlinearity of `mpg` and if we had not added the median spline, the graph would not overly bother us.

[MalloWS \(1986\)](#) proposed an augmented component-plus-residual plot that is often more sensitive to detecting nonlinearity:

```
. acprplot mpg, mspline msopts(bands(13))
```



It does do somewhat better.

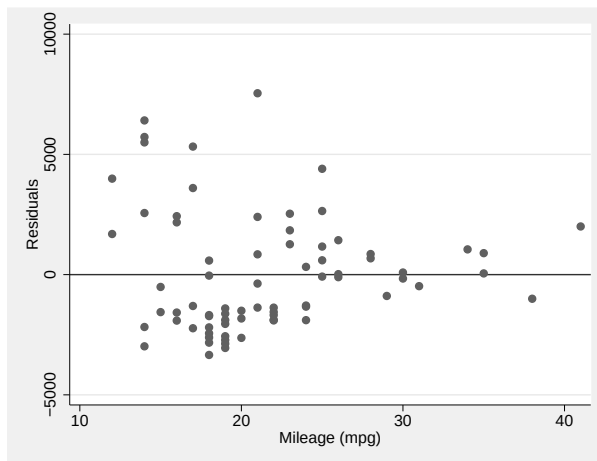
◀

## Residual-versus-predictor plots

### ► Example 8: rvpplot

The residual-versus-predictor plot is a simple way to look for violations of the regression assumptions. If the assumptions are correct, there should be no pattern in the graph. Using our `price` on `mpg` and `weight` model, we type

```
. rvpplot mpg, yline(0)
```



Remember, any pattern counts as a problem, and in this graph, we see that the variation in the residuals decreases as `mpg` increases.

◀



## Leverage statistics

In addition to providing fitted values and the associated standard errors, the `predict` command can also be used to generate various statistics used to detect the influence of individual observations. This section provides a brief introduction to leverage (`hat`) statistics, and some of the following subsections discuss other influence statistics produced by `predict`.

### ► Example 9: diagonal elements of projection matrix

The diagonal elements of the projection matrix, obtained by the `hat` option, are a measure of distance in explanatory variable space. `leverage` is a synonym for `hat`.

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. regress mpg weight c.weight#c.weight foreign
(output omitted)

. predict xdist, hat

. summarize xdist, detail
```

| Leverage |             |          |                    |
|----------|-------------|----------|--------------------|
|          | Percentiles | Smallest |                    |
| 1%       | .0251334    | .0251334 |                    |
| 5%       | .0255623    | .0251334 |                    |
| 10%      | .0259213    | .0253883 | Obs 74             |
| 25%      | .0278442    | .0255623 | Sum of Wgt. 74     |
| 50%      | .04103      |          | Mean .0540541      |
|          |             | Largest  | Std. Dev. .0459218 |
| 75%      | .0631279    | .1593606 |                    |
| 90%      | .0854584    | .1593606 | Variance .0021088  |
| 95%      | .1593606    | .2326124 | Skewness 3.440809  |
| 99%      | .3075759    | .3075759 | Kurtosis 16.95135  |

Some 5% of our sample has an `xdist` measure in excess of 0.15. Let's force them to reveal their identities:

```
. list foreign make mpg if xdist>.15, divider
```

|     | foreign  | make              | mpg |
|-----|----------|-------------------|-----|
| 24. | Domestic | Ford Fiesta       | 28  |
| 26. | Domestic | Linc. Continental | 12  |
| 27. | Domestic | Linc. Mark V      | 12  |
| 43. | Domestic | Plym. Champ       | 34  |

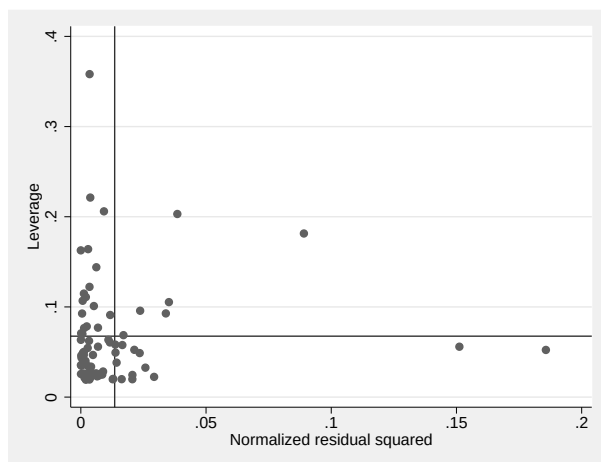
To understand why these cars are on this list, we must remember that the explanatory variables in our model are `weight` and `foreign` and that `xdist` measures distance in this metric. The Ford Fiesta and the Plymouth Champ are the two lightest domestic cars in our data. The Lincolns are the two heaviest domestic cars.

## L-R plots

### ► Example 10: lvr2plot

One of the most useful diagnostic graphs is provided by `lvr2plot` (leverage-versus-residual-squared plot), a graph of leverage against the (normalized) residuals squared.

```
. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)
. regress price weight foreign##c.mpg
(output omitted)
. lvr2plot
```

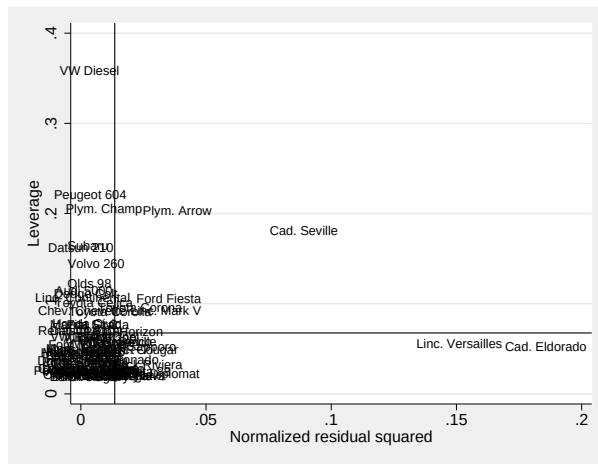


The lines on the chart show the average values of leverage and the (normalized) residuals squared. Points above the horizontal line have higher-than-average leverage; points to the right of the vertical line have larger-than-average residuals.

One point immediately catches our eye, and four more make us pause. The point at the top of the graph has high leverage and a smaller-than-average residual. The other points that bother us all have higher-than-average leverage, two with smaller-than-average residuals and two with larger-than-average residuals.

A less pretty but more useful version of the above graph specifies that `make` be used as the symbol (see [G] [marker\\_label\\_options](#)):

```
. lvr2plot, mlabel(make) mlabp(0) m(none) mlabsz(small)
```



The VW Diesel, Plymouth Champ, Plymouth Arrow, and Peugeot 604 are the points that cause us the most concern. When we further examine our data, we discover that the VW Diesel is the only diesel in our data and that the data for the Plymouth Arrow were entered incorrectly into the computer. No such simple explanations were found for the Plymouth Champ and Peugeot 604.

◀

## Standardized and studentized residuals

The terms standardized and studentized residuals have meant different things to different authors. In Stata, `predict` defines the standardized residual as  $\hat{e}_i = e_i / (s\sqrt{1 - h_i})$  and the studentized residual as  $r_i = e_i / (s_{(i)}\sqrt{1 - h_i})$ , where  $s_{(i)}$  is the root mean squared error of a regression with the  $i$ th observation removed. Stata's definition of the studentized residual is the same as the one given in [Bollen and Jackman \(1990, 264\)](#) and is what [Chatterjee and Hadi \(1988, 74\)](#) call the “externally studentized” residual. Stata’s “standardized” residual is the same as what [Chatterjee and Hadi \(1988, 74\)](#) call the “internally studentized” residual.

Standardized and studentized residuals are attempts to adjust residuals for their standard errors. Although the  $\epsilon_i$  theoretical residuals are homoskedastic by assumption (i.e., they all have the same variance), the calculated  $e_i$  are not. In fact,

$$\text{Var}(e_i) = \sigma^2(1 - h_i)$$

where  $h_i$  are the leverage measures obtained from the diagonal elements of hat matrix. Thus observations with the greatest leverage have corresponding residuals with the smallest variance.

Standardized residuals use the root mean squared error of the regression for  $\sigma$ . Studentized residuals use the root mean squared error of a regression omitting the observation in question for  $\sigma$ . In general, studentized residuals are preferable to standardized residuals for purposes of outlier identification. Studentized residuals can be interpreted as the  $t$  statistic for testing the significance of a dummy variable equal to 1 in the observation in question and 0 elsewhere ([Belsley, Kuh, and Welsch 1980](#)). Such a dummy variable would effectively absorb the observation and so remove its influence in determining the other coefficients in the model. Caution must be exercised here, however, because of the simultaneous testing problem. You cannot simply list the residuals that would be individually significant at the 5% level—their joint significance would be far less (their joint significance level would be far greater).

### ► Example 11: standardized and studentized residuals

In the opening remarks for this entry, we distinguished residuals from leverage and speculated on the impact of an observation with a small residual but large leverage. If we had adjusted the residuals for their standard errors, however, the adjusted residual would have been (relatively) larger and perhaps large enough so that we could simply examine the adjusted residuals. Taking our price on weight and foreign##c.mpg model, we can obtain the in-sample standardized and studentized residuals by typing

```
. predict esta if e(sample), rstandard
. predict estu if e(sample), rstudent
```

In *L-R plots*, we discovered that the VW Diesel has the highest leverage in our data, but a corresponding small residual. The standardized and studentized residuals for the VW Diesel are

```
. list make price esta estu if make=="VW Diesel"
```

|     | make      | price | esta     | estu     |
|-----|-----------|-------|----------|----------|
| 74. | VW Diesel | 5,397 | .6142691 | .6114758 |

The studentized residual of 0.611 can be interpreted as the  $t$  statistic for including a dummy variable for VW Diesel in our regression. Such a variable would not be significant.

◀

## DFITS, Cook's Distance, and Welsch Distance

DFITS ([Welsch and Kuh 1977](#)), Cook's Distance ([Cook 1977](#)), and Welsch Distance ([Welsch 1982](#)) are three attempts to summarize the information in the leverage versus residual-squared plot into one statistic. That is, the goal is to create an index that is affected by the size of the residuals—outliers—and the size of  $h_i$ —leverage. Viewed mechanically, one way to write DFITS ([Bollen and Jackman 1990, 265](#)) is

$$\text{DFITS}_i = r_i \sqrt{\frac{h_i}{1 - h_i}}$$

where  $r_i$  are the studentized residuals. Thus large residuals increase the value of DFITS, as do large values of  $h_i$ . Viewed more traditionally, DFITS is a scaled difference between predicted values for the  $i$ th case when the regression is fit with and without the  $i$ th observation, hence the name.

The mechanical relationship between DFITS and Cook's Distance,  $D_i$  ([Bollen and Jackman 1990, 266](#)), is

$$D_i = \frac{1}{k} \frac{s_{(i)}^2}{s^2} \text{DFITS}_i^2$$

where  $k$  is the number of variables (including the constant) in the regression,  $s$  is the root mean squared error of the regression, and  $s_{(i)}$  is the root mean squared error when the  $i$ th observation is omitted. Viewed more traditionally,  $D_i$  is a scaled measure of the distance between the coefficient vectors when the  $i$ th observation is omitted.

The mechanical relationship between DFITS and Welsch's Distance,  $W_i$  ([Chatterjee and Hadi 1988, 123](#)), is

$$W_i = \text{DFITS}_i \sqrt{\frac{n-1}{1 - h_i}}$$

The interpretation of  $W_i$  is more difficult, as it is based on the empirical influence curve. Although DFITS and Cook's distance are similar, the Welsch distance measure includes another normalization by leverage.

Belsley, Kuh, and Welsch (1980, 28) suggest that DFITS values greater than  $2\sqrt{k/n}$  deserve more investigation, and so values of Cook's distance greater than  $4/n$  should also be examined (Bollen and Jackman 1990, 265–266). Through similar logic, the cutoff for Welsch distance is approximately  $3\sqrt{k}$  (Chatterjee and Hadi 1988, 124).

## ► Example 12: DFITS influence measure

Using our model of price on weight and foreign##c.mpg, we can obtain the DFITS influence measure:

```
. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)
. regress price weight foreign##c.mpg
(output omitted)
. predict e if e(sample), resid
. predict dfits, dfits
```

We did not specify if e(sample) in computing the DFITS statistic. DFITS is available only over the estimation sample, so specifying if e(sample) would have been redundant. It would have done no harm, but it would not have changed the results.

Our model has  $k = 5$  independent variables ( $k$  includes the constant) and  $n = 74$  observations; following the  $2\sqrt{k/n}$  cutoff advice, we type

```
. list make price e dfits if abs(dfits) > 2*sqrt(5/74), divider
```

|     | make             | price  | e         | dfits     |
|-----|------------------|--------|-----------|-----------|
| 12. | Cad. Eldorado    | 14,500 | 7271.96   | .9564455  |
| 13. | Cad. Seville     | 15,906 | 5036.348  | 1.356619  |
| 24. | Ford Fiesta      | 4,389  | 3164.872  | .5724172  |
| 27. | Linc. Mark V     | 13,594 | 3109.193  | .5200413  |
| 28. | Linc. Versailles | 13,466 | 6560.912  | .8760136  |
| 42. | Plym. Arrow      | 4,647  | -3312.968 | -.9384231 |

We calculate Cook's distance and list the observations greater than the suggested  $4/n$  cutoff:

(Continued on next page)

```
. predict cooksd if e(sample), cooksd
. list make price e cooksd if cooksd > 4/74, divider
```

|     | make             | price  | e         | cooksd   |
|-----|------------------|--------|-----------|----------|
| 12. | Cad. Eldorado    | 14,500 | 7271.96   | .1492676 |
| 13. | Cad. Seville     | 15,906 | 5036.348  | .3328515 |
| 24. | Ford Fiesta      | 4,389  | 3164.872  | .0638815 |
| 28. | Linc. Versailles | 13,466 | 6560.912  | .1308004 |
| 42. | Plym. Arrow      | 4,647  | -3312.968 | .1700736 |

Here we used `if e(sample)` because Cook’s distance is not restricted to the estimation sample by default. It is worth comparing this list with the preceding one.

Finally, we use Welsch distance and the suggested  $3\sqrt{k}$  cutoff:

```
. predict wd, welsch
. list make price e wd if abs(wd) > 3*sqrt(5), divider
```

|     | make             | price  | e         | wd        |
|-----|------------------|--------|-----------|-----------|
| 12. | Cad. Eldorado    | 14,500 | 7271.96   | 8.394372  |
| 13. | Cad. Seville     | 15,906 | 5036.348  | 12.81125  |
| 28. | Linc. Versailles | 13,466 | 6560.912  | 7.703005  |
| 42. | Plym. Arrow      | 4,647  | -3312.968 | -8.981481 |

Here we did not need to specify `if e(sample)` because `welsch` automatically restricts the prediction to the estimation sample.



COVRATIO

COVRATIO (Belsley, Kuh, and Welsch 1980) measures the influence of the  $i$ th observation by considering the effect on the variance–covariance matrix of the estimates. The measure is the ratio of the determinants of the covariances matrix, with and without the  $i$ th observation. The resulting formula is

$$\text{COVRATIO}_i = \frac{1}{1 - h_i} \left( \frac{n - k - \hat{e}_i^2}{n - k - 1} \right)^k$$

where  $\hat{e}_i$  is the standardized residual.

For noninfluential observations, the value of COVRATIO is approximately 1. Large values of the residuals or large values of leverage will cause deviations from 1, although if both are large, COVRATIO may tend back toward 1 and therefore not identify such observations (Chatterjee and Hadi 1988, 139).

Belsley, Kuh, and Welsch (1980) suggest that observations for which

$$|\text{COVRATIO}_i - 1| \geq \frac{3k}{n}$$

are worthy of further examination.

### ► Example 13: COVRATIO influence measure

Using our model of price on weight and foreign##c.mpg, we can obtain the COVRATIO measure and list the observations outside the suggested cutoff by typing

```
. predict covr, covratio
. list make price e covr if abs(covr-1) >= 3*5/74, divider
```

|     | make             | price  | e         | covr     |
|-----|------------------|--------|-----------|----------|
| 12. | Cad. Eldorado    | 14,500 | 7271.96   | .3814242 |
| 13. | Cad. Seville     | 15,906 | 5036.348  | .7386969 |
| 28. | Linc. Versailles | 13,466 | 6560.912  | .4761695 |
| 43. | Plym. Champ      | 4,425  | 1621.747  | 1.27782  |
| 53. | Audi 5000        | 9,690  | 591.2883  | 1.206842 |
| 57. | Datsun 210       | 4,589  | 19.81829  | 1.284801 |
| 64. | Peugeot 604      | 12,990 | 1037.184  | 1.348219 |
| 66. | Subaru           | 3,798  | -909.5894 | 1.264677 |
| 71. | VW Diesel        | 5,397  | 999.7209  | 1.630653 |
| 74. | Volvo 260        | 11,995 | 1327.668  | 1.211888 |

The covratio option automatically restricts the prediction to the estimation sample.

◀

## DFBETAS

DFBETAS are perhaps the most direct influence measure of interest to model builders. DFBETAS focus on one coefficient and measure the difference between the regression coefficient when the  $i$ th observation is included and excluded, the difference being scaled by the estimated standard error of the coefficient. [Belsley, Kuh, and Welsch \(1980, 28\)](#) suggest observations with  $|DFBETA_i| > 2/\sqrt{n}$  as deserving special attention, but it is also common practice to use 1 ([Bollen and Jackman 1990, 267](#)), meaning that the observation shifted the estimate at least one standard error.

### ► Example 14: DFBETAS influence measure; the dfbeta() option

Using our model of price on weight and foreign##c.mpg, let's first ask which observations have the greatest impact on the determination of the coefficient on 1.foreign. We will use the suggested  $2/\sqrt{n}$  cutoff:

```
. sort foreign make
. predict dfor, dfbeta(1.foreign)
. list make price foreign dfor if abs(dfor) > 2/sqrt(74), divider
```

|     | make             | price  | foreign  | dfor      |
|-----|------------------|--------|----------|-----------|
| 12. | Cad. Eldorado    | 14,500 | Domestic | -.5290519 |
| 13. | Cad. Seville     | 15,906 | Domestic | .8243419  |
| 28. | Linc. Versailles | 13,466 | Domestic | -.5283729 |
| 42. | Plym. Arrow      | 4,647  | Domestic | -.6622424 |
| 43. | Plym. Champ      | 4,425  | Domestic | .2371104  |
| 64. | Peugeot 604      | 12,990 | Foreign  | .2552032  |
| 69. | Toyota Corona    | 5,719  | Foreign  | -.256431  |

The Cadillac Seville shifted the coefficient on 1.foreign 0.82 standard deviations!

Now let us ask which observations have the greatest effect on the mpg coefficient:

```
. predict dmpg, dfbeta(mpg)
. list make price mpg dmpg if abs(dmpg) > 2/sqrt(74), divider
```

|     | make             | price  | mpg | dmpg      |
|-----|------------------|--------|-----|-----------|
| 12. | Cad. Eldorado    | 14,500 | 14  | -.5970351 |
| 13. | Cad. Seville     | 15,906 | 21  | 1.134269  |
| 28. | Linc. Versailles | 13,466 | 14  | -.6069287 |
| 42. | Plym. Arrow      | 4,647  | 28  | -.8925859 |
| 43. | Plym. Champ      | 4,425  | 34  | .3186909  |

Once again we see the Cadillac Seville heading the list, indicating that our regression results may be dominated by this one car.

◀

### ► Example 15: DFBETAs influence measure; the dfbeta command

We can use `predict, dfbeta()` or the `dfbeta` command to generate the DFBETAs. `dfbeta` makes up names for the new variables automatically and, without arguments, generates the DFBETAs for all the variables in the regression:

```
. dfbeta
 _dfbeta_1: dfbeta(weight)
 _dfbeta_2: dfbeta(1.foreign)
 _dfbeta_3: dfbeta(mpg)
 _dfbeta_4: dfbeta(1.foreign#c.mpg)
```

`dfbeta` created four new variables in our dataset: `_dfbeta_1`, containing the DFBETAs for `weight`; `_dfbeta_2`, containing the DFBETAs for `mpg`; and so on. Had we wanted only the DFBETAs for `mpg` and `weight`, we might have typed

```
. dfbeta mpg weight
 _dfbeta_5: dfbeta(weight)
 _dfbeta_6: dfbeta(mpg)
```

In the example above, we typed `dfbeta mpg weight` instead of `dfbeta`; if we had typed `dfbeta` followed by `dfbeta mpg weight`, here is what would have happened:

```
. dfbeta
 _dfbeta_7: dfbeta(weight)
 _dfbeta_8: dfbeta(1.foreign)
 _dfbeta_9: dfbeta(mpg)
 _dfbeta_10: dfbeta(1.foreign#c.mpg)

. dfbeta mpg weight
 _dfbeta_11: dfbeta(weight)
 _dfbeta_12: dfbeta(mpg)
```

`dfbeta` would have made up different names for the new variables. `dfbeta` never replaces existing variables—it instead makes up a different name, so we need to pay attention to `dfbeta`'s output.

◀



## Formal tests for violations of assumptions

This section introduces some regression diagnostic commands that are designed to test for certain violations that `rvfplot` less formally attempts to detect. `estat ovtest` provides Ramsey’s test for omitted variables—a pattern in the residuals. `estat hettest` provides a test for heteroskedasticity—the increasing or decreasing variation in the residuals with fitted values, with respect to the explanatory variables, or with respect to yet other variables. The score test implemented in `estat hettest` (Breusch and Pagan 1979; Cook and Weisberg 1983) performs a score test of the null hypothesis that  $b = 0$  against the alternative hypothesis of multiplicative heteroskedasticity. `estat szroeter` provides a rank test for heteroskedasticity, which is an alternative to the score test computed by `estat hettest`. Finally, `estat imtest` computes an information matrix test, including an orthogonal decomposition into tests for heteroskedasticity, skewness, and kurtosis (Cameron and Trivedi 1990). The heteroskedasticity test computed by `estat imtest` is similar to the general test for heteroskedasticity that was proposed by White (1980). Cameron and Trivedi (2009, chap. 3) discuss most of these tests and provides more examples.

### ► Example 16: `estat ovtest`, `estat hettest`, `estat szroeter`, and `estat imtest`

We run these commands just mentioned on our model:

```
. estat ovtest
Ramsey RESET test using powers of the fitted values of price
 Ho: model has no omitted variables
 F(3, 66) = 7.77
 Prob > F = 0.0002

. estat hettest
Breusch-Pagan / Cook-Weisberg tests for heteroskedasticity
 Ho: Constant variance
 variables: fitted values of price
 chi2(1) = 6.50
 Prob > chi2 = 0.0108
```

Testing for heteroskedasticity in the right-hand-side variables is requested by specifying the `rhs` option. By specifying the `mtest(bonferroni)` option, we request that tests be conducted for each of the variables, with a Bonferroni adjustment for the  $p$ -values to accommodate our testing multiple hypotheses.

```
. estat hettest, rhs mtest(bonf)
Breusch-Pagan / Cook-Weisberg test for heteroskedasticity
 Ho: Constant variance
```

| Variable     | chi2  | df | p        |
|--------------|-------|----|----------|
| weight       | 15.24 | 1  | 0.0004 # |
| 1.foreign    | 6.15  | 1  | 0.0525 # |
| mpg          | 9.04  | 1  | 0.0106 # |
| foreign#     |       |    |          |
| c.mpg        |       |    |          |
| 1            | 6.02  | 1  | 0.0566 # |
| simultaneous | 15.60 | 4  | 0.0036   |

# Bonferroni adjusted  $p$ -values

```
. estat szroeter, rhs mtest(holm)

Szroeter's test for homoskedasticity
Ho: variance constant
Ha: variance monotonic in variable
```

| Variable  | chi2  | df | p        |
|-----------|-------|----|----------|
| weight    | 17.07 | 1  | 0.0001 # |
| 1.foreign | 6.15  | 1  | 0.0131 # |
| mpg       | 11.45 | 1  | 0.0021 # |
| foreign#  |       |    |          |
| c.mpg     |       |    |          |
| 1         | 6.17  | 1  | 0.0260 # |

# Holm adjusted p-values

Finally, we request the information matrix test, which is a conditional moments test with second-, third-, and fourth-order moment conditions.

```
. estat imtest

Cameron & Trivedi's decomposition of IM-test
```

| Source             | chi2  | df | p      |
|--------------------|-------|----|--------|
| Heteroskedasticity | 18.86 | 10 | 0.0420 |
| Skewness           | 11.69 | 4  | 0.0198 |
| Kurtosis           | 2.33  | 1  | 0.1273 |
| Total              | 32.87 | 15 | 0.0049 |

We find evidence for omitted variables, heteroskedasticity, and nonnormal skewness.

So, why bother with the various graphical commands when the tests seem so much easier to interpret? In part, it is a matter of taste: both are designed to uncover the same problem, and both are, in fact, going about it in similar ways. One is based on a formal calculation, whereas the other is based on personal judgment in evaluating a graph. On the other hand, the tests are seeking evidence of specific problems, whereas judgment is more general. The careful analyst will use both.

We performed the omitted-variable test first. Omitted variables are a more serious problem than heteroskedasticity or the violations of higher moment conditions tested by `estat imtest`. If this were not a manual, having found evidence of omitted variables, we would never have run the `estat hettest`, `estat szroeter`, and `estat imtest` commands, at least not until we solved the omitted-variable problem.



□ Technical note

`estat ovtest` and `estat hettest` both perform two flavors of their respective tests. By default, `estat ovtest` looks for evidence of omitted variables by fitting the original model augmented by  $\hat{y}^2$ ,  $\hat{y}^3$ , and  $\hat{y}^4$ , which are the fitted values from the original model. Under the assumption of no misspecification, the coefficients on the powers of the fitted values will be zero. With the `rhs` option, `estat ovtest` instead augments the original model with powers (second through fourth) of the explanatory variables (except for dummy variables).

`estat hettest`, by default, looks for heteroskedasticity by modeling the variance as a function of the fitted values. If, however, we specify a variable or variables, the variance will be modeled as a function of the specified variables. In our example, if we had, a priori, some reason to suspect heteroskedasticity and that the heteroskedasticity is a function of a car's weight, then using a test that focuses on weight would be more powerful than the more general tests such as White's test or the first term in the Cameron–Trivedi decomposition test.

`estat hettest`, by default, computes the original Breusch–Pagan/Cook–Weisberg test, which includes the assumption of normally distributed errors. [Koenker \(1981\)](#) derived an  $N * R^2$  version of this test that drops the normality assumption. [Wooldridge \(2009\)](#) gives an  $F$  statistic version that does not require the normality assumption.

□

## Variance inflation factors

Problems arise in regression when the predictors are highly correlated. In this situation, there may be a significant change in the regression coefficients if you add or delete an independent variable. The estimated standard errors of the fitted coefficients are inflated, or the estimated coefficients may not be statistically significant even though a statistical relation exists between the dependent and independent variables.

Data analysts rely on these facts to check informally for the presence of multicollinearity. `estat vif`, another command for use after `regress`, calculates the variance inflation factors and tolerances for each of the independent variables.

The output shows the variance inflation factors together with their reciprocals. Some analysts compare the reciprocals with a predetermined tolerance. In the comparison, if the reciprocal of the VIF is smaller than the tolerance, the associated predictor variable is removed from the regression model. However, most analysts rely on informal rules of thumb applied to the VIF; see [Chatterjee and Hadi \(2006\)](#). According to these rules, there is evidence of multicollinearity if

1. The largest VIF is greater than 10 (some choose a more conservative threshold value of 30).
2. The mean of all the VIFs is considerably larger than 1.

### ► Example 17: `estat vif`

We examine a regression model fit using the ubiquitous automobile dataset:

|                                                                                    |           |           |            |                        |                      |          |
|------------------------------------------------------------------------------------|-----------|-----------|------------|------------------------|----------------------|----------|
| <code>. regress price mpg rep78 trunk headroom length turn displ gear_ratio</code> |           |           |            |                        |                      |          |
| Source                                                                             | SS        | df        | MS         | Number of obs = 69     |                      |          |
| Model                                                                              | 264102049 | 8         | 33012756.2 | F( 8, 60) = 6.33       |                      |          |
| Residual                                                                           | 312694909 | 60        | 5211581.82 | Prob > F = 0.0000      |                      |          |
|                                                                                    |           |           |            | R-squared = 0.4579     |                      |          |
|                                                                                    |           |           |            | Adj R-squared = 0.3856 |                      |          |
| Total                                                                              | 576796959 | 68        | 8482308.22 | Root MSE = 2282.9      |                      |          |
| price                                                                              | Coef.     | Std. Err. | t          | P> t                   | [95% Conf. Interval] |          |
| mpg                                                                                | -144.84   | 82.12751  | -1.76      | 0.083                  | -309.1195            | 19.43948 |
| rep78                                                                              | 727.5783  | 337.6107  | 2.16       | 0.035                  | 52.25638             | 1402.9   |
| trunk                                                                              | 44.02061  | 108.141   | 0.41       | 0.685                  | -172.2935            | 260.3347 |
| headroom                                                                           | -807.0996 | 435.5802  | -1.85      | 0.069                  | -1678.39             | 64.19061 |
| length                                                                             | -8.688914 | 34.89848  | -0.25      | 0.804                  | -78.49626            | 61.11843 |
| turn                                                                               | -177.9064 | 137.3455  | -1.30      | 0.200                  | -452.6383            | 96.82551 |
| displacement                                                                       | 30.73146  | 7.576952  | 4.06       | 0.000                  | 15.5753              | 45.88762 |
| gear_ratio                                                                         | 1500.119  | 1110.959  | 1.35       | 0.182                  | -722.1303            | 3722.368 |
| _cons                                                                              | 6691.976  | 7457.906  | 0.90       | 0.373                  | -8226.057            | 21610.01 |

```
. estat vif
```

| Variable     | VIF  | 1/VIF    |
|--------------|------|----------|
| length       | 8.22 | 0.121614 |
| displacement | 6.50 | 0.153860 |
| turn         | 4.85 | 0.205997 |
| gear_ratio   | 3.45 | 0.290068 |
| mpg          | 3.03 | 0.330171 |
| trunk        | 2.88 | 0.347444 |
| headroom     | 1.80 | 0.554917 |
| rep78        | 1.46 | 0.686147 |
| Mean VIF     | 4.02 |          |

The results are mixed. Although we have no VIFs greater than 10, the mean VIF is greater than 1, though not considerably so. We could continue the investigation of collinearity, but given that other authors advise that collinearity is a problem only when VIFs exist that are greater than 30 (contradicting our rule above), we will not do so here.



► Example 18: estat vif, with strong evidence of multicollinearity

This example comes from a dataset described in [Kutner, Nachtsheim, and Neter \(2004, 257\)](#) that examines body fat as modeled by caliper measurements on the triceps, midarm, and thigh.

```
. use http://www.stata-press.com/data/r11/bodyfat, clear
(Body Fat)
. regress bodyfat tricep thigh midarm
```

| Source   | SS         | df | MS         | Number of obs = | 20     |
|----------|------------|----|------------|-----------------|--------|
| Model    | 396.984607 | 3  | 132.328202 | F( 3, 16) =     | 21.52  |
| Residual | 98.4049068 | 16 | 6.15030667 | Prob > F =      | 0.0000 |
| Total    | 495.389513 | 19 | 26.0731323 | R-squared =     | 0.8014 |
|          |            |    |            | Adj R-squared = | 0.7641 |
|          |            |    |            | Root MSE =      | 2.48   |

| bodyfat | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|---------|-----------|-----------|-------|-------|----------------------|----------|
| triceps | 4.334085  | 3.015511  | 1.44  | 0.170 | -2.058512            | 10.72668 |
| thigh   | -2.856842 | 2.582015  | -1.11 | 0.285 | -8.330468            | 2.616785 |
| midarm  | -2.186056 | 1.595499  | -1.37 | 0.190 | -5.568362            | 1.19625  |
| _cons   | 117.0844  | 99.78238  | 1.17  | 0.258 | -94.44474            | 328.6136 |

```
. estat vif
```

| Variable | VIF    | 1/VIF    |
|----------|--------|----------|
| triceps  | 708.84 | 0.001411 |
| thigh    | 564.34 | 0.001772 |
| midarm   | 104.61 | 0.009560 |
| Mean VIF | 459.26 |          |

Here we see strong evidence of multicollinearity in our model. More investigation reveals that the measurements on the thigh and the triceps are highly correlated:

```
. corr triceps thigh midarm
(obs=20)
```

|         | triceps | thigh  | midarm |
|---------|---------|--------|--------|
| triceps | 1.0000  |        |        |
| thigh   | 0.9238  | 1.0000 |        |
| midarm  | 0.4578  | 0.0847 | 1.0000 |

If we remove the predictor `tricep` from the model (because it had the highest VIF), we get

```
. regress bodyfat thigh midarm
```

| Source   | SS         | df | MS         | Number of obs = | 20     |
|----------|------------|----|------------|-----------------|--------|
| Model    | 384.279748 | 2  | 192.139874 | F( 2, 17) =     | 29.40  |
| Residual | 111.109765 | 17 | 6.53586854 | Prob > F =      | 0.0000 |
|          |            |    |            | R-squared =     | 0.7757 |
|          |            |    |            | Adj R-squared = | 0.7493 |
| Total    | 495.389513 | 19 | 26.0731323 | Root MSE =      | 2.5565 |

| bodyfat | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|---------|-----------|-----------|-------|-------|----------------------|----------|
| thigh   | .8508818  | .1124482  | 7.57  | 0.000 | .6136367             | 1.088127 |
| midarm  | .0960295  | .1613927  | 0.60  | 0.560 | -.2444792            | .4365383 |
| _cons   | -25.99696 | 6.99732   | -3.72 | 0.002 | -40.76001            | -11.2339 |

```
. estat vif
```

| Variable | VIF  | 1/VIF    |
|----------|------|----------|
| midarm   | 1.01 | 0.992831 |
| thigh    | 1.01 | 0.992831 |
| Mean VIF | 1.01 |          |

Note how the coefficients change and how the estimated standard errors for each of the regression coefficients become much smaller. The calculated value of  $R^2$  for the overall regression for the subset model does not appreciably decline when we remove the correlated predictor. Removing an independent variable from the model is one way to deal with multicollinearity. Other methods include ridge regression, weighted least squares, and restricting the use of the fitted model to data that follow the same pattern of multicollinearity. In economic studies, it is sometimes possible to estimate the regression coefficients from different subsets of the data by using cross-section and time series.

◀

All examples above demonstrated the use of centered VIFs. As pointed out by [Belsley \(1991\)](#), the centered VIFs may fail to discover collinearity involving the constant term. One solution is to use the uncentered VIFs instead. According to the definition of the uncentered VIFs, the constant is viewed as a legitimate explanatory variable in a regression model, which allows one to obtain the VIF value for the constant term.

#### ► Example 19: `estat vif`, with strong evidence of collinearity with the constant term

Consider the extreme example in which one of the regressors is highly correlated with the constant. We simulate the data and examine both centered and uncentered VIF diagnostics after fitted regression model as follows.

```
. use http://www.stata-press.com/data/r11/extreme_collin
. summarize
(output omitted)
. regress y one x z
```

| Source   | SS         | df | MS         |                        |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | 223801.985 | 3  | 74600.6617 | Number of obs = 100    |  |  |
| Residual | 2642.42124 | 96 | 27.5252213 | F( 3, 96) = 2710.27    |  |  |
| Total    | 226444.406 | 99 | 2287.31723 | Prob > F = 0.0000      |  |  |
|          |            |    |            | R-squared = 0.9883     |  |  |
|          |            |    |            | Adj R-squared = 0.9880 |  |  |
|          |            |    |            | Root MSE = 5.2464      |  |  |

| y     | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|-------|-----------|-----------|-------|-------|----------------------|----------|
| one   | -3.278582 | 10.5621   | -0.31 | 0.757 | -24.24419            | 17.68702 |
| x     | 2.038696  | .0242673  | 84.01 | 0.000 | 1.990526             | 2.086866 |
| z     | 4.863137  | .2681036  | 18.14 | 0.000 | 4.330956             | 5.395319 |
| _cons | 9.760075  | 10.50935  | 0.93  | 0.355 | -11.10082            | 30.62097 |

```
. estat vif
```

| Variable | VIF  | 1/VIF    |
|----------|------|----------|
| z        | 1.03 | 0.968488 |
| x        | 1.03 | 0.971307 |
| one      | 1.00 | 0.995425 |
| Mean VIF | 1.02 |          |

```
. estat vif, uncentered
```

| Variable  | VIF    | 1/VIF    |
|-----------|--------|----------|
| one       | 402.94 | 0.002482 |
| intercept | 401.26 | 0.002492 |
| z         | 2.93   | 0.341609 |
| x         | 1.13   | 0.888705 |
| Mean VIF  | 202.06 |          |

According to the values of the centered VIFs (1.03, 1.03, 1.00), no harmful collinearity is detected in the model. However, by the construction of these simulated data, we know that `one` is highly collinear with the constant term. As such, the large values of uncentered VIFs for `one` (402.94) and `intercept` (401.26) reveal high collinearity of the variable `one` with the constant term.



Saved results

`estat hettest` saves the following results for the (multivariate) score test in `r()`:

- Scalars
- `r(chi2)`

$\chi^2$  test statistic
- `r(df)`

#df for the asymptotic  $\chi^2$  distribution under  $H_0$
- `r(p)`

p-value

`estat hettest`, `fstat` saves results for the (multivariate) score test in `r()`:

#### Scalars

`r(F)` test statistic  
`r(df_m)` #df of the test for the  $F$  distribution under  $H_0$   
`r(df_r)` #df of the residuals for the  $F$  distribution under  $H_0$   
`r(p)`  $p$ -value

`estat hettest` (if `mtest` is specified) and `estat szroeter` save the following in `r()`:

#### Matrices

`r(mtest)` a matrix of test results, with rows corresponding to the univariate tests

`mtest[.,1]`  $\chi^2$  test statistic  
`mtest[.,2]` #df  
`mtest[.,3]` unadjusted  $p$ -value  
`mtest[.,4]` adjusted  $p$ -value (if an `mtest()` adjustment method is specified)

#### Macros

`r(mtmetho)` adjustment method for  $p$ -values

`estat imtest` saves the following in `r()`:

#### Scalars

`r(chi2_t)` IM-test statistic ( $= r(chi2_h) + r(chi2_s) + r(chi2_k)$ )  
`r(df_t)` df for limiting  $\chi^2$  distribution under  $H_0$  ( $= r(df_h) + r(df_s) + r(df_k)$ )  
`r(chi2_h)` heteroskedasticity test statistic  
`r(df_h)` df for limiting  $\chi^2$  distribution under  $H_0$   
`r(chi2_s)` skewness test statistic  
`r(df_s)` df for limiting  $\chi^2$  distribution under  $H_0$   
`r(chi2_k)` kurtosis test statistic  
`r(df_k)` df for limiting  $\chi^2$  distribution under  $H_0$   
`r(chi2_w)` White's heteroskedasticity test (if `white` specified)  
`r(df_w)` df for limiting  $\chi^2$  distribution under  $H_0$

`estat ovtest` saves the following in `r()`:

#### Scalars

|                   |                      |                      |                             |
|-------------------|----------------------|----------------------|-----------------------------|
| <code>r(p)</code> | two-sided $p$ -value | <code>r(df)</code>   | degrees of freedom          |
| <code>r(F)</code> | $F$ statistic        | <code>r(df_r)</code> | residual degrees of freedom |

## Methods and formulas

All regression fit and diagnostic commands are implemented as ado-files.

See [Hamilton \(2009, chap. 7\)](#), [Kohler and Kreuter \(2009, sec. 8.3\)](#), or [Baum \(2006, chap. 5\)](#) for an overview of using Stata to perform regression diagnostics. See [Peracchi \(2001, chap. 8\)](#) for a mathematically rigorous discussion of diagnostics.

## Methods and formulas for predict

Assume that you have already fit the regression model

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{e}$$

where  $\mathbf{X}$  is  $n \times k$ .

Denote the previously estimated coefficient vector by  $\mathbf{b}$  and its estimated variance matrix by  $\mathbf{V}$ . `predict` works by recalling various aspects of the model, such as  $\mathbf{b}$ , and combining that information with the data currently in memory. Let  $\mathbf{x}_j$  be the  $j$ th observation currently in memory, and let  $s^2$  be the mean squared error of the regression.

Let  $\mathbf{V} = s^2(\mathbf{X}'\mathbf{X})^{-1}$ . Let  $k$  be the number of independent variables including the intercept, if any, and let  $y_j$  be the observed value of the dependent variable.

The *predicted value* (`xb` option) is defined as  $\hat{y}_j = \mathbf{x}_j\mathbf{b}$ .

Let  $\ell_j$  represent a lower bound for an observation  $j$  and  $u_j$  represent an upper bound. The probability that  $y_j|\mathbf{x}_j$  would be observed in the interval  $(\ell_j, u_j)$ —the `pr( $\ell$ ,  $u$ )` option—is

$$P(\ell_j, u_j) = \Pr(\ell_j < \mathbf{x}_j\mathbf{b} + e_j < u_j) = \Phi\left(\frac{u_j - \hat{y}_j}{s}\right) - \Phi\left(\frac{\ell_j - \hat{y}_j}{s}\right)$$

where for the `pr( $\ell$ ,  $u$ )`, `e( $\ell$ ,  $u$ )`, and `ystar( $\ell$ ,  $u$ )` options,  $\ell_j$  and  $u_j$  can be anywhere in the range  $(-\infty, +\infty)$ .

The option `e( $\ell$ ,  $u$ )` computes the expected value of  $y_j|\mathbf{x}_j$  conditional on  $y_j|\mathbf{x}_j$  being in the interval  $(\ell_j, u_j)$ , that is, when  $y_j|\mathbf{x}_j$  is censored. It can be expressed as

$$E(\ell_j, u_j) = E(\mathbf{x}_j\mathbf{b} + e_j \mid \ell_j < \mathbf{x}_j\mathbf{b} + e_j < u_j) = \hat{y}_j - s \frac{\phi\left(\frac{u_j - \hat{y}_j}{s}\right) - \phi\left(\frac{\ell_j - \hat{y}_j}{s}\right)}{\Phi\left(\frac{u_j - \hat{y}_j}{s}\right) - \Phi\left(\frac{\ell_j - \hat{y}_j}{s}\right)}$$

where  $\phi$  is the normal density and  $\Phi$  is the cumulative normal.

You can also compute `ystar( $\ell$ ,  $u$ )`—the expected value of  $y_j|\mathbf{x}_j$ , where  $y_j$  is assumed truncated at  $\ell_j$  and  $u_j$ :

$$y_j^* = \begin{cases} \ell_j & \text{if } \mathbf{x}_j\mathbf{b} + e_j \leq \ell_j \\ \mathbf{x}_j\mathbf{b} + u & \text{if } \ell_j < \mathbf{x}_j\mathbf{b} + e_j < u_j \\ u_j & \text{if } \mathbf{x}_j\mathbf{b} + e_j \geq u_j \end{cases}$$

This computation can be expressed in several ways, but the most intuitive formulation involves a combination of the two statistics just defined:

$$y_j^* = P(-\infty, \ell_j)\ell_j + P(\ell_j, u_j)E(\ell_j, u_j) + P(u_j, +\infty)u_j$$

A diagonal element of the projection matrix (`hat`) or (`leverage`) is given by

$$h_j = \mathbf{x}_j(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_j'$$

The *standard error of the prediction* (the `stdp` option) is defined as  $s_{p_j} = \sqrt{\mathbf{x}_j\mathbf{V}\mathbf{x}_j'}$

and can also be written as  $s_{p_j} = s\sqrt{h_j}$ .



The *standard error of the forecast* (`stdf`) is defined as  $s_{f_j} = s\sqrt{1 + h_j}$ .

The *standard error of the residual* (`stdr`) is defined as  $s_{r_j} = s\sqrt{1 - h_j}$ .

The *residuals* (`residuals`) are defined as  $\hat{e}_j = y_j - \hat{y}_j$ .

The *standardized residuals* (`rstandard`) are defined as  $\hat{e}_{s_j} = \hat{e}_j / s_{r_j}$ .

The *studentized residuals* (`rstudent`) are defined as

$$r_j = \frac{\hat{e}_j}{s_{(j)}\sqrt{1 - h_j}}$$

where  $s_{(j)}$  represents the root mean squared error with the  $j$ th observation removed, which is given by

$$s_{(j)}^2 = \frac{s^2(T - k)}{T - k - 1} - \frac{\hat{e}_j^2}{(T - k - 1)(1 - h_j)}$$

Cook's  $D$  (`cooksD`) is given by

$$D_j = \frac{\hat{e}_{s_j}^2 (s_{p_j} / s_{r_j})^2}{k} = \frac{h_j \hat{e}_j^2}{ks^2(1 - h_j)^2}$$

DFITS (`dfits`) is given by

$$\text{DFITS}_j = r_j \sqrt{\frac{h_j}{1 - h_j}}$$

Welsch distance (`welsch`) is given by

$$W_j = \frac{r_j \sqrt{h_j(n - 1)}}{1 - h_j}$$

COVRATIO (`covratio`) is given by

$$\text{COVRATIO}_j = \frac{1}{1 - h_j} \left( \frac{n - k - \hat{e}_j^2}{n - k - 1} \right)^k$$

The DFBETAS (`dfbeta`) for a particular regressor  $x_i$  are given by

$$\text{DFBETA}_j = \frac{r_j u_j}{\sqrt{U^2(1 - h_j)}}$$

where  $u_j$  are the residuals obtained from a regression of  $x_j$  on the remaining  $x$ 's and  $U^2 = \sum_j u_j^2$ .

The omitted-variable test ([Ramsey 1969](#)) reported by `ovtest` fits the regression  $y_i = \mathbf{x}_i \mathbf{b} + \mathbf{z}_i \mathbf{t} + u_i$  and then performs a standard  $F$  test of  $\mathbf{t} = \mathbf{0}$ . The default test uses  $\mathbf{z}_i = (\hat{y}_i^2, \hat{y}_i^3, \hat{y}_i^4)$ . If `rhs` is specified,  $\mathbf{z}_i = (x_{1i}^2, x_{1i}^3, x_{1i}^4, x_{2i}^2, \dots, x_{mi}^4)$ . In either case, the variables are normalized to have minimum 0 and maximum 1 before powers are calculated.

The test for heteroskedasticity (Breusch and Pagan 1979; Cook and Weisberg 1983) models  $\text{Var}(e_i) = \sigma^2 \exp(\mathbf{z}\mathbf{t})$ , where  $\mathbf{z}$  is a variable list specified by the user, the list of right-hand-side variables, or the fitted values  $\mathbf{x}\hat{\beta}$ . The test is of  $\mathbf{t} = \mathbf{0}$ . Mechanically, `estat hettest` fits the augmented regression  $\hat{e}_i^2/\hat{\sigma}^2 = a + \mathbf{z}_i\mathbf{t} + v_i$ .

The original Breusch–Pagan/Cook–Weisberg version of the test assumes that the  $e_i$  are normally distributed under the null hypothesis which implies that the score test statistic  $S$  is equal to the model sum of squares from the augmented regression divided by 2. Under the null hypothesis,  $S$  has the  $\chi^2$  distribution with  $m$  degrees of freedom, where  $m$  is the number of columns of  $\mathbf{z}$ .

Koenker (1981) derived a score test of the null hypothesis that  $\mathbf{t} = \mathbf{0}$  under the assumption that the  $e_i$  are independent and identically distributed (i.i.d.). Koenker showed that  $S = N * R^2$  has a large-sample  $\chi^2$  distribution with  $m$  degrees of freedom, where  $N$  is the number of observations and  $R^2$  is the  $R$ -squared in the augmented regression and  $m$  is the number of columns of  $\mathbf{z}$ . `estat hettest`, `iid` produces this version of the test.

Wooldridge (2009) showed that an  $F$  test of  $\mathbf{t} = \mathbf{0}$  in the augmented regression can also be used under the assumption that the  $e_i$  are i.i.d. `estat hettest`, `fstat` produces this version of the test.

Szroeter's class of tests for homoskedasticity against the alternative that the residual variance increases in some variable  $x$  is defined in terms of

$$H = \frac{\sum_{i=1}^n h(x_i)e_i^2}{\sum_{i=1}^n e_i^2}$$

where  $h(x)$  is some weight function that increases in  $x$  (Szroeter 1978).  $H$  is a weighted average of the  $h(x)$ , with the squared residuals serving as weights. Under homoskedasticity,  $H$  should be approximately equal to the unweighted average of  $h(x)$ . Large values of  $H$  suggest that  $e_i^2$  tends to be large where  $h(x)$  is large; i.e., the variance indeed increases in  $x$ , whereas small values of  $H$  suggest that the variance actually decreases in  $x$ . `estat szroeter` uses  $h(x_i) = \text{rank}(x_i \text{ in } x_1 \dots x_n)$ ; see Judge et al. [1985, 452] for details. `estat szroeter` displays a normalized version of  $H$ ,

$$Q = \sqrt{\frac{6n}{n^2 - 1}} H$$

which is approximately  $N(0, 1)$  distributed under the null (homoskedasticity).

`estat hettest` and `estat szroeter` provide adjustments of  $p$ -values for multiple testing. The supported methods are described in [R] [test](#).

`estat imtest` performs the information matrix test for the regression model, as well as an orthogonal decomposition into tests for heteroskedasticity  $\delta_1$ , nonnormal skewness  $\delta_2$ , and nonnormal kurtosis  $\delta_3$  (Cameron and Trivedi 1990; Long and Trivedi 1992). The decomposition is obtained via three auxiliary regressions. Let  $e$  be the regression residuals,  $\hat{\sigma}^2$  be the maximum likelihood estimate of  $\sigma^2$  in the regression,  $n$  be the number of observations,  $X$  be the set of  $k$  variables specified with `estat imtest`, and  $R_{\text{un}}^2$  be the uncentered  $R^2$  from a regression.  $\delta_1$  is obtained as  $nR_{\text{un}}^2$  from a regression of  $e^2 - \hat{\sigma}^2$  on the cross-products of the variables in  $X$ .  $\delta_2$  is computed as  $nR_{\text{un}}^2$  from a regression of  $e^3 - 3\hat{\sigma}^2 e$  on  $X$ . Finally,  $\delta_3$  is obtained as  $nR_{\text{un}}^2$  from a regression of  $e^4 - 6\hat{\sigma}^2 e^2 - 3\hat{\sigma}^4$  on  $X$ .  $\delta_1$ ,  $\delta_2$ , and  $\delta_3$  are asymptotically  $\chi^2$  distributed with  $1/2k(k+1)$ ,  $K$ , and 1 degree of freedom. The information test statistic  $\delta = \delta_1 + \delta_2 + \delta_3$  is asymptotically  $\chi^2$  distributed with  $1/2k(k+3)$  degrees of freedom. White's test for heteroskedasticity is computed as  $nR^2$  from a regression of  $\hat{u}^2$  on  $X$  and the cross-products of the variables in  $X$ . This test statistic is usually close to  $\delta_1$ .

`estat vif` calculates the centered variance inflation factor ( $\text{VIF}_c$ ) (Chatterjee and Hadi 2006, 235–239) for  $x_j$ , given by

$$\text{VIF}_c(x_j) = \frac{1}{1 - \widehat{R}_j^2}$$

where  $\widehat{R}_j^2$  is the square of the centered multiple correlation coefficient that results when  $x_j$  is regressed with intercept against all the other explanatory variables.

The uncentered variance inflation factor ( $\text{VIF}_{uc}$ ) (Belsley 1991, 28–29) for  $x_j$  is given by

$$\text{VIF}_{uc}(x_j) = \frac{1}{1 - \widetilde{R}_j^2}$$

where  $\widetilde{R}_j^2$  is the square of the uncentered multiple correlation coefficient that results when  $x_j$  is regressed without intercept against all the other explanatory variables including the constant term.

## Acknowledgments

`estat ovtest` and `estat hettest` are based on programs originally written by Richard Goldstein (1991, 1992). `estat imtest`, `estat szroeter`, and the current version of `estat hettest` were written by Jeroen Weesie, Department of Sociology, Utrecht University, The Netherlands; `estat imtest` is based in part on code written by J. Scott Long, Department of Sociology, Indiana University.

## References

- Adkins, L. C., and R. C. Hill. 2008. *Using Stata for Principles of Econometrics*. 3rd ed. Hoboken, NJ: Wiley.
- Baum, C. F. 2006. *An Introduction to Modern Econometrics Using Stata*. College Station, TX: Stata Press.
- Baum, C. F., N. J. Cox, and V. L. Wiggins. 2000. sg137: Tests for heteroskedasticity in regression error distribution. *Stata Technical Bulletin* 55: 15–17. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 147–149. College Station, TX: Stata Press.
- Baum, C. F., and V. L. Wiggins. 2000a. sg135: Test for autoregressive conditional heteroskedasticity in regression error distribution. *Stata Technical Bulletin* 55: 13–14. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 143–144. College Station, TX: Stata Press.
- . 2000b. sg136: Tests for serial correlation in regression error distribution. *Stata Technical Bulletin* 55: 14–15. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 145–147. College Station, TX: Stata Press.
- Belsley, D. A. 1991. *Conditional Diagnostics: Collinearity and Weak Data in Regression*. New York: Wiley.
- Belsley, D. A., E. Kuh, and R. E. Welsch. 1980. *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. New York: Wiley.
- Bollen, K. A., and R. W. Jackman. 1990. Regression diagnostics: An expository treatment of outliers and influential cases. In *Modern Methods of Data Analysis*, ed. J. Fox and J. S. Long, 257–291. Newbury Park, CA: Sage.
- Breusch, T. S., and A. R. Pagan. 1979. A simple test for heteroscedasticity and random coefficient variation. *Econometrica* 47: 1287–1294.
- Cameron, A. C., and P. K. Trivedi. 1990. The information matrix test and its applied alternative hypotheses. Working paper 372, University of California–Davis, Institute of Governmental Affairs.
- . 2009. *Microeconometrics Using Stata*. College Station, TX: Stata Press.
- Chatterjee, S., and A. S. Hadi. 1986. Influential observations, high leverage points, and outliers in linear regression. *Statistical Science* 1: 379–393.
- . 1988. *Sensitivity Analysis in Linear Regression*. New York: Wiley.
- . 2006. *Regression Analysis by Example*. 4th ed. New York: Wiley.
- Cook, R. D. 1977. Detection of influential observation in linear regression. *Technometrics* 19: 15–18.

- Cook, R. D., and S. Weisberg. 1982. *Residuals and Influence in Regression*. New York: Chapman & Hall/CRC.
- . 1983. Diagnostics for heteroscedasticity in regression. *Biometrika* 70: 1–10.
- Cox, N. J. 2004. [Speaking Stata: Graphing model diagnostics](#). *Stata Journal* 4: 449–475.
- DeMaris, A. 2004. *Regression with Social Data: Modeling Continuous and Limited Response Variables*. Hoboken, NJ: Wiley.
- Ezekiel, M. 1924. A method of handling curvilinear correlation for any number of variables. *Journal of the American Statistical Association* 19: 431–453.
- Garrett, J. M. 2000. [sg157: Predicted values calculated from linear or logistic regression models](#). *Stata Technical Bulletin* 58: 27–30. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 258–261. College Station, TX: Stata Press.
- Goldstein, R. 1991. [srd5: Ramsey test for heteroscedasticity and omitted variables](#). *Stata Technical Bulletin* 2: 27. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, p. 177. College Station, TX: Stata Press.
- . 1992. [srd14: Cook–Weisberg test of heteroscedasticity](#). *Stata Technical Bulletin* 10: 27–28. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 183–184. College Station, TX: Stata Press.
- Hamilton, L. C. 1992. *Regression with Graphics: A Second Course in Applied Statistics*. Belmont, CA: Duxbury.
- . 2009. *Statistics with Stata (Updated for Version 10)*. Belmont, CA: Brooks/Cole.
- Hardin, J. W. 1995. [sg32: Variance inflation factors and variance-decomposition proportions](#). *Stata Technical Bulletin* 24: 17–22. Reprinted in *Stata Technical Bulletin Reprints*, vol. 4, pp. 154–160. College Station, TX: Stata Press.
- Hill, R. C., W. E. Griffiths, and G. C. Lim. 2008. *Principles of Econometrics*. 3rd ed. Hoboken, NJ: Wiley.
- Hoaglin, D. C., and P. J. Kempthorne. 1986. Comment [on Chatterjee and Hadi 1986]. *Statistical Science* 1: 408–412.
- Hoaglin, D. C., and R. E. Welsch. 1978. The hat matrix in regression and ANOVA. *American Statistician* 32: 17–22.
- Judge, G. G., W. E. Griffiths, R. C. Hill, H. Lütkepohl, and T.-C. Lee. 1985. *The Theory and Practice of Econometrics*. 2nd ed. New York: Wiley.
- Koenker, R. 1981. A note on studentizing a test for heteroskedasticity. *Journal of Econometrics* 17: 107–112.
- Kohler, U., and F. Kreuter. 2009. *Data Analysis Using Stata*. 2nd ed. College Station, TX: Stata Press.
- Kutner, M. H., C. J. Nachtsheim, and J. Neter. 2004. *Applied Linear Regression Models*. 4th ed. New York: McGraw–Hill/Irwin.
- Larsen, W. A., and S. J. McCleary. 1972. The use of partial residual plots in regression analysis. *Technometrics* 14: 781–790.
- Long, J. S., and J. Freese. 2000. [sg145: Scalar measures of fit for regression models](#). *Stata Technical Bulletin* 56: 34–40. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 197–205. College Station, TX: Stata Press.
- Long, J. S., and P. K. Trivedi. 1992. Some specification tests for the linear regression model. *Sociological Methods and Research* 21: 161–204. Reprinted in *Testing Structural Equation Models*, ed. K. A. Bollen and J. S. Long, pp. 66–110. Newbury Park, CA: Sage.
- Mallows, C. L. 1986. Augmented partial residuals. *Technometrics* 28: 313–319.
- Mosteller, F., and J. W. Tukey. 1977. *Data Analysis and Regression: A Second Course in Statistics*. Reading, MA: Addison–Wesley.
- Peracchi, F. 2001. *Econometrics*. Chichester, UK: Wiley.
- Ramsey, J. B. 1969. Tests for specification errors in classical linear least-squares regression analysis. *Journal of the Royal Statistical Society, Series B* 31: 350–371.
- Ramsey, J. B., and P. Schmidt. 1976. Some further results on the use of OLS and BLUS residuals in specification error tests. *Journal of the American Statistical Association* 71: 389–390.
- Rousseeuw, P. J., and A. M. Leroy. 1987. *Robust Regression and Outlier Detection*. New York: Wiley.
- Szroeter, J. 1978. A class of parametric tests for heteroscedasticity in linear econometric models. *Econometrica* 46: 1311–1327.
- Velleman, P. F. 1986. Comment [on Chatterjee and Hadi 1986]. *Statistical Science* 1: 412–413.
- Velleman, P. F., and R. E. Welsch. 1981. Efficient computing of regression diagnostics. *American Statistician* 35: 234–242.

- Weesie, J. 2001. [sg161: Analysis of the turning point of a quadratic specification](#). *Stata Technical Bulletin* 60: 18–20. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 273–277. College Station, TX: Stata Press.
- Weisberg, S. 2005. *Applied Linear Regression*. 3rd ed. New York: Wiley.
- Welsch, R. E. 1982. Influence functions and regression diagnostics. In *Modern Data Analysis*, ed. R. L. Launer and A. F. Siegel, 149–169. New York: Academic Press.
- . 1986. Comment [on Chatterjee and Hadi 1986]. *Statistical Science* 1: 403–405.
- Welsch, R. E., and E. Kuh. 1977. Linear Regression Diagnostics. Technical Report 923-77, Massachusetts Institute of Technology, Cambridge, MA.
- White, H. 1980. A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica* 48: 817–838.
- Wooldridge, J. M. 2009. *Introductory Econometrics: A Modern Approach*. 4th ed. Cincinnati, OH: South-Western.

## Also see

[R] [regress](#) — Linear regression

[R] [regress postestimation time series](#) — Postestimation tools for regress with time series

[U] [20 Estimation and postestimation commands](#)

## Description

The following postestimation commands for time series are available for `regress`:

| command                      | description                                                            |
|------------------------------|------------------------------------------------------------------------|
| <code>estat archlm</code>    | test for ARCH effects in the residuals                                 |
| <code>estat bgodfrey</code>  | Breusch–Godfrey test for higher-order serial correlation               |
| <code>estat durbinalt</code> | Durbin’s alternative test for serial correlation                       |
| <code>estat dwatson</code>   | Durbin–Watson $d$ statistic to test for first-order serial correlation |

These commands provide regression diagnostic tools specific to time series. You must `tsset` your data before using these commands.

`estat archlm` tests for time-dependent volatility. `estat bgodfrey`, `estat durbinalt`, and `estat dwatson` test for serial correlation in the residuals of a linear regression. For non–time-series regression diagnostic tools, see [\[R\] regress postestimation](#).

`estat archlm` performs Engle’s Lagrange multiplier (LM) test for the presence of autoregressive conditional heteroskedasticity.

`estat bgodfrey` performs the Breusch–Godfrey test for higher-order serial correlation in the disturbance. This test does not require that all the regressors be strictly exogenous.

`estat durbinalt` performs Durbin’s alternative test for serial correlation in the disturbance. This test does not require that all the regressors be strictly exogenous.

`estat dwatson` computes the Durbin–Watson  $d$  statistic ([Durbin and Watson 1950](#)) to test for first-order serial correlation in the disturbance when all the regressors are strictly exogenous.

## Syntax for estat archlm

| <code>estat archlm [ , <i>archlm_options</i> ]</code> |                                                                  |
|-------------------------------------------------------|------------------------------------------------------------------|
| <i>archlm_options</i>                                 | description                                                      |
| <code>lags(<i>numlist</i>)</code>                     | test <i>numlist</i> lag orders                                   |
| <code>force</code>                                    | allow test after <code>regress</code> , <code>vce(robust)</code> |

## Options for estat archlm

`lags(numlist)` specifies a list of numbers, indicating the lag orders to be tested. The test will be performed separately for each order. The default is order one.

`force` allows the test to be run after `regress`, `vce(robust)`. The command will not work if the `vce(cluster clustvar)` option is specified with [\[R\] regress](#).

## Syntax for estat bgodfrey

estat bgodfrey [ , bgodfrey\_options ]

| bgodfrey_options | description                                                            |
|------------------|------------------------------------------------------------------------|
| lags(numlist)    | test numlist lag orders                                                |
| nomiss0          | do not use Davidson and MacKinnon’s approach                           |
| small            | obtain <i>p</i> -values by using the <i>F</i> or <i>t</i> distribution |

## Options for estat bgodfrey

- lags(numlist) specifies a list of numbers, indicating the lag orders to be tested. The test will be performed separately for each order. The default is order one.
- nomiss0 specifies that Davidson and MacKinnon’s approach (1993, 358), which replaces the missing values in the initial observations on the lagged residuals in the auxiliary regression with zeros, not be used.
- small specifies that the *p*-values of the test statistics be obtained using the *F* or *t* distribution instead of the default chi-squared or normal distribution.

## Syntax for estat durbinalt

estat durbinalt [ , durbinalt\_options ]

| durbinalt_options | description                                                         |
|-------------------|---------------------------------------------------------------------|
| lags(numlist)     | test numlist lag orders                                             |
| nomiss0           | do not use Davidson and MacKinnon’s approach                        |
| robust            | compute standard errors using the robust/sandwich estimator         |
| small             | obtain <i>p</i> -values using the <i>F</i> or <i>t</i> distribution |
| force             | allow test after regress, vce(robust) or after newey                |

## Options for estat durbinalt

- lags(numlist) specifies a list of numbers, indicating the lag orders to be tested. The test will be performed separately for each order. The default is order one.
- nomiss0 specifies that Davidson and MacKinnon’s approach (1993, 358), which replaces the missing values in the initial observations on the lagged residuals in the auxiliary regression with zeros, not be used.
- robust specifies that the Huber/White/sandwich robust estimator of the variance–covariance matrix be used in Durbin’s alternative test.
- small specifies that the *p*-values of the test statistics be obtained using the *F* or *t* distribution instead of the default chi-squared or normal distribution. This option may not be specified with robust, which always uses an *F* or *t* distribution.

force allows the test to be run after `regress`, `vce(robust)` and after `newey` (see [R] [regress](#) and [TS] [newey](#)). The command will not work if the `vce(cluster clustvar)` option is specified with [R] [regress](#).

## Syntax for estat dwatson

```
estat dwatson
```

## Remarks

The Durbin–Watson test is used to determine whether the error term in a linear regression model follows an AR(1) process. For the linear model

$$y_t = \mathbf{x}_t\beta + u_t$$

the AR(1) process can be written as

$$u_t = \rho u_{t-1} + \epsilon_t$$

In general, an AR(1) process requires only that  $\epsilon_t$  be independent and identically distributed (i.i.d.). The Durbin–Watson test, however, requires  $\epsilon_t$  to be distributed  $N(0, \sigma^2)$  for the statistic to have an exact distribution. Also, the Durbin–Watson test can be applied only when the regressors are strictly exogenous. A regressor  $x$  is strictly exogenous if  $\text{Corr}(x_s, u_t) = 0$  for all  $s$  and  $t$ , which precludes the use of the Durbin–Watson statistic with models where lagged values of the dependent variable are included as regressors.

The null hypothesis of the test is that there is no first-order autocorrelation. The Durbin–Watson  $d$  statistic can take on values between 0 and 4 and under the null  $d$  is equal to 2. Values of  $d$  less than 2 suggest positive autocorrelation ( $\rho > 0$ ), whereas values of  $d$  greater than 2 suggest negative autocorrelation ( $\rho < 0$ ). Calculating the exact distribution of the  $d$  statistic is difficult, but empirical upper and lower bounds have been established based on the sample size and the number of regressors. Extended tables for the  $d$  statistic have been published by [Savin and White \(1977\)](#). For example, suppose you have a model with 30 observations and three regressors (including the constant term). For a test of the null hypothesis of no autocorrelation versus the alternative of positive autocorrelation, the lower bound of the  $d$  statistic is 1.284, and the upper bound is 1.567 at the 5% significance level. You would reject the null if  $d < 1.284$ , and you would fail to reject if  $d > 1.567$ . A value falling within the range (1.284, 1.567) leads to no conclusion about whether or not to reject the null hypothesis.

When lagged dependent variables are included among the regressors, the past values of the error term are correlated with those lagged variables at time  $t$ , implying that they are not strictly exogenous regressors. The inclusion of covariates that are not strictly exogenous causes the  $d$  statistic to be biased toward the acceptance of the null hypothesis. [Durbin \(1970\)](#) suggested an alternative test for models with lagged dependent variables and extended that test to the more general AR( $p$ ) serial correlation process

$$u_t = \rho_1 u_{t-1} + \cdots + \rho_p u_{t-p} + \epsilon_t$$

where  $\epsilon_t$  is i.i.d. with variance  $\sigma^2$  but is not assumed or required to be normal for the test.

The null hypothesis of Durbin’s alternative test is

$$H_0 : \rho_1 = 0, \dots, \rho_p = 0$$



and the alternative is that at least one of the  $\rho$ 's is nonzero. Although the null hypothesis was originally derived for an  $AR(p)$  process, this test turns out to have power against  $MA(p)$  processes as well. Hence, the actual null of this test is that there is no serial correlation up to order  $p$  because the  $MA(p)$  and the  $AR(p)$  models are locally equivalent alternatives under the null. See [Godfrey \(1988, 113–115\)](#) for a discussion of this result.

Durbin's alternative test is in fact a LM test, but it is most easily computed with a Wald test on the coefficients of the lagged residuals in an auxiliary OLS regression of the residuals on their lags and all the covariates in the original regression. Consider the linear regression model

$$y_t = \beta_1 x_{1t} + \cdots + \beta_k x_{kt} + u_t \quad (1)$$

in which the covariates  $x_1$  through  $x_k$  are not assumed to be strictly exogenous and  $u_t$  is assumed to be i.i.d. and to have finite variance. The process is also assumed to be stationary. (See [Wooldridge \[2009\]](#) for a discussion of stationarity.) Estimating the parameters in (1) by OLS obtains the residuals  $\hat{u}_t$ . Next another OLS regression is performed of  $\hat{u}_t$  on  $\hat{u}_{t-1}, \dots, \hat{u}_{t-p}$  and the other regressors,

$$\hat{u}_t = \gamma_1 \hat{u}_{t-1} + \cdots + \gamma_p \hat{u}_{t-p} + \beta_1 x_{1t} + \cdots + \beta_k x_{kt} + \epsilon_t \quad (2)$$

where  $\epsilon_t$  stands for the random-error term in this auxiliary OLS regression. Durbin's alternative test is then obtained by performing a Wald test that  $\gamma_1, \dots, \gamma_p$  are jointly zero. The test can be made robust to an unknown form of heteroskedasticity by using a robust VCE estimator when estimating the regression in (2). When there are only strictly exogenous regressors and  $p = 1$ , this test is asymptotically equivalent to the Durbin–Watson test.

The Breusch–Godfrey test is also an LM test of the null hypothesis of no autocorrelation versus the alternative that  $u_t$  follows an  $AR(p)$  or  $MA(p)$  process. Like Durbin's alternative test, it is based on the auxiliary regression (2), and it is computed as  $NR^2$ , where  $N$  is the number of observations and  $R^2$  is the simple  $R^2$  from the regression. This test and Durbin's alternative test are asymptotically equivalent. The test statistic  $NR^2$  has an asymptotic  $\chi^2$  distribution with  $p$  degrees of freedom. It is valid with or without the strict exogeneity assumption but is not robust to conditional heteroskedasticity, even if a robust VCE is used when fitting (2).

In fitting (2), the values of the lagged residuals will be missing in the initial periods. As noted by Davidson and MacKinnon (1993), the residuals will not be orthogonal to the other covariates in the model in this restricted sample, which implies that the  $R^2$  from the auxiliary regression will not be zero when the lagged residuals are left out. Hence, Breusch and Godfrey's  $NR^2$  version of the test may overreject in small samples. To correct this problem, Davidson and MacKinnon (1993) recommend setting the missing values of the lagged residuals to zero and running the auxiliary regression in (2) over the full sample used in (1). This small-sample correction has become conventional for both the Breusch–Godfrey and Durbin's alternative test, and it is the default for both commands. Specifying the `nomiss0` option overrides this default behavior and treats the initial missing values generated by regressing on the lagged residuals as missing. Hence, `nomiss0` causes these initial observations to be dropped from the sample of the auxiliary regression.

Durbin's alternative test and the Breusch–Godfrey test were originally derived for the case covered by `regress` without the `vce(robust)` option. However, after `regress`, `vce(robust)` and `newey`, Durbin's alternative test is still valid and can be invoked if the `robust` and `force` options are specified.

➤ Example 1: tests for serial correlation

Using data from [Klein \(1950\)](#), we first fit an OLS regression of consumption on the government wage bill:

```
. use http://www.stata-press.com/data/r11/klein
. tsset yr
 time variable: yr, 1920 to 1941
 delta: 1 unit
. regress consump wagegovt
```

| Source   | SS         | df | MS         |                        |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | 532.567711 | 1  | 532.567711 | Number of obs = 22     |  |  |
| Residual | 601.207167 | 20 | 30.0603584 | F( 1, 20) = 17.72      |  |  |
|          |            |    |            | Prob > F = 0.0004      |  |  |
|          |            |    |            | R-squared = 0.4697     |  |  |
|          |            |    |            | Adj R-squared = 0.4432 |  |  |
|          |            |    |            | Root MSE = 5.4827      |  |  |
| Total    | 1133.77488 | 21 | 53.9892799 |                        |  |  |

|          | Coef.    | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|----------|----------|-----------|-------|-------|----------------------|----------|
| consump  |          |           |       |       |                      |          |
| wagegovt | 2.50744  | .5957173  | 4.21  | 0.000 | 1.264796             | 3.750085 |
| _cons    | 40.84699 | 3.192183  | 12.80 | 0.000 | 34.18821             | 47.50577 |

If we assume that `wagegovt` is a strictly exogenous variable, we can use the Durbin–Watson test to check for first-order serial correlation in the errors.

```
. estat dwatson
Durbin-Watson d-statistic(2, 22) = .3217998
```

The Durbin–Watson  $d$  statistic, 0.32, is far from the center of its distribution ( $d = 2.0$ ). Given 22 observations and two regressors (including the constant term) in the model, the lower 5% bound is about 0.997, much greater than the computed  $d$  statistic. Assuming that `wagegovt` is strictly exogenous, we can reject the null of no first-order serial correlation. Rejecting the null hypothesis does not necessarily mean an AR process; other forms of misspecification may also lead to a significant test statistic. If we are willing to assume that the errors follow an AR(1) process and that `wagegovt` is strictly exogenous, we could refit the model using `arima` or `prais` and model the error process explicitly; see [\[TS\] arima](#) and [\[TS\] prais](#).

If we are not willing to assume that `wagegovt` is strictly exogenous, we could instead use Durbin’s alternative test or the Breusch–Godfrey to test for first-order serial correlation. Because we have only 22 observations, we will use the `small` option.

```
. estat durbinalt, small
Durbin’s alternative test for autocorrelation
```

| lags( <i>p</i> ) | F      | df        | Prob > F |
|------------------|--------|-----------|----------|
| 1                | 35.035 | ( 1, 19 ) | 0.0000   |

H0: no serial correlation

```
. estat bgodfrey, small
Breusch-Godfrey LM test for autocorrelation
```

| lags( <i>p</i> ) | F      | df        | Prob > F |
|------------------|--------|-----------|----------|
| 1                | 14.264 | ( 1, 19 ) | 0.0013   |

H0: no serial correlation

Both tests strongly reject the null of no first-order serial correlation, so we decide to refit the model with two lags of `consump` included as regressors and then rerun `estat durbinalt` and `estat bgodfrey`. Because the revised model includes lagged values of the dependent variable, the Durbin–Watson test is not applicable.

```
. regress consump wagegovt L.consump L2.consump
```

| Source   | SS         | df | MS         | Number of obs = | 20     |
|----------|------------|----|------------|-----------------|--------|
| Model    | 702.660311 | 3  | 234.220104 | F( 3, 16) =     | 44.01  |
| Residual | 85.1596011 | 16 | 5.32247507 | Prob > F =      | 0.0000 |
| Total    | 787.819912 | 19 | 41.4642059 | R-squared =     | 0.8919 |
|          |            |    |            | Adj R-squared = | 0.8716 |
|          |            |    |            | Root MSE =      | 2.307  |

| consump  | Coef.    | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|----------|----------|-----------|-------|-------|----------------------|----------|
| wagegovt | .6904282 | .3295485  | 2.10  | 0.052 | -.0081835            | 1.38904  |
| consump  |          |           |       |       |                      |          |
| L1.      | 1.420536 | .197024   | 7.21  | 0.000 | 1.002864             | 1.838208 |
| L2.      | -.650888 | .1933351  | -3.37 | 0.004 | -1.06074             | -.241036 |
| _cons    | 9.209073 | 5.006701  | 1.84  | 0.084 | -1.404659            | 19.82281 |

```
. estat durbinalt, small lags(1/2)
```

Durbin’s alternative test for autocorrelation

| lags(p) | F     | df        | Prob > F |
|---------|-------|-----------|----------|
| 1       | 0.080 | ( 1, 15 ) | 0.7805   |
| 2       | 0.260 | ( 2, 14 ) | 0.7750   |

H0: no serial correlation

```
. estat bgodfrey, small lags(1/2)
```

Breusch–Godfrey LM test for autocorrelation

| lags(p) | F     | df        | Prob > F |
|---------|-------|-----------|----------|
| 1       | 0.107 | ( 1, 15 ) | 0.7484   |
| 2       | 0.358 | ( 2, 14 ) | 0.7056   |

H0: no serial correlation

Although `wagegovt` and the constant term are no longer statistically different from zero at the 5% level, the output from `estat durbinalt` and `estat bgodfrey` indicates that including the two lags of `consump` has removed any serial correlation from the errors.

◀

Engle (1982) suggests an LM test for checking for autoregressive conditional heteroskedasticity (ARCH) in the errors. The  $p$ th-order ARCH model can be written as

$$\begin{aligned}\sigma_t^2 &= E(u_t^2 | u_{t-1}, \dots, u_{t-p}) \\ &= \gamma_0 + \gamma_1 u_{t-1}^2 + \dots + \gamma_p u_{t-p}^2\end{aligned}$$

To test the null hypothesis of no autoregressive conditional heteroskedasticity (i.e.,  $\gamma_1 = \dots = \gamma_p = 0$ ), we first fit the OLS model (1), obtain the residuals  $\hat{u}_t$ , and run another OLS regression on the lagged residuals:

$$\hat{u}_t^2 = \gamma_0 + \gamma_1 \hat{u}_{t-1}^2 + \dots + \gamma_p \hat{u}_{t-p}^2 + \epsilon \tag{3}$$

The test statistic is  $NR^2$ , where  $N$  is the number of observations in the sample and  $R^2$  is the  $R^2$  from the regression in (3). Under the null hypothesis, the test statistic follows a  $\chi^2_p$  distribution.

➤ Example 2: estat archlm

We refit the original model that does not include the two lags of `consump` and then use `estat archlm` to see if there is any evidence that the errors are autoregressive conditional heteroskedastic.

```
. regress consump wagegovt
```

| Source   | SS         | df | MS         |
|----------|------------|----|------------|
| Model    | 532.567711 | 1  | 532.567711 |
| Residual | 601.207167 | 20 | 30.0603584 |
| Total    | 1133.77488 | 21 | 53.9892799 |

|  | Number of obs = | 22     |
|--|-----------------|--------|
|  | F( 1, 20) =     | 17.72  |
|  | Prob > F =      | 0.0004 |
|  | R-squared =     | 0.4697 |
|  | Adj R-squared = | 0.4432 |
|  | Root MSE =      | 5.4827 |

| consump  | Coef.    | Std. Err. | t     | P> t  | [95% Conf. Interval] |
|----------|----------|-----------|-------|-------|----------------------|
| wagegovt | 2.50744  | .5957173  | 4.21  | 0.000 | 1.264796 3.750085    |
| _cons    | 40.84699 | 3.192183  | 12.80 | 0.000 | 34.18821 47.50577    |

```
. estat archlm, lags(1 2 3)
```

LM test for autoregressive conditional heteroskedasticity (ARCH)

| lags(p) | chi2  | df | Prob > chi2 |
|---------|-------|----|-------------|
| 1       | 5.543 | 1  | 0.0186      |
| 2       | 9.431 | 2  | 0.0090      |
| 3       | 9.039 | 3  | 0.0288      |

H0: no ARCH effects      vs.    H1: ARCH(p) disturbance

`estat archlm` shows the results for tests of ARCH(1), ARCH(2), and ARCH(3) effects, respectively. At the 5% significance level, all three tests reject the null hypothesis that the errors are not autoregressive conditional heteroskedastic. See [TS] [arch](#) for information on fitting ARCH models.



## Saved results

`estat archlm` saves the following in `r()`:

|                      |                                   |                        |                       |
|----------------------|-----------------------------------|------------------------|-----------------------|
| Scalars              |                                   |                        |                       |
| <code>r(N)</code>    | number of observations            | <code>r(N_gaps)</code> | number of gaps        |
| <code>r(k)</code>    | number of regressors              |                        |                       |
| Macros               |                                   |                        |                       |
| <code>r(lags)</code> | lag order                         |                        |                       |
| Matrices             |                                   |                        |                       |
| <code>r(arch)</code> | test statistic for each lag order | <code>r(p)</code>      | two-sided $p$ -values |
| <code>r(df)</code>   | degrees of freedom                |                        |                       |

`estat bgodfrey` saves the following in `r()`:

|                      |                                               |                        |                       |
|----------------------|-----------------------------------------------|------------------------|-----------------------|
| Scalars              |                                               |                        |                       |
| <code>r(N)</code>    | number of observations                        | <code>r(N_gaps)</code> | number of gaps        |
| <code>r(k)</code>    | number of regressors                          |                        |                       |
| Macros               |                                               |                        |                       |
| <code>r(lags)</code> | lag order                                     |                        |                       |
| Matrices             |                                               |                        |                       |
| <code>r(chi2)</code> | $\chi^2$ statistic for each lag order         | <code>r(p)</code>      | two-sided $p$ -values |
| <code>r(F)</code>    | $F$ statistic for each lag order (small only) | <code>r(df)</code>     | degrees of freedom    |
| <code>r(df_r)</code> | residual degrees of freedom (small only)      |                        |                       |

`estat durbinalt` saves the following in `r()`:

|                      |                                               |                        |                       |
|----------------------|-----------------------------------------------|------------------------|-----------------------|
| Scalars              |                                               |                        |                       |
| <code>r(N)</code>    | number of observations                        | <code>r(N_gaps)</code> | number of gaps        |
| <code>r(k)</code>    | number of regressors                          |                        |                       |
| Macros               |                                               |                        |                       |
| <code>r(lags)</code> | lag order                                     |                        |                       |
| Matrices             |                                               |                        |                       |
| <code>r(chi2)</code> | $\chi^2$ statistic for each lag order         | <code>r(p)</code>      | two-sided $p$ -values |
| <code>r(F)</code>    | $F$ statistic for each lag order (small only) | <code>r(df)</code>     | degrees of freedom    |
| <code>r(df_r)</code> | residual degrees of freedom (small only)      |                        |                       |

`estat dwatson` saves the following in `r()`:

|                   |                        |                        |                         |
|-------------------|------------------------|------------------------|-------------------------|
| Scalars           |                        |                        |                         |
| <code>r(N)</code> | number of observations | <code>r(N_gaps)</code> | number of gaps          |
| <code>r(k)</code> | number of regressors   | <code>r(dw)</code>     | Durbin–Watson statistic |

## Methods and formulas

`estat archlm`, `estat bgodfrey`, `estat durbinalt`, and `estat dwatson` are implemented as ado-files.

Consider the regression

$$y_t = \beta_1 x_{1t} + \cdots + \beta_k x_{kt} + u_t \tag{4}$$

in which some of the covariates are not strictly exogenous. In particular, some of the  $x_{it}$  may be lags of the dependent variable. We are interested in whether the  $u_t$  are serially correlated.

The Durbin–Watson  $d$  statistic reported by `estat dwatson` is

$$d = \frac{\sum_{t=1}^{n-1} (\hat{u}_{t+1} - \hat{u}_t)^2}{\sum_{t=1}^n \hat{u}_t^2}$$

where  $\hat{u}_t$  represents the residual of the  $t$ th observation.

To compute Durbin’s alternative test and the Breusch–Godfrey test against the null hypothesis that there is no  $p$ th order serial correlation, we fit the regression in (4), compute the residuals, and then fit the following auxiliary regression of the residuals  $\hat{u}_t$  on  $p$  lags of  $\hat{u}_t$  and on all the covariates in the original regression in (4):

$$\hat{u}_t = \gamma_1 \hat{u}_{t-1} + \cdots + \gamma_p \hat{u}_{t-p} + \beta_1 x_{1t} + \cdots + \beta_k x_{kt} + \epsilon \quad (5)$$

Durbin’s alternative test is computed by performing a Wald test to determine whether the coefficients of  $\hat{u}_{t-1}, \dots, \hat{u}_{t-p}$  are jointly different from zero. By default, the statistic is assumed to be distributed  $\chi^2(p)$ . When `small` is specified, the statistic is assumed to follow an  $F(p, N - p - k)$  distribution. The reported  $p$ -value is a two-sided  $p$ -value. When `robust` is specified, the Wald test is performed using the Huber/White/sandwich estimator of the variance–covariance matrix, and the test is robust to an unspecified form of heteroskedasticity.

The Breusch–Godfrey test is computed as  $NR^2$ , where  $N$  is the number of observations in the auxiliary regression (5) and  $R^2$  is the  $R^2$  from the same regression (5). Like Durbin’s alternative test, the Breusch–Godfrey test is asymptotically distributed  $\chi^2(p)$ , but specifying `small` causes the  $p$ -value to be computed using an  $F(p, N - p - k)$ .

By default, the initial missing values of the lagged residuals are replaced with zeros, and the auxiliary regression is run over the full sample used in the original regression of (4). Specifying the `nomiss0` option causes these missing values to be treated as missing values, and the observations are dropped from the sample.

Engle’s LM test for ARCH( $p$ ) effects fits an OLS regression of  $\hat{u}_t^2$  on  $\hat{u}_{t-1}^2, \dots, \hat{u}_{t-p}^2$ :

$$\hat{u}_t^2 = \gamma_0 + \gamma_1 \hat{u}_{t-1}^2 + \cdots + \gamma_p \hat{u}_{t-p}^2 + \epsilon$$

The test statistic is  $nR^2$  and is asymptotically distributed  $\chi^2(p)$ .

## Acknowledgment

The original versions of `estat archlm`, `estat bgodfrey`, and `estat durbinalt` were written by Christopher F. Baum, Boston College.

## References

- Baum, C. F. 2006. *An Introduction to Modern Econometrics Using Stata*. College Station, TX: Stata Press.
- Baum, C. F., and V. L. Wiggins. 2000a. sg135: Test for autoregressive conditional heteroskedasticity in regression error distribution. *Stata Technical Bulletin* 55: 13–14. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 143–144. College Station, TX: Stata Press.
- . 2000b. sg136: Tests for serial correlation in regression error distribution. *Stata Technical Bulletin* 55: 14–15. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 145–147. College Station, TX: Stata Press.
- Beran, R. J., and N. I. Fisher. 1998. A conversation with Geoff Watson. *Statistical Science* 13: 75–93.
- Breusch, T. S. 1978. Testing for autocorrelation in dynamic linear models. *Australian Economic Papers* 17: 334–355.
- Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.
- Durbin, J. 1970. Testing for serial correlation in least-squares regressions when some of the regressors are lagged dependent variables. *Econometrica* 38: 410–421.
- Durbin, J., and G. S. Watson. 1950. Testing for serial correlation in least squares regression. I. *Biometrika* 37: 409–428.

- . 1951. Testing for serial correlation in least squares regression. II. *Biometrika* 38: 159–177.
- Engle, R. F. 1982. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica* 50: 987–1007.
- Fisher, N. I., and P. Hall. 1998. Geoffrey Stuart Watson: Tributes and obituary (3 December 1921–3 January 1998). *Australian and New Zealand Journal of Statistics* 40: 257–267.
- Godfrey, L. G. 1978. Testing against general autoregressive and moving average error models when the regressors include lagged dependent variables. *Econometrics* 46: 1293–1301.
- . 1988. *Misspecification Tests in Econometrics: The Lagrange Multiplier Principle and Other Approaches*. Econometric Society Monographs, No. 16. Cambridge: Cambridge University Press.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- Klein, L. R. 1950. *Economic Fluctuations in the United States 1921–1941*. New York: Wiley.
- Phillips, P. C. B. 1988. The *ET* Interview: Professor James Durbin. *Econometric Theory* 4: 125–157.
- Savin, N. E., and K. J. White. 1977. The Durbin–Watson test for serial correlation with extreme sample sizes or many regressors. *Econometrica* 45: 1989–1996.
- Wooldridge, J. M. 2009. *Introductory Econometrics: A Modern Approach*. 4th ed. Cincinnati, OH: South-Western.

James Durbin (1923– ) is a British statistician who was born in Wigan, near Manchester. He studied mathematics at Cambridge and after military service and various research posts joined the London School of Economics in 1950. His many contributions to statistics have centered on serial correlation, time series, sample survey methodology, goodness-of-fit tests, and sample distribution functions, with emphasis on applications in the social sciences.

Geoffrey Stuart Watson (1921–1998) was born in Victoria, Australia, and earned degrees at Melbourne University and North Carolina State University. After a visit to the University of Cambridge, he returned to Australia, working at Melbourne and then the Australian National University. Following periods at Toronto and Johns Hopkins, he settled at Princeton. His best-known contributions to statistics, in a wide-ranging career, include the Durbin–Watson test for serial correlation, the Nadaraya–Watson estimator in nonparametric regression, and methods for directional data.

Leslie G. Godfrey (1946– ) was born in London and earned degrees at the Universities of Exeter and London. He now holds a chair in econometrics at the University of York. His interests center on implementation and interpretation of tests of econometric models, including nonnested models.

Trevor Stanley Breusch (1949– ) was born in Queensland and earned degrees at the University of Queensland and Australian National University (ANU). After a post at the University of Southampton, he returned to work at ANU. His background is in econometric methods and his recent interests include political values and social attitudes, earnings and income, and measurement of underground economic activity.

## Also see

[R] **regress** — Linear regression

[TS] **tsset** — Declare data to be time-series data

[R] **regress postestimation** — Postestimation tools for regress

# Title

#review — Review previous commands

## Syntax

```
#review [#1 [#2]]
```

## Description

The `#review` command displays the last few lines typed at the terminal.

## Remarks

`#review` (pronounced *pound-review*) is a Stata preprocessor command. `#commands` do not generate a return code or generate ordinary Stata errors. The only error message associated with `#commands` is “unrecognized `#command`”.

The `#review` command displays the last few lines typed at the terminal. If no arguments follow `#review`, the last five lines typed at the terminal are displayed. The first argument specifies the number of lines to be reviewed, so `#review 10` displays the last 10 lines typed. The second argument specifies the number of lines to be displayed, so `#review 10 5` displays five lines, starting at the 10th previous line.

Stata reserves a buffer for `#review` lines and stores as many previous lines in the buffer as will fit, rolling out the oldest line to make room for the newest. Requests to `#review` lines no longer stored will be ignored. Only lines typed at the terminal are placed in the `#review` buffer. See [\[U\] 10.5 Editing previous lines in Stata](#).

### ► Example 1

Typing `#review` by itself will show the last five lines you typed at the terminal:

```
. #review
5 use mydata
4 * comments go into the #review buffer, too
3 describe
2 tabulate marriage educ [freq=number]
1 tabulate marriage educ [freq=number], chi2
. _
```

Typing `#review 15 2` shows the 15th and 14th previous lines:

```
. #review 15 2
15 replace x=. if x<200
14 summarize x
. _
```





Syntax

*Perform nonparametric ROC analysis*

```
roctab refvar classvar [if] [in] [weight] [, roctab_options]
```

*Test equality of ROC areas*

```
roccomp refvar classvar [classvars] [if] [in] [weight] [, roccomp_options]
```

*Test equality of ROC area against a standard ROC curve*

```
rocgold refvar goldvar classvar [classvars] [if] [in] [weight] [, rocgold_options]
```

| roctab_options                                                | description                                                                               |
|---------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| Main                                                          |                                                                                           |
| <u>lorenz</u>                                                 | report Gini and Pietra indices                                                            |
| <u>binomial</u>                                               | calculate exact binomial confidence intervals                                             |
| <u>detail</u>                                                 | show details on sensitivity/specificity for each cutpoint                                 |
| <u>table</u>                                                  | display the raw data in a $2 \times k$ contingency table                                  |
| <u>bamber</u>                                                 | calculate standard errors by using the Bamber method                                      |
| <u>hanley</u>                                                 | calculate standard errors by using the Hanley method                                      |
| <u>graph</u>                                                  | graph the ROC curve                                                                       |
| <u>norefln</u>                                                | suppress plotting the 45-degree reference line                                            |
| <u>summary</u>                                                | report the area under the ROC curve                                                       |
| <u>specificity</u>                                            | graph sensitivity versus specificity                                                      |
| <u>level(#)</u>                                               | set confidence level; default is <code>level(95)</code>                                   |
| Plot                                                          |                                                                                           |
| <u>plotopts(plot_options)</u>                                 | affect rendition of the ROC curve                                                         |
| Reference line                                                |                                                                                           |
| <u>rlopts(cline_options)</u>                                  | affect rendition of the reference line                                                    |
| Add plots                                                     |                                                                                           |
| <u>addplot(plot)</u>                                          | add other plots to the generated graph                                                    |
| Y axis, X axis, Titles, Legend, Overall                       |                                                                                           |
| <u>twoway_options</u>                                         | any options other than <code>by()</code> documented in <a href="#">[G] twoway_options</a> |
| fweights are allowed; see <a href="#">[U] 11.1.6 weight</a> . |                                                                                           |

| <i>roccomp_options</i>                                        | description                                                  |
|---------------------------------------------------------------|--------------------------------------------------------------|
| Main                                                          |                                                              |
| <code>by(<i>varname</i>)</code>                               | split into groups by variable                                |
| <code>test(<i>matname</i>)</code>                             | use contrast matrix for comparing ROC areas                  |
| <code>graph</code>                                            | graph the ROC curve                                          |
| <code>norefln</code>                                          | suppress plotting the 45-degree reference line               |
| <code>separate</code>                                         | place each ROC curve on its own graph                        |
| <code>summary</code>                                          | report the area under the ROC curve                          |
| <code>binormal</code>                                         | estimate areas by using binormal distribution assumption     |
| <code>line#opts(<i>cline_options</i>)</code>                  | affect rendition of the #th binormal fit line                |
| <code>level(#)</code>                                         | set confidence level; default is <code>level(95)</code>      |
| Plots                                                         |                                                              |
| <code>plot#opts(<i>plot_options</i>)</code>                   | affect rendition of the #th ROC curve                        |
| Reference line                                                |                                                              |
| <code>rlopts(<i>cline_options</i>)</code>                     | affect rendition of the reference line                       |
| Y axis, X axis, Titles, Legend, Overall                       |                                                              |
| <code>twoway_options</code>                                   | any options documented in <a href="#">[G] twoway_options</a> |
| fweights are allowed; see <a href="#">[U] 11.1.6 weight</a> . |                                                              |

| <i>rocgold_options</i>                                        | description                                                          |
|---------------------------------------------------------------|----------------------------------------------------------------------|
| Main                                                          |                                                                      |
| <code>sidak</code>                                            | adjust the significance probability by using Šidák’s method          |
| <code>test(<i>matname</i>)</code>                             | use contrast matrix for comparing ROC areas                          |
| <code>graph</code>                                            | graph the ROC curve                                                  |
| <code>norefln</code>                                          | suppress plotting the 45-degree reference line                       |
| <code>separate</code>                                         | place each ROC curve on its own graph                                |
| <code>summary</code>                                          | report the area under the ROC curve                                  |
| <code>binormal</code>                                         | estimate areas by using binormal distribution assumption             |
| <code>line#opts(<i>cline_options</i>)</code>                  | affect rendition of the #th binormal fit line                        |
| <code>level(#)</code>                                         | set confidence level; default is <code>level(95)</code>              |
| Plots                                                         |                                                                      |
| <code>plot#opts(<i>plot_options</i>)</code>                   | affect rendition of the #th ROC curve; plot 1 is the “gold standard” |
| Reference line                                                |                                                                      |
| <code>rlopts(<i>cline_options</i>)</code>                     | affect rendition of the reference line                               |
| Y axis, X axis, Titles, Legend, Overall                       |                                                                      |
| <code>twoway_options</code>                                   | any options documented in <a href="#">[G] twoway_options</a>         |
| fweights are allowed; see <a href="#">[U] 11.1.6 weight</a> . |                                                                      |

|                             |                                            |
|-----------------------------|--------------------------------------------|
| <i>plot_options</i>         | description                                |
| <i>marker_options</i>       | change look of markers (color, size, etc.) |
| <i>marker_label_options</i> | add marker labels; change look or position |
| <i>cline_options</i>        | change the look of the line                |

## Menu

### roctab

Statistics > Epidemiology and related > ROC analysis > Nonparametric ROC analysis

### roccomp

Statistics > Epidemiology and related > ROC analysis > Test equality of two or more ROC areas

### rocgold

Statistics > Epidemiology and related > ROC analysis > Test equality of ROC area against gold standard

## Description

The above commands are used to perform receiver operating characteristic (ROC) analyses with rating and discrete classification data.

The two variables *refvar* and *classvar* must be numeric. The reference variable indicates the true state of the observation, such as diseased and nondiseased or normal and abnormal, and must be coded as 0 and 1. The rating or outcome of the diagnostic test or test modality is recorded in *classvar*, which must be at least ordinal, with higher values indicating higher risk.

**roctab** performs nonparametric ROC analyses. By default, **roctab** calculates the area under the ROC curve. Optionally, **roctab** can plot the ROC curve, display the data in tabular form, and produce Lorenz-like plots.

**roccomp** tests the equality of two or more ROC areas obtained from applying two or more test modalities to the same sample or to independent samples. **roccomp** expects the data to be in wide form when comparing areas estimated from the same sample and in long form for areas estimated from independent samples.

**rocgold** independently tests the equality of the ROC area of each of several test modalities, specified by *classvar*, against a “gold standard” ROC curve, *goldvar*. For each comparison, **rocgold** reports the raw and the Bonferroni-adjusted significance probability. Optionally, Šidák’s adjustment for multiple comparisons can be obtained.

See [R] **rocfit** for a command that fits maximum-likelihood ROC models.

## Options for roctab

Main

**lorenz** specifies that Gini and Pietra indices be reported. Optionally, **graph** will plot the Lorenz-like curve.

**binomial** specifies that exact binomial confidence intervals be calculated.

**detail** outputs a table displaying the sensitivity, specificity, the percentage of subjects correctly classified, and two likelihood ratios for each possible cutpoint of *classvar*.

**table** outputs a  $2 \times k$  contingency table displaying the raw data.

**bamber** specifies that the standard error for the area under the ROC curve be calculated using the method suggested by [Bamber \(1975\)](#). Otherwise, standard errors are obtained as suggested by [DeLong, DeLong, and Clarke-Pearson \(1988\)](#).

**hanley** specifies that the standard error for the area under the ROC curve be calculated using the method suggested by [Hanley and McNeil \(1982\)](#). Otherwise, standard errors are obtained as suggested by [DeLong, DeLong, and Clarke-Pearson \(1988\)](#).

**graph** produces graphical output of the ROC curve. If **lorenz** is specified, graphical output of a Lorenz-like curve will be produced.

**norefl** suppresses plotting the 45-degree reference line from the graphical output of the ROC curve.

**summary** reports the area under the ROC curve, its standard error, and its confidence interval. If **lorenz** is specified, Lorenz indices are reported. This option is needed only when also specifying **graph**.

**specificity** produces a graph of sensitivity versus specificity instead of sensitivity versus  $(1 - \text{specificity})$ . **specificity** implies **graph**.

**level(#)** specifies the confidence level, as a percentage, for the confidence intervals. The default is **level(95)** or as set by **set level**; see [\[R\] level](#).

---

#### Plot

**plotopts(plot\_options)** affects the rendition of the plotted ROC curve—the curve’s plotted points connected by lines. The *plot\_options* can affect the size and color of markers, whether and how the markers are labeled, and whether and how the points are connected; see [\[G\] marker\\_options](#), [\[G\] marker\\_label\\_options](#), and [\[G\] cline\\_options](#).

---

#### Reference line

**rlopts(cline\_options)** affects the rendition of the reference line; see [\[G\] cline\\_options](#).

---

#### Add plots

**addplot(plot)** provides a way to add other plots to the generated graph; see [\[G\] addplot\\_option](#).

---

#### Y axis, X axis, Titles, Legend, Overall

*twoway\_options* are any of the options documented in [\[G\] twoway\\_options](#), excluding **by()**. These include options for titling the graph (see [\[G\] title\\_options](#)) and for saving the graph to disk (see [\[G\] saving\\_option](#)).

## Options for roccomp and rocgold

### Main

`by(varname)` (roccomp only) is required when comparing independent ROC areas. The `by()` variable identifies the groups to be compared.

`sidak` (rocgold only) requests that the significance probability be adjusted for the effect of multiple comparisons by using Šidák's method. Bonferroni's adjustment is reported by default.

`test(matname)` specifies the contrast matrix to be used when comparing ROC areas. By default, the null hypothesis that all areas are equal is tested.

`graph` produces graphical output of the ROC curve.

`norefline` suppresses plotting the 45-degree reference line from the graphical output of the ROC curve.

`separate` is meaningful only with roccomp and specifies that each ROC curve be placed on its own graph rather than one curve on top of the other.

`summary` reports the area under the ROC curve, its standard error, and its confidence interval. This option is needed only when also specifying `graph`.

`binormal` specifies that the areas under the ROC curves to be compared should be estimated using the binormal distribution assumption. By default, areas to be compared are computed using the trapezoidal rule.

`line#opts(cline_options)` affects the rendition of the line representing the #th ROC curve drawn using the binormal distribution assumption; see [G] [cline\\_options](#). These lines are drawn only if the `binormal` option is specified.

`level(#)` specifies the confidence level, as a percentage, for the confidence intervals. The default is `level(95)` or as set by `set level`; see [R] [level](#).

### Plots

`plot#opts(plot_options)` affects the rendition of the #th ROC curve—the curve's plotted points connected by lines. The `plot_options` can affect the size and color of markers, whether and how the markers are labeled, and whether and how the points are connected; see [G] [marker\\_options](#), [G] [marker\\_label\\_options](#), and [G] [cline\\_options](#).

For rocgold, `plot1opts()` are applied to the ROC for the gold standard.

### Reference line

`rlopts(cline_options)` affects the rendition of the reference line; see [G] [cline\\_options](#).

### Y axis, X axis, Titles, Legend, Overall

`twoway_options` are any of the options documented in [G] [twoway\\_options](#). These include options for titling the graph (see [G] [title\\_options](#)), options for saving the graph to disk (see [G] [saving\\_option](#)), and the `by()` option (see [G] [by\\_option](#)).

## Remarks

Remarks are presented under the following headings:

- Introduction*
- Nonparametric ROC curves*
- Lorenz-like curves*
- Comparing areas under the ROC curve*
- Correlated data*
- Independent data*
- Comparing areas with a gold standard*

## Introduction

ROC analysis quantifies the accuracy of diagnostic tests or other evaluation modalities used to discriminate between two states or conditions, which are here referred to as normal and abnormal. The discriminatory accuracy of a diagnostic test is measured by its ability to correctly classify known normal and abnormal subjects. The analysis uses the ROC curve, a graph of the sensitivity versus  $1 - \text{specificity}$  of the diagnostic test. The sensitivity is the fraction of positive cases that are correctly classified by the diagnostic test, whereas the specificity is the fraction of negative cases that are correctly classified. Thus the sensitivity is the true-positive rate, and the specificity is the true-negative rate.

The global performance of a diagnostic test is commonly summarized by the area under the ROC curve. This area can be interpreted as the probability that the result of a diagnostic test of a randomly selected abnormal subject will be greater than the result of the same diagnostic test from a randomly selected normal subject. The greater the area under the ROC curve, the better the global performance of the diagnostic test.

Both nonparametric methods and parametric (semiparametric) methods have been suggested for generating the ROC curve and for calculating its area. The following sections present these approaches, and the last section presents tests for comparing areas under ROC curves.

See [Pepe \(2003\)](#) for a discussion of ROC analysis. Pepe has posted Stata datasets and programs used to reproduce results presented in the book (<http://www.stata.com/bookstore/pepe.html>).

## Nonparametric ROC curves

The points on the nonparametric ROC curve are generated using each possible outcome of the diagnostic test as a classification cutpoint and computing the corresponding sensitivity and  $1 - \text{specificity}$ . These points are then connected by straight lines, and the area under the resulting ROC curve is computed using the trapezoidal rule.

### ► Example 1

[Hanley and McNeil \(1982\)](#) presented data from a study in which a reviewer was asked to classify, using a nine-point scale, a random sample of 109 tomographic images from patients with neurological problems. The rating scale was as follows: 1 = definitely normal, 2 = probably normal, 3 = questionable, 4 = probably abnormal, and 5 = definitely abnormal. The true disease status was normal for 58 of the patients and abnormal for the remaining 51 patients.

Here we list 9 of the 109 observations:

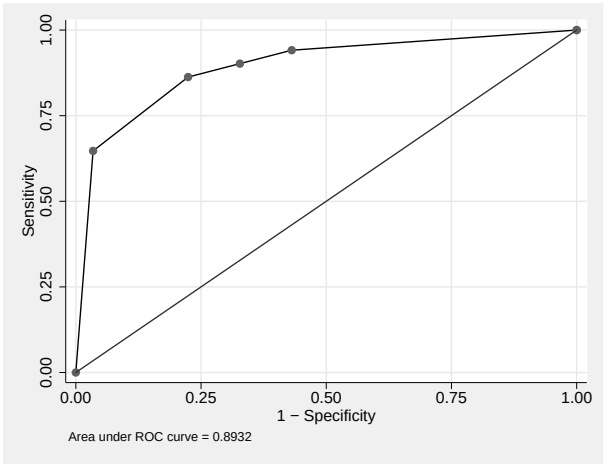
```
. use http://www.stata-press.com/data/r11/hanley
. list disease rating in 1/9
```

|    | disease | rating |
|----|---------|--------|
| 1. | 1       | 5      |
| 2. | 0       | 1      |
| 3. | 1       | 5      |
| 4. | 0       | 4      |
| 5. | 0       | 1      |
| 6. | 0       | 3      |
| 7. | 1       | 5      |
| 8. | 0       | 5      |
| 9. | 0       | 1      |

For each observation, `disease` identifies the true disease status of the subject (0=normal, 1=abnormal), and `rating` contains the classification value assigned by the reviewer.

We can use `roctab` to calculate and plot the nonparametric ROC curve by specifying both the `summary` and `graph` options. By also specifying the `table` option, we obtain a contingency table summarizing our dataset.

```
. roctab disease rating, table graph summary
```



| disease | rating   |           |                                             |         |    | Total |
|---------|----------|-----------|---------------------------------------------|---------|----|-------|
|         | 1        | 2         | 3                                           | 4       | 5  |       |
| 0       | 33       | 6         | 6                                           | 11      | 2  | 58    |
| 1       | 3        | 2         | 2                                           | 11      | 33 | 51    |
| Total   | 36       | 8         | 8                                           | 22      | 35 | 109   |
| Obs     | ROC Area | Std. Err. | -Asymptotic Normal-<br>[95% Conf. Interval] |         |    |       |
| 109     | 0.8932   | 0.0307    | 0.83295                                     | 0.95339 |    |       |

By default, `roctab` reports the area under the curve, its standard error, and its confidence interval. The `graph` option can be used to plot the ROC curve.

The ROC curve is plotted by computing the sensitivity and specificity using each value of the rating variable as a possible cutpoint. A point is plotted on the graph for each of the cutpoints. These plotted points are joined by straight lines to form the ROC curve, and the area under the ROC curve is computed using the trapezoidal rule.

We can tabulate the computed sensitivities and specificities for each of the possible cutpoints by specifying `detail`.

```
. roctab disease rating, detail
```

Detailed report of Sensitivity and Specificity

| Cutpoint | Sensitivity | Specificity | Correctly<br>Classified | LR+     | LR-    |
|----------|-------------|-------------|-------------------------|---------|--------|
| ( >= 1 ) | 100.00%     | 0.00%       | 46.79%                  | 1.0000  |        |
| ( >= 2 ) | 94.12%      | 56.90%      | 74.31%                  | 2.1835  | 0.1034 |
| ( >= 3 ) | 90.20%      | 67.24%      | 77.98%                  | 2.7534  | 0.1458 |
| ( >= 4 ) | 86.27%      | 77.59%      | 81.65%                  | 3.8492  | 0.1769 |
| ( >= 5 ) | 64.71%      | 96.55%      | 81.65%                  | 18.7647 | 0.3655 |
| ( > 5 )  | 0.00%       | 100.00%     | 53.21%                  |         | 1.0000 |

| Obs | ROC<br>Area | Std. Err. | -Asymptotic Normal—<br>[95% Conf. Interval] |         |
|-----|-------------|-----------|---------------------------------------------|---------|
| 109 | 0.8932      | 0.0307    | 0.83295                                     | 0.95339 |

Each cutpoint in the table indicates the ratings used to classify tomographs as being from an abnormal subject. For example, the first cutpoint ( $\geq 1$ ) indicates that all tomographs rated as 1 or greater are classified as coming from abnormal subjects. Because all tomographs have a rating of 1 or greater, all are considered abnormal. Consequently, all abnormal cases are correctly classified (sensitivity = 100%), but none of the normal patients is classified correctly (specificity = 0%). For the second cutpoint ( $\geq 2$ ), tomographs with ratings of 1 are classified as normal, and those with ratings of 2 or greater are classified as abnormal. The resulting sensitivity and specificity are 94.12% and 56.90%, respectively. Using this cutpoint, we correctly classified 74.31% of the 109 tomographs. Similar interpretations can be used on the remaining cutpoints. As mentioned, each cutpoint corresponds to a point on the nonparametric ROC curve. The first cutpoint ( $\geq 1$ ) corresponds to the point at (1,1), and the last cutpoint ( $> 5$ ) corresponds to the point at (0,0).

`detail` also reports two likelihood ratios suggested by [Choi \(1998\)](#): the likelihood ratio for a positive test result (LR+) and the likelihood ratio for a negative test result (LR-). The LR+ is the ratio of the probability of a positive test among the truly positive subjects to the probability of a positive test among the truly negative subjects. The LR- is the ratio of the probability of a negative test among the truly positive subjects to the probability of a negative test among the truly negative subjects. Choi points out that LR+ corresponds to the slope of the line from the origin to the point on the ROC curve determined by the cutpoint. Similarly, LR- corresponds to the slope from the point (1,1) to the point on the ROC curve determined by the cutpoint.

By default, `roctab` calculates the standard error for the area under the curve by using an algorithm suggested by [DeLong, DeLong, and Clarke-Pearson \(1988\)](#) and asymptotic normal confidence intervals. Optionally, standard errors based on methods suggested by [Bamber \(1975\)](#) or [Hanley and McNeil \(1982\)](#) can be computed by specifying `bamber` or `hanley`, respectively, and an exact binomial confidence interval can be obtained by specifying `binomial`.



```
. roctab disease rating, bamber
```

| Obs | ROC Area | Bamber Std. Err. | -Asymptotic Normal—<br>[95% Conf. Interval] |         |
|-----|----------|------------------|---------------------------------------------|---------|
| 109 | 0.8932   | 0.0306           | 0.83317                                     | 0.95317 |

```
. roctab disease rating, hanley binomial
```

| Obs | ROC Area | Hanley Std. Err. | — Binomial Exact —<br>[95% Conf. Interval] |         |
|-----|----------|------------------|--------------------------------------------|---------|
| 109 | 0.8932   | 0.0320           | 0.81559                                    | 0.94180 |

Lorenz-like curves

For applications where it is known that the risk status increases or decreases monotonically with increasing values of the diagnostic test, the ROC curve and associated indices are useful in assessing the overall performance of a diagnostic test. When the risk status does not vary monotonically with increasing values of the diagnostic test, however, the resulting ROC curve can be nonconvex and its indices can be unreliable. For these situations, [Lee \(1999\)](#) proposed an alternative to the ROC analysis based on Lorenz-like curves and the associated Pietra and Gini indices.

[Lee \(1999\)](#) mentions at least three specific situations where results from Lorenz curves are superior to those obtained from ROC curves: 1) a diagnostic test with similar means but very different standard deviations in the abnormal and normal populations, 2) a diagnostic test with bimodal distributions in either the normal or abnormal population, and 3) a diagnostic test distributed symmetrically in the normal population and skewed in the abnormal.

When the risk status increases or decreases monotonically with increasing values of the diagnostic test, the ROC and Lorenz curves yield interchangeable results.

➤ Example 2

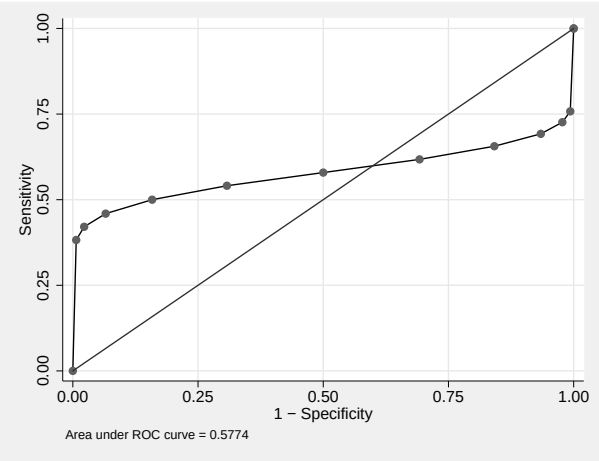
To illustrate the use of the `lorenz` option, we constructed a fictitious dataset that yields results similar to those presented in Table III of [Lee \(1999\)](#). The data assume that a 12-point rating scale was used to classify 442 diseased and 442 healthy subjects. We list a few of the observations.

```
. use http://www.stata-press.com/data/r11/lorenz, clear
. list in 1/7, noobs sep(0)
```

| disease | class | pop |
|---------|-------|-----|
| 0       | 5     | 66  |
| 1       | 11    | 17  |
| 0       | 6     | 85  |
| 0       | 3     | 19  |
| 0       | 10    | 19  |
| 0       | 2     | 7   |
| 1       | 4     | 16  |

The data consist of 24 observations: 12 observations from diseased individuals and 12 from nondiseased individuals. Each observation corresponds to one of the 12 classification values of the rating-scale variable, `class`. The number of subjects represented by each observation is given by the `pop` variable, making this a frequency-weighted dataset. The data were generated assuming a binormal distribution of the latent variable with similar means for the normal and abnormal populations but with the standard deviation for the abnormal population five times greater than that of the normal population.

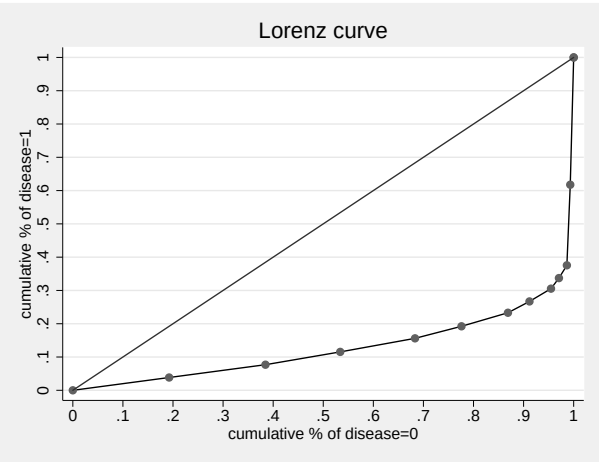
```
. roctab disease class [fweight=pop], graph summary
```



| Obs | ROC Area | Std. Err. | -Asymptotic Normal-<br>[95% Conf. Interval] |         |
|-----|----------|-----------|---------------------------------------------|---------|
| 884 | 0.5774   | 0.0215    | 0.53517                                     | 0.61959 |

The resulting ROC curve is nonconvex or, as termed by Lee, “wiggly”. Lee argues that for this and similar situations, the Lorenz curve and indices are preferred.

```
. roctab disease class [fweight=pop], lorenz graph summary
```



Lorenz curve

|                |        |
|----------------|--------|
| Pietra index = | 0.6493 |
| Gini index =   | 0.7441 |

Like ROC curves, a more bowed Lorenz curve suggests a better diagnostic test. This bowedness is quantified by the Pietra index, which is geometrically equivalent to twice the largest triangle that

can be inscribed in the area between the curve and the diagonal line, and the Gini index, which is equivalent to twice the area between the Lorenz curve and the diagonal. [Lee \(1999\)](#) provides several additional interpretations for the Pietra and Gini indices.

◀

## Comparing areas under the ROC curve

The area under multiple ROC curves can be compared by using `roccomp`. The command syntax is slightly different if the ROC curves are correlated (i.e., different diagnostic tests are applied to the same sample) or independent (i.e., diagnostic tests are applied to different samples).

## Correlated data

### ► Example 3

[Hanley and McNeil \(1983\)](#) presented data from an evaluation of two computer algorithms designed to reconstruct CT images from phantoms. We will call these two algorithms' modalities 1 and 2. A sample of 112 phantoms was selected; 58 phantoms were considered normal, and the remaining 54 were abnormal. Each of the two modalities was applied to each phantom, and the resulting images were rated by a reviewer using a six-point scale: 1 = definitely normal, 2 = probably normal, 3 = possibly normal, 4 = possibly abnormal, 5 = probably abnormal, and 6 = definitely abnormal. Because each modality was applied to the same sample of phantoms, the two sets of outcomes are correlated.

We list the first 7 observations:

```
. use http://www.stata-press.com/data/r11/ct, clear
. list in 1/7, sep(0)
```

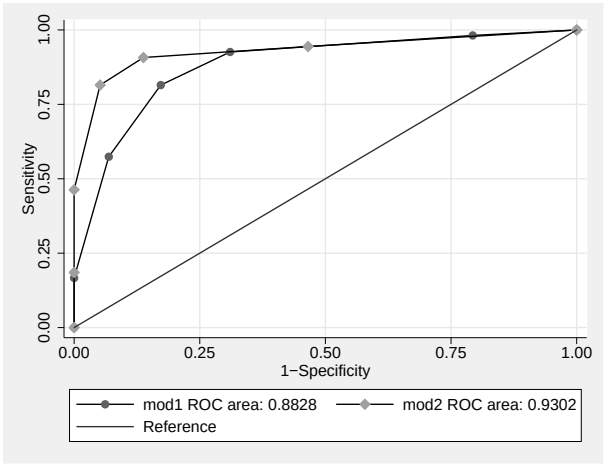
|    | mod1 | mod2 | status |
|----|------|------|--------|
| 1. | 2    | 1    | 0      |
| 2. | 5    | 5    | 1      |
| 3. | 2    | 1    | 0      |
| 4. | 2    | 3    | 0      |
| 5. | 5    | 6    | 1      |
| 6. | 2    | 2    | 0      |
| 7. | 3    | 2    | 0      |

The data are in wide form, which is required when dealing with correlated data. Each observation corresponds to one phantom. The variable `mod1` identifies the rating assigned for the first modality, and `mod2` identifies the rating assigned for the second modality. The true status of the phantoms is given by `status=0` if they are normal and `status=1` if they are abnormal. The observations with at least one missing rating were dropped from the analysis.

We plot the two ROC curves and compare their areas.

*(Continued on next page)*

```
. roccomp status mod1 mod2, graph summary
```



|      | Obs | ROC Area | Std. Err. | -Asymptotic Normal-<br>[95% Conf. Interval] |         |
|------|-----|----------|-----------|---------------------------------------------|---------|
| mod1 | 112 | 0.8828   | 0.0317    | 0.82067                                     | 0.94498 |
| mod2 | 112 | 0.9302   | 0.0256    | 0.88005                                     | 0.98042 |

Ho: area(mod1) = area(mod2)  
chi2(1) = 2.31      Prob>chi2 = 0.1282

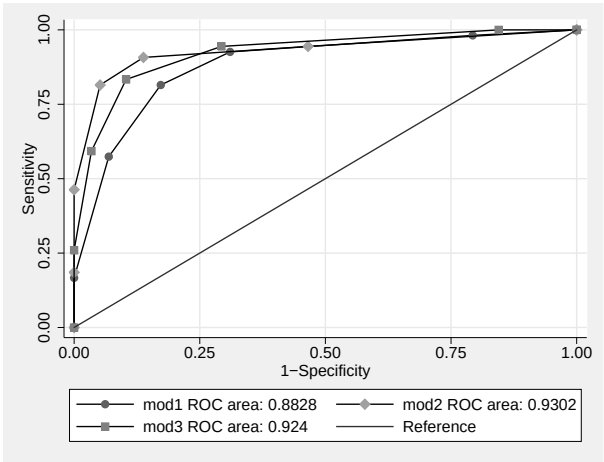
By default, `roccomp`, with the `graph` option specified, plots the ROC curves on the same graph. Optionally the curves can be plotted side by side, each on its own graph, by also specifying `separate`.

For each curve, `roccomp` reports summary statistics and provides a test for the equality of the area under the curves, using an algorithm suggested by [DeLong, DeLong, and Clarke-Pearson \(1988\)](#).

Although the area under the ROC curve for modality 2 is larger than that of modality 1, the chi-squared test yielded a significance probability of 0.1282, suggesting that there is no significant difference between these two areas.

The `roccomp` command can also be used to compare more than two ROC areas. To illustrate this, we modified the previous dataset by including a fictitious third modality.

```
. use http://www.stata-press.com/data/r11/ct2
. roccomp status mod1 mod2 mod3, graph summary
```



|      | Obs | ROC Area | Std. Err. | —Asymptotic Normal—<br>[95% Conf. Interval] |         |
|------|-----|----------|-----------|---------------------------------------------|---------|
| mod1 | 112 | 0.8828   | 0.0317    | 0.82067                                     | 0.94498 |
| mod2 | 112 | 0.9302   | 0.0256    | 0.88005                                     | 0.98042 |
| mod3 | 112 | 0.9240   | 0.0241    | 0.87670                                     | 0.97132 |

Ho: area(mod1) = area(mod2) = area(mod3)  
chi2(2) = 6.54 Prob>chi2 = 0.0381

By default, `roccomp` tests whether the areas under the ROC curves are all equal. Other comparisons can be tested by creating a contrast matrix and specifying `test(matname)`, where *matname* is the name of the contrast matrix.

For example, assume that we are interested in testing whether the area under the ROC for `mod1` is equal to that of `mod3`. To do this, we can first create an appropriate contrast matrix and then specify its name with the `test()` option.

Of course, this is a trivial example because we could have just specified

```
. roccomp status mod1 mod3
```

without including `mod2` to obtain the same test results. However, for illustration, we will continue with this example.

The contrast matrix must have its number of columns equal to the number of *classvars* (i.e., the total number of ROC curves) and a number of rows less than or equal to the number of *classvars*, and the elements of each row must add to zero.

(Continued on next page)

```
. matrix C=(1,0,-1)
. roccomp status mod1 mod2 mod3, test(C)
```

|      | Obs | ROC<br>Area | Std. Err. | -Asymptotic Normal—<br>[95% Conf. Interval] |
|------|-----|-------------|-----------|---------------------------------------------|
| mod1 | 112 | 0.8828      | 0.0317    | 0.82067 0.94498                             |
| mod2 | 112 | 0.9302      | 0.0256    | 0.88005 0.98042                             |
| mod3 | 112 | 0.9240      | 0.0241    | 0.87670 0.97132                             |

```
Ho: Comparison as defined by contrast matrix: C
chi2(1) = 5.25 Prob>chi2 = 0.0220
```

Although all three areas are reported, the comparison is made using the specified contrast matrix.

Perhaps more interesting would be a comparison of the area from mod1 and the average area of mod2 and mod3.

```
. matrix C=(1,-.5,-.5)
. roccomp status mod1 mod2 mod3, test(C)
```

|      | Obs | ROC<br>Area | Std. Err. | -Asymptotic Normal—<br>[95% Conf. Interval] |
|------|-----|-------------|-----------|---------------------------------------------|
| mod1 | 112 | 0.8828      | 0.0317    | 0.82067 0.94498                             |
| mod2 | 112 | 0.9302      | 0.0256    | 0.88005 0.98042                             |
| mod3 | 112 | 0.9240      | 0.0241    | 0.87670 0.97132                             |

```
Ho: Comparison as defined by contrast matrix: C
chi2(1) = 3.43 Prob>chi2 = 0.0642
```

Other contrasts could be made. For example, we could test if mod3 is different from at least one of the other two by first creating the following contrast matrix:

```
. matrix C=(-1, 0, 1 \ 0, -1, 1)
. matrix list C
C[2,3]
 c1 c2 c3
r1 -1 0 1
r2 0 -1 1
```



Independent data

➤ Example 4

In [example 3](#), we noted that because each test modality was applied to the same sample of phantoms, the classification outcomes were correlated. Now assume that we have collected the same data presented by [Hanley and McNeil \(1983\)](#), except that we applied the first test modality to one sample of phantoms and the second test modality to a different sample of phantoms. The resulting measurements are now considered independent.

Here are a few of the observations.

```
. use http://www.stata-press.com/data/r11/ct3
. list in 1/7, sep(0)
```

|    | pop | status | rating | mod |
|----|-----|--------|--------|-----|
| 1. | 12  | 0      | 1      | 1   |
| 2. | 31  | 0      | 1      | 2   |
| 3. | 1   | 1      | 1      | 1   |
| 4. | 3   | 1      | 1      | 2   |
| 5. | 28  | 0      | 2      | 1   |
| 6. | 19  | 0      | 2      | 2   |
| 7. | 3   | 1      | 2      | 1   |

The data are in long form, which is required when dealing with independent data. The data consist of 24 observations: 6 observations corresponding to abnormal phantoms and 6 to normal phantoms evaluated using the first modality, and similarly 6 observations corresponding to abnormal phantoms and 6 to normal phantoms evaluated using the second modality. The number of phantoms corresponding to each observation is given by the `pop` variable. Once again we have frequency-weighted data. The variable `mod` identifies the modality, and `rating` is the assigned classification.

We can better view our data by using the `table` command.

```
. table status rating [fw=pop], by(mod) row col
```

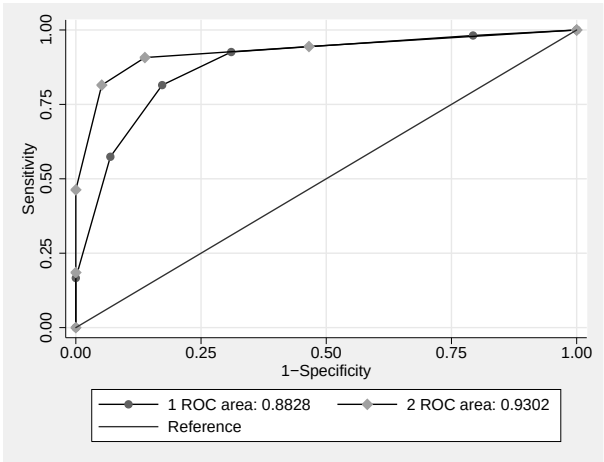
| mod and<br>status |       | rating |    |    |    |    |    | Total |
|-------------------|-------|--------|----|----|----|----|----|-------|
|                   |       | 1      | 2  | 3  | 4  | 5  | 6  |       |
| 1                 | 0     | 12     | 28 | 8  | 6  | 4  |    | 58    |
|                   | 1     | 1      | 3  | 6  | 13 | 22 | 9  | 54    |
|                   | Total | 13     | 31 | 14 | 19 | 26 | 9  | 112   |
| 2                 | 0     | 31     | 19 | 5  | 3  |    |    | 58    |
|                   | 1     | 3      | 2  | 5  | 19 | 15 | 10 | 54    |
|                   | Total | 34     | 21 | 10 | 22 | 15 | 10 | 112   |

The `status` variable indicates the true status of the phantoms: `status` = 0 if they are normal and `status` = 1 if they are abnormal.

We now compare the areas under the two ROC curves.

*(Continued on next page)*

```
. roccomp status rating [fw=pop], by(mod) graph summary
```



| mod | Obs | ROC Area | Std. Err. | -Asymptotic Normal—<br>[95% Conf. Interval] |         |
|-----|-----|----------|-----------|---------------------------------------------|---------|
| 1   | 112 | 0.8828   | 0.0317    | 0.82067                                     | 0.94498 |
| 2   | 112 | 0.9302   | 0.0256    | 0.88005                                     | 0.98042 |

```
Ho: area(1) = area(2)
chi2(1) = 1.35 Prob>chi2 = 0.2447
```



Comparing areas with a gold standard

The area under multiple ROC curves can be compared with a gold standard using `rocgold`. The command syntax is similar to that of `roccomp`. The tests are corrected for the effect of multiple comparisons.

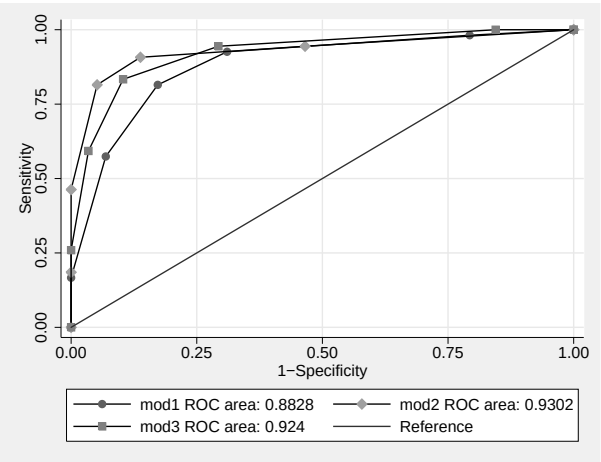
➤ Example 5

We will use the same data (presented by [Hanley and McNeil \[1983\]](#)) as in the `roccomp` examples. Let's assume that the first modality is considered to be the standard against which both the second and third modalities are compared.

We want to plot and compare both the areas of the ROC curves of `mod2` and `mod3` with `mod1`. Because we consider `mod1` to be the gold standard, it is listed first after the reference variable in the `rocgold` command line.



```
. use http://www.stata-press.com/data/r11/ct2
. rocgold status mod1 mod2 mod3, graph summary
```



|                 | ROC<br>Area | Std. Err. | chi2   | df | Pr>chi2 | Bonferroni<br>Pr>chi2 |
|-----------------|-------------|-----------|--------|----|---------|-----------------------|
| mod1 (standard) | 0.8828      | 0.0317    |        |    |         |                       |
| mod2            | 0.9302      | 0.0256    | 2.3146 | 1  | 0.1282  | 0.2563                |
| mod3            | 0.9240      | 0.0241    | 5.2480 | 1  | 0.0220  | 0.0439                |

Equivalently, we could have done this in two steps by using the `roccomp` command.

```
. roccomp status mod1 mod2, graph summary
. roccomp status mod1 mod3, graph summary
```

Saved results

`roctab` saves the following in `r()`:

|                    |                                                    |                        |                          |
|--------------------|----------------------------------------------------|------------------------|--------------------------|
| Scalars            |                                                    |                        |                          |
| <code>r(N)</code>  | number of observations                             | <code>r(area)</code>   | area under the ROC curve |
| <code>r(se)</code> | standard error for the area under the ROC curve    | <code>r(pietra)</code> | Pietra index             |
| <code>r(lb)</code> | lower bound of CI for the area under the ROC curve | <code>r(gini)</code>   | Gini index               |
| <code>r(ub)</code> | upper bound of CI for the area under the ROC curve |                        |                          |

`roccomp` saves the following in `r()`:

|                     |                            |                      |                             |
|---------------------|----------------------------|----------------------|-----------------------------|
| Scalars             |                            |                      |                             |
| <code>r(N_g)</code> | number of groups           | <code>r(df)</code>   | $\chi^2$ degrees of freedom |
| <code>r(p)</code>   | significance probability   | <code>r(chi2)</code> | $\chi^2$                    |
| Matrices            |                            |                      |                             |
| <code>r(V)</code>   | variance-covariance matrix |                      |                             |

rocgold saves the following in `r()`:

|                      |                                    |                       |                                          |
|----------------------|------------------------------------|-----------------------|------------------------------------------|
| Scalars              |                                    |                       |                                          |
| <code>r(N_g)</code>  | number of groups                   |                       |                                          |
| Matrices             |                                    |                       |                                          |
| <code>r(V)</code>    | variance–covariance matrix         | <code>r(p)</code>     | significance-probability vector          |
| <code>r(chi2)</code> | $\chi^2$ vector                    | <code>r(p_adj)</code> | adjusted significance-probability vector |
| <code>r(df)</code>   | $\chi^2$ degrees-of-freedom vector |                       |                                          |

## Methods and formulas

`roctab`, `roccomp`, and `rocgold` are implemented as ado-files.

Assume that we applied a diagnostic test to each of  $N_n$  normal and  $N_a$  abnormal subjects. Further assume that the higher the outcome value of the diagnostic test, the higher the risk of the subject being abnormal. Let  $\hat{\theta}$  be the estimated area under the curve, and let  $X_i, i = 1, 2, \dots, N_a$  and  $Y_j, j = 1, 2, \dots, N_n$  be the values of the diagnostic test for the abnormal and normal subjects, respectively.

Methods and formulas are presented under the following headings:

- Nonparametric ROC*
- Comparing areas under the ROC curve*

## Nonparametric ROC

The points on the nonparametric ROC curve are generated using each possible outcome of the diagnostic test as a classification cutpoint and computing the corresponding sensitivity and 1–specificity. These points are then connected by straight lines, and the area under the resulting ROC curve is computed using the trapezoidal rule.

The default standard error for the area under the ROC curve is computed using the algorithm described by [DeLong, DeLong, and Clarke-Pearson \(1988\)](#). For each abnormal subject,  $i$ , define

$$V_{10}(X_i) = \frac{1}{N_n} \sum_{j=1}^{N_n} \psi(X_i, Y_j)$$

and for each normal subject,  $j$ , define

$$V_{01}(Y_j) = \frac{1}{N_a} \sum_{i=1}^{N_a} \psi(X_i, Y_j)$$

where

$$\psi(X, Y) = \begin{cases} 1 & Y < X \\ \frac{1}{2} & Y = X \\ 0 & Y > X \end{cases}$$

Define

$$S_{10} = \frac{1}{N_a - 1} \sum_{i=1}^{N_a} \{V_{10}(X_i) - \hat{\theta}\}^2$$

and

$$S_{01} = \frac{1}{N_n - 1} \sum_{j=1}^{N_n} \{V_{01}(Y_j) - \hat{\theta}\}^2$$

The variance of the estimated area under the ROC curve is given by

$$\text{var}(\hat{\theta}) = \frac{1}{N_a} S_{10} + \frac{1}{N_n} S_{01}$$

The **hanley** standard error for the area under the ROC curve is computed using the algorithm described by [Hanley and McNeil \(1982\)](#). It requires the calculation of two quantities:  $Q_1$  is  $\Pr(\text{two randomly selected abnormal subjects will both have a higher score than a randomly selected normal subject})$ , and  $Q_2$  is  $\Pr(\text{one randomly selected abnormal subject will have a higher score than any two randomly selected normal subjects})$ . The Hanley and McNeil variance of the estimated area under the ROC curve is

$$\text{var}(\hat{\theta}) = \frac{\hat{\theta}(1 - \hat{\theta}) + (N_a - 1)(Q_1 - \hat{\theta}^2) + (N_n - 1)(Q_2 - \hat{\theta}^2)}{N_a N_n}$$

The **bamber** standard error for the area under the ROC curve is computed using the algorithm described by [Bamber \(1975\)](#). For any two  $Y$  values,  $Y_j$  and  $Y_k$ , and any  $X_i$  value, define

$$b_{yyx} = p(Y_j, Y_k < X_i) + p(X_i < Y_j, Y_k) - 2p(Y_j < X_i < Y_k)$$

and similarly, for any two  $X$  values,  $X_i$  and  $X_l$ , and any  $Y_j$  value, define

$$b_{xxy} = p(X_i, X_l < Y_j) + p(Y_j < X_i, X_l) - 2p(X_i < Y_j < X_l)$$

Bamber's unbiased estimate of the variance for the area under the ROC curve is

$$\text{var}(\hat{\theta}) = \frac{1}{4} (N_a - 1)(N_n - 1) \{p(X \neq Y) + (N_a - 1)b_{xxy} + (N_n - 1)b_{yyx} - 4(N_a + N_n - 1)(\hat{\theta} - 0.5)^2\}$$

Asymptotic confidence intervals are constructed and reported by default, assuming a normal distribution for the area under the ROC curve.

Exact binomial confidence intervals are calculated as described in [\[R\] ci](#), with  $p$  equal to the area under the ROC curve.

## Comparing areas under the ROC curve

Areas under ROC curves are compared using an algorithm suggested by [DeLong, DeLong, and Clarke-Pearson \(1988\)](#). Let  $\hat{\theta} = (\hat{\theta}^1, \hat{\theta}^2, \dots, \hat{\theta}^k)$  be a vector representing the areas under  $k$  ROC curves. For the  $r$ th area, define

$$V_{10}^r(X_i) = \frac{1}{N_n} \sum_{j=1}^{N_n} \psi(X_i^r, Y_j^r)$$

and for each normal subject,  $j$ , define

$$V_{01}^r(Y_j) = \frac{1}{N_a} \sum_{i=1}^{N_a} \psi(X_i^r, Y_j^r)$$

where

$$\psi(X^r, Y^r) = \begin{cases} 1 & Y^r < X^r \\ \frac{1}{2} & Y^r = X^r \\ 0 & Y^r > X^r \end{cases}$$

Define the  $k \times k$  matrix  $\mathbf{S}_{10}$  such that the  $(r, s)$ th element is

$$S_{10}^{r,s} = \frac{1}{N_a - 1} \sum_{i=1}^{N_a} \{V_{10}^r(X_i) - \hat{\theta}^r\} \{V_{10}^s(X_i) - \hat{\theta}^s\}$$

and  $\mathbf{S}_{01}$  such that the  $(r, s)$ th element is

$$S_{01}^{r,s} = \frac{1}{N_n - 1} \sum_{j=1}^{N_n} \{V_{01}^r(Y_j) - \hat{\theta}^r\} \{V_{01}^s(Y_j) - \hat{\theta}^s\}$$

Then the covariance matrix is

$$S = \frac{1}{N_a} S_{10} + \frac{1}{N_n} S_{01}$$

Let  $\mathbf{L}$  be a contrast matrix defining the comparison, so that

$$(\hat{\theta} - \theta)' \mathbf{L}' (\mathbf{L} \mathbf{S} \mathbf{L}')^{-1} \mathbf{L} (\hat{\theta} - \theta)$$

has a chi-squared distribution with degrees of freedom equal to the rank of  $\mathbf{L} \mathbf{S} \mathbf{L}'$ .

## References

Bamber, D. 1975. The area above the ordinal dominance graph and the area below the receiver operating characteristic graph. *Journal of Mathematical Psychology* 12: 387–415.

Choi, B. C. K. 1998. Slopes of a receiver operating characteristic curve and likelihood ratios for a diagnostic test. *American Journal of Epidemiology* 148: 1127–1132.

Cleves, M. A. 1999. [sg120: Receiver operating characteristic \(ROC\) analysis](#). *Stata Technical Bulletin* 52: 19–33. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 212–229. College Station, TX: Stata Press.

———. 2000. [sg120.2: Correction to roccomp command](#). *Stata Technical Bulletin* 54: 26. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, p. 231. College Station, TX: Stata Press.

———. 2002a. [Comparative assessment of three common algorithms for estimating the variance of the area under the nonparametric receiver operating characteristic curve](#). *Stata Journal* 2: 280–289.

———. 2002b. [From the help desk: Comparing areas under receiver operating characteristic curves from two or more probit or logit models](#). *Stata Journal* 2: 301–313.

DeLong, E. R., D. M. DeLong, and D. L. Clarke-Pearson. 1988. Comparing the areas under two or more correlated receiver operating characteristic curves: A nonparametric approach. *Biometrics* 44: 837–845.

Erdreich, L. S., and E. T. Lee. 1981. Use of relative operating characteristic analysis in epidemiology: A method for dealing with subjective judgment. *American Journal of Epidemiology* 114: 649–662.

- Hanley, J. A., and B. J. McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* 143: 29–36.
- . 1983. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology* 148: 839–843.
- Juul, S. 2008. *An Introduction to Stata for Health Researchers*. 2nd ed. College Station, TX: Stata Press.
- Lee, W. C. 1999. Probabilistic analysis of global performances of diagnostic tests: Interpreting the Lorenz curve-based summary measures. *Statistics in Medicine* 18: 455–471.
- Ma, G., and W. J. Hall. 1993. Confidence bands for the receiver operating characteristic curves. *Medical Decision Making* 13: 191–197.
- Pepe, M. S. 2003. *The Statistical Evaluation of Medical Tests for Classification and Prediction*. New York: Oxford University Press.
- Reichenheim, M. E., and A. Ponce de Leon. 2002. Estimation of sensitivity and specificity arising from validity studies with incomplete design. *Stata Journal* 2: 267–279.
- Seed, P. T., and A. Tobías. 2001. *sbe36.1: Summary statistics for diagnostic tests*. *Stata Technical Bulletin* 59: 25–27. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 90–93. College Station, TX: Stata Press.
- Tobías, A. 2000. *sbe36: Summary statistics report for diagnostic tests*. *Stata Technical Bulletin* 56: 16–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 87–90. College Station, TX: Stata Press.
- Working, H., and H. Hotelling. 1929. Application of the theory of error to the interpretation of trends. *Journal of the American Statistical Association* 24 (Suppl.): 73–85.

## Also see

[R] [logistic](#) — Logistic regression, reporting odds ratios

[R] [rocfitt](#) — Fit ROC models

## Syntax

rocfitt *refvar* *classvar* [*if*] [*in*] [*weight*] [, *rocfitt\_options*]

| <i>rocfitt_options</i>                                                                          | description                                                               |
|-------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------|
| Model                                                                                           |                                                                           |
| <u>c</u> ontinuous( <i>#</i> )                                                                  | divide <i>classvar</i> into <i>#</i> groups of approximately equal length |
| <u>g</u> enerate( <i>newvar</i> )                                                               | create <i>newvar</i> containing classification groups                     |
| SE                                                                                              |                                                                           |
| <u>v</u> ce( <i>vcetype</i> )                                                                   | <i>vcetype</i> may be oim or opg                                          |
| Reporting                                                                                       |                                                                           |
| <u>l</u> evel( <i>#</i> )                                                                       | set confidence level; default is level(95)                                |
| Maximization                                                                                    |                                                                           |
| <i>maximize_options</i>                                                                         | control the maximization process; seldom used                             |
| fweights are allowed; see [U] 11.1.6 weight.                                                    |                                                                           |
| See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands. |                                                                           |

## Menu

Statistics > Epidemiology and related > ROC analysis > ROC models assuming a binormal distribution

## Description

rocfitt fits maximum-likelihood ROC models assuming a binormal distribution of the latent variable.

The two variables *refvar* and *classvar* must be numeric. The reference variable indicates the true state of the observation, such as diseased and nondiseased or normal and abnormal, and must be coded as 0 and 1. The rating or outcome of the diagnostic test or test modality is recorded in *classvar*, which must be at least ordinal, with higher values indicating higher risk.

See [R] [roc](#) for other commands designed to perform receiver operating characteristic (ROC) analyses with rating and discrete classification data.

## Options

Model

continuous(*#*) specifies that the continuous *classvar* be divided into *#* groups of approximately equal length. This option is required when *classvar* takes on more than 20 distinct values.

`continuous(.)` may be specified to indicate that *classvar* be used as it is, even though it could have more than 20 distinct values.

`generate(newvar)` specifies the new variable that is to contain the values indicating the groups produced by `continuous(#)`. `generate()` may be specified only with `continuous()`.

#### SE

`vce(vctype)` specifies the type of standard error reported. *vctype* may be either `oim` or `opg`; see [R] [vce\\_option](#).

#### Reporting

`level(#)`; see [R] [estimation options](#).

#### Maximization

*maximize\_options*: [difficult](#), [technique\(algorithm\\_spec\)](#), [iterate\(#\)](#), [\[no\]log](#), [trace](#), [gradient](#), [showstep](#), [hessian](#), [showtolerance](#), [tolerance\(#\)](#), [ltolerance\(#\)](#), [nrtolerance\(#\)](#), [nonrtolerance](#), [from\(init\\_specs\)](#); see [R] [maximize](#). These options are seldom used.

Setting the optimization type to `technique(bhhh)` resets the default *vctype* to `vce(opg)`.

## Remarks

Dorfman and Alf Jr. (1969) developed a generalized approach for obtaining maximum likelihood estimates of the parameters for a smooth fitting ROC curve. The most commonly used method for ordinal data, and the one implemented here, is based upon the binormal model; see [Pepe \(2003\)](#), [Pepe, Longton, and Janes \(2009\)](#), and [Janes, Longton, and Pepe \(2009\)](#) for methods of ROC analysis for continuous data, including methods for adjusting for covariates.

The model assumes the existence of an unobserved, continuous, latent variable that is normally distributed (perhaps after a monotonic transformation) in both the normal and abnormal populations with means  $\mu_n$  and  $\mu_a$  and variances  $\sigma_n^2$  and  $\sigma_a^2$ , respectively. The model further assumes that the  $K$  categories of the rating variable result from partitioning the unobserved latent variable by  $K - 1$  fixed boundaries. The method fits a straight line to the empirical ROC points plotted using normal probability scales on both axes. Maximum likelihood estimates of the line's slope and intercept and the  $K - 1$  boundaries are obtained simultaneously. See [Methods and formulas](#) for details.

The intercept from the fitted line is a measurement of  $(\mu_a - \mu_n)/\sigma_a$ , and the slope measures  $\sigma_n/\sigma_a$ .

Thus the intercept is the standardized difference between the two latent population means, and the slope is the ratio of the two standard deviations. The null hypothesis that there is no difference between the two population means is evaluated by testing that the intercept = 0, and the null hypothesis that the variances in the two populations are equal is evaluated by testing that the slope = 1.

## ► Example 1

We use Hanley and McNeil's (1982) dataset, described in [example 1](#) of [R] [roc](#), to fit a smooth ROC curve assuming a binormal model.

```
. use http://www.stata-press.com/data/r11/hanley
. rocfit disease rating
Fitting binormal model:
Iteration 0: log likelihood = -123.68069
Iteration 1: log likelihood = -123.64867
Iteration 2: log likelihood = -123.64855
Iteration 3: log likelihood = -123.64855
Binormal model of disease on rating Number of obs = 109
Goodness-of-fit chi2(2) = 0.21
Prob > chi2 = 0.9006
Log likelihood = -123.64855
```

|           | Coef.    | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|-----------|----------|-----------|-------|-------|----------------------|----------|
| intercept | 1.656782 | 0.310456  | 5.34  | 0.000 | 1.048300             | 2.265265 |
| slope (*) | 0.713002 | 0.215882  | -1.33 | 0.092 | 0.289881             | 1.136123 |
| /cut1     | 0.169768 | 0.165307  | 1.03  | 0.152 | -0.154227            | 0.493764 |
| /cut2     | 0.463215 | 0.167235  | 2.77  | 0.003 | 0.135441             | 0.790990 |
| /cut3     | 0.766860 | 0.174808  | 4.39  | 0.000 | 0.424243             | 1.109477 |
| /cut4     | 1.797938 | 0.299581  | 6.00  | 0.000 | 1.210770             | 2.385106 |

| Index    | Indices from binormal fit |           |                      |  |  |
|----------|---------------------------|-----------|----------------------|--|--|
|          | Estimate                  | Std. Err. | [95% Conf. Interval] |  |  |
| ROC area | 0.911331                  | 0.029506  | 0.853501   0.969161  |  |  |
| delta(m) | 2.323671                  | 0.502370  | 1.339044   3.308298  |  |  |
| d(e)     | 1.934361                  | 0.257187  | 1.430284   2.438438  |  |  |
| d(a)     | 1.907771                  | 0.259822  | 1.398530   2.417012  |  |  |

(\*) z test for slope==1

rocfit outputs the MLE for the intercept and slope of the fitted regression line along with, here, four boundaries (because there are five ratings) labeled /cut1 through /cut4. Also rocfit computes and reports four indices based on the fitted ROC curve: the area under the curve (labeled ROC area),  $\delta(m)$  (labeled delta(m)),  $d_e$  (labeled d(e)), and  $d_a$  (labeled d(a)). More information about these indices can be found in *Methods and formulas* and in *Erdreich and Lee (1981)*.





## Saved results

rocfit saves the following in `e()`:

### Scalars

|                            |                                                   |
|----------------------------|---------------------------------------------------|
| <code>e(N)</code>          | number of observations                            |
| <code>e(k)</code>          | number of parameters                              |
| <code>e(k_eq)</code>       | number of equations                               |
| <code>e(k_eq_model)</code> | number of equations in model Wald test            |
| <code>e(k_dv)</code>       | number of dependent variables                     |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints    |
| <code>e(df_m)</code>       | model degrees of freedom                          |
| <code>e(ll)</code>         | log likelihood                                    |
| <code>e(chi2_gf)</code>    | goodness-of-fit $\chi^2$                          |
| <code>e(df_gf)</code>      | goodness-of-fit degrees of freedom                |
| <code>e(p_gf)</code>       | $\chi^2$ goodness-of-fit significance probability |
| <code>e(area)</code>       | area under the ROC curve                          |
| <code>e(se_area)</code>    | standard error for the area under the ROC curve   |
| <code>e(deltam)</code>     | <code>delta(m)</code>                             |
| <code>e(se_delm)</code>    | standard area for <code>delta(m)</code>           |
| <code>e(de)</code>         | <code>d(e)</code> index                           |
| <code>e(se_de)</code>      | standard error for <code>d(e)</code> index        |
| <code>e(da)</code>         | <code>d(a)</code> index                           |
| <code>e(se_da)</code>      | standard error for <code>d(a)</code> index        |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                         |
| <code>e(ic)</code>         | number of iterations                              |
| <code>e(rc)</code>         | return code                                       |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                       |

### Macros

|                                 |                                                                          |
|---------------------------------|--------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>rocfit</code>                                                      |
| <code>e(cmdline)</code>         | command as typed                                                         |
| <code>e(depvar)</code>          | <i>refvar</i> and <i>classvar</i>                                        |
| <code>e(wtype)</code>           | weight type                                                              |
| <code>e(wexp)</code>            | weight expression                                                        |
| <code>e(title)</code>           | title in estimation output                                               |
| <code>e(chi2type)</code>        | GOF; type of model $\chi^2$ test                                         |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                           |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                            |
| <code>e(opt)</code>             | type of optimization                                                     |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>       | type of ml method                                                        |
| <code>e(user)</code>            | name of likelihood-evaluator program                                     |
| <code>e(technique)</code>       | maximization technique                                                   |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular              |
| <code>e(crittype)</code>        | optimization criterion                                                   |
| <code>e(properties)</code>      | b V                                                                      |

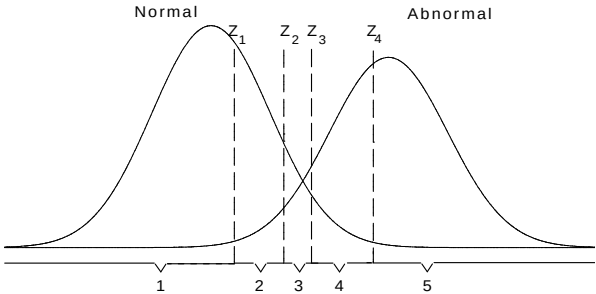
|             |                                              |
|-------------|----------------------------------------------|
| Matrices    |                                              |
| e(b)        | coefficient vector                           |
| e(ilog)     | iteration log (up to 20 iterations)          |
| e(gradient) | gradient vector                              |
| e(V)        | variance–covariance matrix of the estimators |
| e(ml_h)     | derivative tolerance, (abs(b)+1e-3)*1e-3     |
| e(ml_scale) | derivative scale factor                      |
| Functions   |                                              |
| e(sample)   | marks estimation sample                      |

## Methods and formulas

rocfits is implemented as an ado-file.

Dorfman and Alf Jr. (1969) developed a general procedure for obtaining maximum likelihood estimates of the parameters of a smooth-fitting ROC curve. The most common method, and the one implemented in Stata, is based upon the binormal model.

The model assumes that there is an unobserved continuous latent variable that is normally distributed in both the normal and abnormal populations. The idea is better explained with the following illustration:



The latent variable is assumed to be normally distributed for both the normal and abnormal subjects, perhaps after a monotonic transformation, with means  $\mu_n$  and  $\mu_a$  and variances  $\sigma_n^2$  and  $\sigma_a^2$ , respectively.

This latent variable is assumed to be partitioned into the  $k$  categories of the rating variable by  $k - 1$  fixed boundaries. In the above figure, the  $k = 5$  categories of the rating variable identified on the bottom result from the partition of the four boundaries  $Z_1$  through  $Z_4$ .

Let  $R_j$  for  $j = 1, 2, \dots, k$  indicate the categories of the rating variable, let  $i = 1$  if the subject belongs to the normal group, and let  $i = 2$  if the subject belongs to the abnormal group.

Then

$$p(R_j|i = 1) = F(Z_j) - F(Z_{j-1})$$

where  $Z_k = (x_k - \mu_n)/\sigma_n$ ,  $F$  is the cumulative normal distribution,  $F(Z_0) = 0$ , and  $F(Z_k) = 1$ . Also,

$$p(R_j|i=2) = F(bZ_j - a) - F(bZ_{j-1} - a)$$

where  $b = \sigma_n/\sigma_a$  and  $a = (\mu_a - \mu_n)/\sigma_a$ .

The parameters  $a$ ,  $b$  and the  $k-1$  fixed boundaries  $Z_j$  are simultaneously estimated by maximizing the log-likelihood function

$$\log L = \sum_{i=1}^2 \sum_{j=1}^k r_{ij} \log \{p(R_j|i)\}$$

where  $r_{ij}$  is the number of  $R_j$ s in group  $i$ .

The area under the fitted ROC curve is computed as

$$\Phi\left(\frac{a}{\sqrt{1+b^2}}\right)$$

where  $\Phi$  is the standard normal cumulative distribution function.

Point estimates for the ROC curve indices are as follows:

$$\delta(m) = \frac{a}{b} \quad d_e = \frac{2a}{b+1} \quad d_a = \frac{a\sqrt{2}}{\sqrt{1+b^2}}$$

Variances for these indices are computed using the delta method.

The  $\delta(m)$  estimates  $(\mu_a - \mu_n)/\sigma_n$ ,  $d_e$  estimates  $2(\mu_a - \mu_n)/(\sigma_a - \sigma_n)$ , and  $d_a$  estimates  $\sqrt{2}(\mu_a - \mu_n)/(\sigma_a^2 - \sigma_n^2)^{1/2}$ .

Simultaneous confidence bands for the entire curve are obtained, as suggested by [Ma and Hall \(1993\)](#), by first obtaining Working–Hotelling (1929) confidence bands for the fitted straight line in normal probability coordinates and then transforming them back to ROC coordinates.

## References

- Bamber, D. 1975. The area above the ordinal dominance graph and the area below the receiver operating characteristic graph. *Journal of Mathematical Psychology* 12: 387–415.
- Choi, B. C. K. 1998. Slopes of a receiver operating characteristic curve and likelihood ratios for a diagnostic test. *American Journal of Epidemiology* 148: 1127–1132.
- Cleves, M. A. 1999. [sg120: Receiver operating characteristic \(ROC\) analysis](#). *Stata Technical Bulletin* 52: 19–33. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 212–229. College Station, TX: Stata Press.
- . 2000. [sg120.1: Two new options added to rocfits command](#). *Stata Technical Bulletin* 53: 18–19. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 230–231. College Station, TX: Stata Press.
- Dorfman, D. D., and E. Alf Jr. 1969. Maximum-likelihood estimation of parameters of signal-detection theory and determination of confidence intervals—rating-method data. *Journal of Mathematical Psychology* 6: 487–496.
- Erdreich, L. S., and E. T. Lee. 1981. Use of relative operating characteristic analysis in epidemiology: A method for dealing with subjective judgment. *American Journal of Epidemiology* 114: 649–662.
- Hanley, J. A., and B. J. McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* 143: 29–36.
- Janes, H., G. Longton, and M. S. Pepe. 2009. [Accommodating covariates in receiver operating characteristic analysis](#). *Stata Journal* 9: 17–39.
- Ma, G., and W. J. Hall. 1993. Confidence bands for the receiver operating characteristic curves. *Medical Decision Making* 13: 191–197.

- Pepe, M. S. 2003. *The Statistical Evaluation of Medical Tests for Classification and Prediction*. New York: Oxford University Press.
- Pepe, M. S., G. Longton, and H. Janes. 2009. Estimation and comparison of receiver operating characteristic curves. *Stata Journal* 9: 1–16.
- Working, H., and H. Hotelling. 1929. Application of the theory of error to the interpretation of trends. *Journal of the American Statistical Association* 24 (Suppl.): 73–85.

## Also see

- [R] **rocfits postestimation** — Postestimation tools for rocfits
- [R] **roc** — Receiver operating characteristic (ROC) analysis
- [U] **20 Estimation and postestimation commands**

Description

The following command is of special interest after `rocfit`:

| command              | description                                                 |
|----------------------|-------------------------------------------------------------|
| <code>rocplot</code> | plot the fitted ROC curve and simultaneous confidence bands |

For information about `rocplot`, see below.

The following standard postestimation commands are also available:

| command                | description                                                                                      |
|------------------------|--------------------------------------------------------------------------------------------------|
| <code>estat</code>     | AIC, BIC, VCE, and estimation sample summary                                                     |
| <code>estimates</code> | cataloging estimation results                                                                    |
| * <code>lincom</code>  | point estimates, standard errors, testing, and inference for linear combinations of coefficients |
| * <code>test</code>    | Wald tests of simple and composite linear hypotheses                                             |

\*See *Using lincom and test* below.

See the corresponding entries in the *Base Reference Manual* for details.

Special-interest postestimation command

`rocplot` plots the fitted ROC curve and simultaneous confidence bands.

# Syntax for rocplot

| rocplot [ , rocplot_options ]           |                                                                              |
|-----------------------------------------|------------------------------------------------------------------------------|
| rocplot_options                         | description                                                                  |
| Main                                    |                                                                              |
| <u>confband</u>                         | display confidence bands                                                     |
| <u>norefl</u> ine                       | suppress plotting the reference line                                         |
| <u>level</u> (#)                        | set confidence level; default is level(95)                                   |
| Plot                                    |                                                                              |
| <u>plot</u> opts(plot_options)          | affect rendition of the ROC points                                           |
| Fit line                                |                                                                              |
| <u>line</u> opts(ccline_options)        | affect rendition of the fitted ROC line                                      |
| CI plot                                 |                                                                              |
| <u>ci</u> opts(area_options)            | affect rendition of the confidence bands                                     |
| Reference line                          |                                                                              |
| <u>rlo</u> pts(ccline_options)          | affect rendition of the reference line                                       |
| Add plots                               |                                                                              |
| <u>add</u> plot(plot)                   | add other plots to the generated graph                                       |
| Y axis, X axis, Titles, Legend, Overall |                                                                              |
| <u>twoway</u> _options                  | any options other than by() documented in [G] <a href="#">twoway_options</a> |
| plot_options                            |                                                                              |
| description                             |                                                                              |
| <u>marker</u> _options                  | change look of markers (color, size, etc.)                                   |
| <u>marker_label</u> _options            | add marker labels; change look or position                                   |
| <u>ccline</u> _options                  | change the look of the line                                                  |

## Menu

Statistics > Epidemiology and related > ROC analysis > Parametric ROC curve after rocfit

## Options for rocplot

Main

**confband** specifies that simultaneous confidence bands be plotted around the ROC curve.

**norefl**ine suppresses plotting the 45-degree reference line from the graphical output of the ROC curve.

**level**(#) specifies the confidence level, as a percentage, for the confidence bands. The default is level(95) or as set by set level; see [R] [level](#).

## Plot

`plotopts(plot_options)` affects the rendition of the plotted ROC points, including the size and color of markers, whether and how the markers are labeled, and whether and how the points are connected. For the full list of available *plot\_options*, see [G] [marker\\_options](#), [G] [marker\\_label\\_options](#), and [G] [cline\\_options](#).

## Fit line

`lineopts(cline_options)` affects the rendition of the fitted ROC line; see [G] [cline\\_options](#).

## CI plot

`ciopts(area_options)` affects the rendition of the confidence bands; see [G] [area\\_options](#).

## Reference line

`rlopts(ccline_options)` affects the rendition of the reference line; see [G] [cline\\_options](#).

## Add plots

`addplot(plot)` provides a way to add other plots to the generated graph. See [G] [addplot\\_option](#).

## Y axis, X axis, Titles, Legend, Overall

*twoway\_options* are any of the options documented in [G] [twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

## Remarks

Remarks are presented under the following headings:

[Using lincom and test](#)

[Using rocplot](#)

## Using lincom and test

`intercept`, `slope`, and `cut#`, shown in [example 1](#) of [R] [rocfits](#), are equation names and not variable names, so they need to be referenced as described in [Special syntaxes after multiple-equation estimation](#) of [R] [test](#). For example, instead of typing

```
. test intercept
intercept not found
r(111);
```

you should type

```
. test [intercept]_cons
(1) [intercept]_cons = 0
 chi2(1) = 28.48
Prob > chi2 = 0.0000
```

## Using rocfits

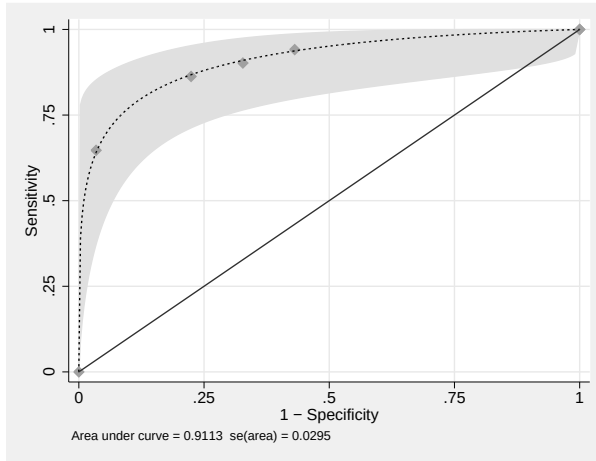
### ► Example 1

In [example 1](#) of [\[R\] rocfits](#), we fit a ROC curve by typing `rocfits disease rating`.

In the output table for our model, we are testing whether the variances of the two latent populations are equal by testing that the slope = 1.

We plot the fitted ROC curve.

```
. rocfits, confband
```



◀

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[\[R\] rocfits](#) — Fit ROC models

[\[U\] 20 Estimation and postestimation commands](#)



Syntax

rologit *depvar* [*indepvars*] [*if*] [*in*] [*weight*] , group(*varname*) [*options*]

| <i>options</i>                    | description                                                                                                                                              |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                             |                                                                                                                                                          |
| * <u>group</u> ( <i>varname</i> ) | identifier variable that links the alternatives                                                                                                          |
| <u>offset</u> ( <i>varname</i> )  | include <i>varname</i> in model with coefficient constrained to 1                                                                                        |
| <u>incomplete</u> (#)             | use # to code unranked alternatives; default is incomplete(0)                                                                                            |
| <u>reverse</u>                    | reverse the preference order                                                                                                                             |
| <u>notestrhs</u>                  | keep right-hand-side variables that do not vary within group                                                                                             |
| <u>ties</u> ( <i>spec</i> )       | method to handle ties: <code>exactm</code> , <code>breslow</code> , <code>efron</code> , or <code>none</code>                                            |
| SE/Robust                         |                                                                                                                                                          |
| <u>vce</u> ( <i>vcetype</i> )     | <i>vcetype</i> may be <code>oim</code> , <code>robust</code> , <code>cluster</code> <i>clustvar</i> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                         |                                                                                                                                                          |
| <u>level</u> (#)                  | set confidence level; default is level(95)                                                                                                               |
| <i>display_options</i>            | control spacing and display of omitted variables and base and empty cells                                                                                |
| Maximization                      |                                                                                                                                                          |
| <i>maximize_options</i>           | control the maximization process; seldom used                                                                                                            |
| <sup>†</sup> <u>coeflegend</u>    | display coefficients' legend instead of coefficient table                                                                                                |

\*group(*varname*) is required.

<sup>†</sup>coeflegend does not appear in the dialog box.

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

`bootstrap`, `by`, `jackknife`, `rolling`, and `statsby` are allowed; see [U] 11.1.10 Prefix commands.

Weights are not allowed with the `bootstrap` prefix.

`fweights`, `iweights`, and `pweights` are allowed, except with `ties(efron)`; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Ordinal outcomes > Rank-ordered logistic regression

Description

`rologit` fits the rank-ordered logistic regression model by maximum likelihood (Beggs, Cardell, and Hausman 1981). This model is also known as the Plackett–Luce model (Marden 1995), as the exploded logit model (Punj and Staelin 1978), and as the choice-based method of conjoint analysis (Hair et al. 2010).

`rologit` expects the data to be in long form, similar to `clogit`, in which each of the ranked alternatives forms an observation; all observations related to an individual are linked together by the variable that you specify in the `group()` option. The distinction from `clogit` is that *depvar* in `rologit` records the rankings of the alternatives, whereas for `clogit`, *depvar* marks only the best alternative by a value not equal to zero. `rologit` interprets equal scores of *depvar* as ties. The ranking information may be incomplete “at the bottom” (least preferred alternatives). That is, unranked alternatives may be coded as 0 or as a common value that may be specified with the `incomplete()` option.

If your data record only the unique best alternative, `rologit` fits the same model as `clogit`.

## Options

---

### Model

`group(varname)` is required, and it specifies the identifier variable (numeric or string) that links the alternatives for an individual, which have been compared and rank ordered with respect to one another.

`offset(varname)`; see [R] [estimation options](#).

`incomplete(#)` specifies the numeric value used to code alternatives that are not ranked. It is assumed that unranked alternatives are less preferred than the ranked alternatives (i.e., the data record the ranking of the most preferred alternatives). It is not assumed that subjects are indifferent between the unranked alternatives. `#` defaults to 0.

`reverse` specifies that in the preference order, a higher number means a less attractive alternative. The default is that higher values indicate more attractive alternatives. The rank-ordered logit model is not symmetric in the sense that reversing the ordering simply leads to a change in the signs of the coefficients.

`notestrhs` suppresses the test that the independent variables vary within (at least some of) the groups. Effects of variables that are always constant are not identified. For instance, a rater’s gender cannot directly affect his or her rankings; it could affect the rankings only via an interaction with a variable that does vary over alternatives.

`ties(spec)` specifies the method for handling ties (indifference between alternatives) (see [ST] [stcox](#) for details):

|                      |                                                                      |
|----------------------|----------------------------------------------------------------------|
| <code>exactm</code>  | exact marginal likelihood (default)                                  |
| <code>breslow</code> | Breslow’s approximation (default if <code>pweights</code> specified) |
| <code>efron</code>   | Efron’s approximation (default if robust VCE)                        |
| <code>none</code>    | no ties allowed                                                      |

---

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

If `ties(exactm)` is specified, *vcetype* may be only `oim`, `bootstrap`, or `jackknife`.

---

### Reporting

`level(#)`; see [R] [estimation options](#).

*display\_options*: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

#### Maximization

*maximize\_options*: `iterate(#)`, `trace`, `[no]log`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`; see [R] [maximize](#). These options are seldom used.

The following option is available with `rologit` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

The rank-ordered logit model can be applied to analyze how decision makers combine attributes of alternatives into overall evaluations of the attractiveness of these alternatives. The model generalizes a version of McFadden's choice model without alternative-specific covariates, as fit by the `clogit` command. It uses richer information about the comparison of alternatives, namely, how decision-makers rank the alternatives rather than just specifying the alternative that they like best.

Remarks are presented under the following headings:

*Examples*  
*Comparing respondents*  
*Incomplete rankings and ties*  
*Clustered choice data*  
*Comparison of rologit and clogit*  
*On reversals of rankings*

## Examples

A popular way to study employer preferences for characteristics of employees is the quasi-experimental “vignette method”. As an example, we consider the research by de Wolf on the labor market position of social science graduates ([de Wolf 2000](#)). This study addresses how the educational portfolio (e.g., general skills versus specific knowledge) affects short-term and long-term labor-market opportunities. De Wolf asked 22 human resource managers (the respondents) to rank order the six most suitable candidates of 20 fictitious applicants and to rank order these six candidates for three jobs, namely, 1) researcher, 2) management trainee, and 3) policy adviser. Applicants were described by 10 attributes, including their age, gender, details of their portfolio, and work experience. In this example, we analyze a subset of the data. Also, to simplify the output, we drop, at random, 10 nonselected applicants per case. The resulting dataset includes 29 cases, consisting of 10 applicants each. The data are in long form: observations correspond to alternatives (the applications), and alternatives that figured in one decision task are identified by the variable `caseid`. We list the observations for `caseid==7`, in which the respondent considered applicants for a social-science research position.

(Continued on next page)

```
. use http://www.stata-press.com/data/r11/evignet
(Vignet study employer prefs (Inge de Wolf 2000))
. list pref female age grades edufit workexp boardexp if caseid==7, noobs
```

| pref | female | age | grades | edufit | workexp    | boardexp |
|------|--------|-----|--------|--------|------------|----------|
| 0    | yes    | 28  | A/B    | no     | none       | no       |
| 0    | no     | 25  | C/D    | yes    | one year   | no       |
| 0    | no     | 25  | C/D    | yes    | none       | yes      |
| 0    | yes    | 25  | C/D    | no     | internship | yes      |
| 1    | no     | 25  | C/D    | yes    | one year   | yes      |
|      |        |     |        |        |            |          |
| 2    | no     | 25  | A/B    | yes    | none       | no       |
| 3    | yes    | 25  | A/B    | yes    | one year   | no       |
| 4    | yes    | 25  | A/B    | yes    | none       | yes      |
| 5    | no     | 25  | A/B    | yes    | internship | no       |
| 6    | yes    | 28  | A/B    | yes    | one year   | yes      |

Here six applicants were selected. The rankings are stored in the variable `pref`, where a value of 6 corresponds to “best among the candidates”, a value of 5 corresponds to “second-best among the candidates”, etc. The applicants with a ranking of 0 were not among the best six candidates for the job. The respondent was not asked to express his preferences among these four applicants, but by the elicitation procedure, it is known that he ranks these four applicants below the six selected applicants. The best candidate was a female, 28 years old, with education fitting the job, with good grades (A/B), with 1 year of work experience, and with experience being a board member of a fraternity, a sports club, etc. The profiles of the other candidates read similarly. Here the respondent completed the task; that is, he selected and rank ordered the six most suitable applicants. Sometimes the respondent performed only part of the task.

```
. list pref female age grades edufit workexp boardexp if caseid==18, noobs
```

| pref | female | age | grades | edufit | workexp    | boardexp |
|------|--------|-----|--------|--------|------------|----------|
| 0    | no     | 25  | C/D    | yes    | none       | yes      |
| 0    | no     | 25  | C/D    | no     | internship | yes      |
| 0    | no     | 28  | C/D    | no     | internship | yes      |
| 0    | yes    | 25  | A/B    | no     | one year   | no       |
| 2    | yes    | 25  | A/B    | no     | none       | yes      |
|      |        |     |        |        |            |          |
| 2    | no     | 25  | A/B    | no     | none       | yes      |
| 2    | no     | 25  | A/B    | no     | one year   | yes      |
| 5    | no     | 25  | A/B    | no     | none       | yes      |
| 5    | no     | 25  | A/B    | no     | none       | yes      |
| 5    | yes    | 25  | A/B    | no     | none       | no       |

The respondent selected the six best candidates and segmented these six candidates into two groups: one group with the three best candidates, and a second group of three candidates that were “still acceptable”. The numbers 2 and 5, indicating these two groups, are arbitrary apart from the implied ranking of the groups. The ties between the candidates in a group indicate that the respondent was not able to rank the candidates within the group.

The purpose of the vignette experiment was to explore and test hypotheses about which of the employees’ attributes are valued by employers, how these attributes are weighted depending on the type of job (described by variable `job` in these data), etc. In the psychometric tradition of [Thurstone \(1927\)](#), *value* is assumed to be linear in the attributes, with the coefficients expressing the direction and weight

of the attributes. In addition, it is assumed that *valuation* is to some extent a random procedure, captured by an additive random term. For instance, if value depends only on an applicant's age and gender, we would have

$$\text{value}(\text{female}_i, \text{age}_i) = \beta_1 \text{female}_i + \beta_2 \text{age}_i + \epsilon_i$$

where the random residual,  $\epsilon_i$ , captures all omitted attributes. Thus  $\beta_1 > 0$  means that the employer assigns higher value to a woman than to a man. Given this conceptualization of value, it is straightforward to model the decision (selection) among alternatives or the ranking of alternatives: the alternative with the highest value is selected (chosen), or the alternatives are ranked according to their value. To complete the specification of a model of choice and of ranking, we assume that the random residual  $\epsilon_i$  follows an “extreme value distribution of type I”, introduced in this context by [Luce \(1959\)](#). This specific assumption is made mostly for computational convenience.

This model is known by many names. Among others, it is known as the rank-ordered logit model in economics ([Beggs, Cardell, and Hausman 1981](#)), as the exploded logit model in marketing research ([Punj and Staelin 1978](#)), as the choice-based conjoint analysis model ([Hair et al. 2010](#)), and as the Plackett–Luce model ([Marden 1995](#)). The model coefficients are estimated using the method of maximum likelihood. The implementation in `rologit` uses an analogy between the rank-ordered logit model and the Cox regression model observed by [Allison and Christakis \(1994\)](#); see [Methods and formulas](#). The `rologit` command implements this method for rankings, whereas `clogit` deals with the variant of choices, that is, only the most highly valued alternative is recorded. In the latter case, the model is also known as the Luce–McFadden choice model. In fact, when the data record the most preferred (unique) alternative and no additional ranking information about preferences is available, `rologit` and `clogit` return the same information, though formatted somewhat differently.

```
. rologit pref female age grades edufit workexp boardexp if job==1, group(caseid)
Iteration 0: log likelihood = -95.41087
Iteration 1: log likelihood = -71.180903
Iteration 2: log likelihood = -68.47734
Iteration 3: log likelihood = -68.345918
Iteration 4: log likelihood = -68.345389
Refining estimates:
Iteration 0: log likelihood = -68.345389

Rank-ordered logistic regression Number of obs = 80
Group variable: caseid Number of groups = 8
No ties in data Obs per group: min = 10
 avg = 10.00
 max = 10
 LR chi2(6) = 54.13
Log likelihood = -68.34539 Prob > chi2 = 0.0000
```

| pref     | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|----------|-----------|-----------|-------|-------|----------------------|----------|
| female   | -.4487287 | .3671307  | -1.22 | 0.222 | -1.168292            | .2708343 |
| age      | -.0984926 | .0820473  | -1.20 | 0.230 | -.2593024            | .0623172 |
| grades   | 3.064534  | .6148245  | 4.98  | 0.000 | 1.8595               | 4.269568 |
| edufit   | .7658064  | .3602366  | 2.13  | 0.034 | .0597556             | 1.471857 |
| workexp  | 1.386427  | .292553   | 4.74  | 0.000 | .8130341             | 1.959821 |
| boardexp | .6944377  | .3762596  | 1.85  | 0.065 | -.0430176            | 1.431893 |

Focusing only on the variables whose coefficients are significant at the 10% level (we are analyzing 8 respondents only!), the estimated value of an applicant for a job of type 1 (research positions) can be written as

$$\text{value} = 3.06 \cdot \text{grades} + 0.77 \cdot \text{edufit} + 1.39 \cdot \text{workexp} + 0.69 \cdot \text{boardexp}$$

Thus employers prefer applicants for a research position (`job==1`) whose educational portfolio fits the job, who have better grades, who have more relevant work experience, and who have (extracurricular) board experience. They do not seem to care much about the sex and age of applicants, which is comforting.

Given these estimates of the valuation by employers, we consider the probabilities that each of the applications is ranked first. Under the assumption that the  $\epsilon_i$  are independent and follow an extreme value type I distribution, [Luce \(1959\)](#) showed that the probability,  $\pi_i$ , that alternative  $i$  is valued higher than alternatives  $2, \dots, k$  can be written in the multinomial logit form

$$\pi_i = \Pr \{ \text{value}_1 > \max(\text{value}_2, \dots, \text{value}_m) \} = \frac{\exp(\text{value}_i)}{\sum_{j=1}^k \exp(\text{value}_i)}$$

The probability of observing a specific ranking can be written as the *product* of such terms, representing a sequential decision interpretation in which the rater first chooses the most preferred alternative, and then the most preferred alternative among the rest, etc.

The probabilities for alternatives to be ranked first are conveniently computed by `predict`.

```
. predict p if e(sample)
(option pr assumed; conditional probability that alternative is ranked first)
(210 missing values generated)
. sort caseid pref p
. list pref p grades edufit workexp boardexp if caseid==7, noobs
```

| pref | p        | grades | edufit | workexp    | boardexp |
|------|----------|--------|--------|------------|----------|
| 0    | .0027178 | C/D    | yes    | none       | yes      |
| 0    | .0032275 | C/D    | no     | internship | yes      |
| 0    | .0064231 | A/B    | no     | none       | no       |
| 0    | .0217202 | C/D    | yes    | one year   | no       |
| 1    | .0434964 | C/D    | yes    | one year   | yes      |
| 2    | .0290762 | A/B    | yes    | none       | no       |
| 3    | .2970933 | A/B    | yes    | one year   | no       |
| 4    | .0371747 | A/B    | yes    | none       | yes      |
| 5    | .1163203 | A/B    | yes    | internship | no       |
| 6    | .4427504 | A/B    | yes    | one year   | yes      |

There clearly is a positive relation between the stated ranking and the predicted probabilities for alternatives to be ranked first, but the association is not perfect. In fact, we would not have expected a perfect association, as the model specifies a (nondegenerate) probability distribution over the possible rankings of the alternatives. These predictions for sets of 10 candidates can also be used to make predictions for subsets of the alternatives. For instance, suppose that only the last three candidates listed in this table would be available. According to parameter estimates of the rank-ordered logit model, the probability that the last of these candidates is selected equals  $0.443 / (0.037 + 0.116 + 0.443) = 0.743$ .

Comparing respondents

The `rologit` model assumes that all respondents, HR managers in large public-sector organizations in The Netherlands, use the *same* valuation function; i.e., they apply the same decision weights. This is the substantive interpretation of the assumption that the  $\beta$ 's are constant between the respondents. To probe this assumption, we could test whether the coefficients vary between different groups of respondents. For a metric characteristic of the HR manager, such as `firmsize`, we can consider a trend-model in the valuation weights,

$$\beta_{ij} = \alpha_{i0} + \alpha_{i1}\text{firmsize}_j$$

and we can test that the slopes  $\alpha_{i1}$  of `firmsize` are zero.

```
. generate firmsize = employer
. rologit pref edufit grades workexp c.firmsize#c.(edufit grades workexp boardexp)
> if job==1, group(caseid) nolog

Rank-ordered logistic regression Number of obs = 80
Group variable: caseid Number of groups = 8
No ties in data Obs per group: min = 10
 avg = 10.00
 max = 10
 LR chi2(7) = 57.17
 Prob > chi2 = 0.0000

Log likelihood = -66.82346
```

| pref                      | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|---------------------------|-----------|-----------|-------|-------|----------------------|----------|
| edufit                    | 1.29122   | 1.13764   | 1.13  | 0.256 | -.9385127            | 3.520953 |
| grades                    | 6.439776  | 2.288056  | 2.81  | 0.005 | 1.955267             | 10.92428 |
| workexp                   | 1.23342   | .8065067  | 1.53  | 0.126 | -.347304             | 2.814144 |
| c.firmsize#<br>c.edufit   | -.0173333 | .0711942  | -0.24 | 0.808 | -.1568714            | .1222048 |
| c.firmsize#<br>c.grades   | -.2099279 | .1218251  | -1.72 | 0.085 | -.4487008            | .028845  |
| c.firmsize#<br>c.workexp  | .0097508  | .0525081  | 0.19  | 0.853 | -.0931632            | .1126649 |
| c.firmsize#<br>c.boardexp | .0382304  | .0227545  | 1.68  | 0.093 | -.0063676            | .0828284 |

```
. testparm c.firmsize#c.(edufit grades workexp boardexp)
(1) c.firmsize#c.edufit = 0
(2) c.firmsize#c.grades = 0
(3) c.firmsize#c.workexp = 0
(4) c.firmsize#c.boardexp = 0

 chi2(4) = 7.14
 Prob > chi2 = 0.1288
```

The Wald test that the slopes of the interacted `firmsize` variables are jointly zero provides no evidence upon which we would reject the null hypothesis; i.e., we do not find evidence against the assumption of constant valuation weights of the attributes by firms of different size. We did not enter `firmsize` as a predictor variable. Characteristics of the decision-making agent do not vary between alternatives. Thus an additive effect of these characteristics on the valuation of alternatives does *not* affect the agent's ranking of alternatives and his choice. Consequently the coefficient of `firmsize` is not identified. `rologit` would in fact have diagnosed the problem and dropped `firmsize` from the analysis. Diagnosing this problem can slow the estimation considerably; the test may be suppressed by specifying the `notestrhs` option.

## Incomplete rankings and ties

`rologit` allows incomplete rankings and ties in the rankings as proposed by Allison and Christakis (1994). `rologit` permits rankings to be incomplete only “at the bottom”; namely, that the ranking of the least attractive alternatives for subjects may not be known—do not confuse this with the situation that a subject is indifferent between these alternatives. This form of incompleteness occurred in the example discussed here, because the respondents were instructed to select and rank only the top six alternatives. It may also be that respondents refused to rank the alternatives that are very unattractive. `rologit` does not allow other forms of incompleteness, for instance, data in which respondents indicate which of four cars they like best, and which one they like least, but not how they rank the two intermediate cars. Another example of incompleteness that cannot be analyzed with `rologit` is data in which respondents select the three alternatives they like best but are not requested to express their preferences among the three selected alternatives.

`rologit` also permits ties in rankings. `rologit` assumes that if a subject expresses a tie between two or more alternatives, he or she actually holds one particular strict preference ordering, but with all possibilities of a strict ordering consistent with the expressed weak ordering being equally probable. For instance, suppose that a respondent ranks alternative 1 highest. He prefers alternatives 2 and 3 over alternative 4, and he is indifferent between alternatives 2 and 3. We assume that this respondent either has the strict preference ordering  $1 > 2 > 3 > 4$  or  $1 > 3 > 2 > 4$ , with both possibilities being equally likely. From a psychometric perspective, it may actually be more appropriate to also assume that the alternatives 2 and 3 are close; for instance, the difference between the associated valuations (utilities) is less than some threshold or minimally discernible difference. Computationally, however, this is a more demanding model.

## Clustered choice data

We have seen that applicants with work experience are in a relatively favorable position. To test whether the effects of work experience vary between the jobs, we can include interactions between the type of job and the attributes of applicants. Such interactions can be obtained using factor variables.

Because some HR managers contributed data for more than one job, we cannot assume that their selection decisions for different jobs are independent. We can account for this by specifying the `vce(cluster clustvar)` option. By treating choice data as incomplete ranking data with only the most preferred alternative marked, `rologit` may be used to estimate the model parameters for clustered choice data.



```
. rologit pref job#c.(female grades edufit workexp), group(caseid)
> vce(cluster employer) nolog
2.job 3.job omitted because of no within-caseid variance
```

|                                   |                    |   |        |
|-----------------------------------|--------------------|---|--------|
| Rank-ordered logistic regression  | Number of obs      | = | 290    |
| Group variable: caseid            | Number of groups   | = | 29     |
| Ties handled via the Efron method | Obs per group: min | = | 10     |
|                                   | avg                | = | 10.00  |
|                                   | max                | = | 10     |
|                                   | Wald chi2(12)      | = | 79.57  |
| Log pseudolikelihood = -296.3855  | Prob > chi2        | = | 0.0000 |

(Std. Err. adjusted for 22 clusters in employer)

| pref              | Coef.     | Robust<br>Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-------------------|-----------|---------------------|-------|-------|----------------------|-----------|
| job               |           |                     |       |       |                      |           |
| 2                 | (omitted) |                     |       |       |                      |           |
| 3                 | (omitted) |                     |       |       |                      |           |
| female            | -.2286609 | .2519883            | -0.91 | 0.364 | -.7225489            | .2652272  |
| grades            | 2.812555  | .8517878            | 3.30  | 0.001 | 1.143081             | 4.482028  |
| edufit            | .7027757  | .2398396            | 2.93  | 0.003 | .2326987             | 1.172853  |
| workexp           | 1.224453  | .3396773            | 3.60  | 0.000 | .5586978             | 1.890208  |
| job#c.female      |           |                     |       |       |                      |           |
| 2                 | .0293815  | .4829166            | 0.06  | 0.951 | -.9171177            | .9758808  |
| 3                 | .1195538  | .3688844            | 0.32  | 0.746 | -.6034463            | .8425538  |
| job#c.grades      |           |                     |       |       |                      |           |
| 2                 | -2.364247 | 1.005963            | -2.35 | 0.019 | -4.335898            | -.3925961 |
| 3                 | -1.88232  | .8995277            | -2.09 | 0.036 | -3.645362            | -.1192782 |
| job#c.edufit      |           |                     |       |       |                      |           |
| 2                 | -.267475  | .4244964            | -0.63 | 0.529 | -1.099473            | .5645226  |
| 3                 | -.3182995 | .3689972            | -0.86 | 0.388 | -1.041521            | .4049217  |
| job#<br>c.workexp |           |                     |       |       |                      |           |
| 2                 | -.6870077 | .3692946            | -1.86 | 0.063 | -1.410812            | .0367964  |
| 3                 | -.4656993 | .4515712            | -1.03 | 0.302 | -1.350763            | .4193639  |

The parameter estimates for the first job type are very similar to those that would have been obtained from an analysis isolated to these data. Differences are due only to an implied change in the method of handling ties. With clustered observations, `rologit` uses Efron's method. If we had specified the `ties(efron)` option with the separate analyses, then the parameter estimates would have been identical to the simultaneous results. Another difference is that `rologit` now reports robust standard errors, adjusted for clustering within respondents. These could have been obtained for the separate analyses, as well by specifying the `vce(robust)` option. In fact, this option would also have forced `rologit` to switch to Efron's approximation as well.

Given the combined results for the three types of jobs, we can test easily whether the weights for the attributes of applicants vary between the jobs, in other words, whether employers are looking for different qualifications in applicants for different jobs. A Wald test for the equality hypothesis of no difference can be obtained with the `testparm` command:

```
. testparm job#c.(female grades edufit workexp)
(1) 2.job#c.female = 0
(2) 3.job#c.female = 0
(3) 2.job#c.grades = 0
(4) 3.job#c.grades = 0
(5) 2.job#c.edufit = 0
(6) 3.job#c.edufit = 0
(7) 2.job#c.workexp = 0
(8) 3.job#c.workexp = 0

 chi2(8) = 14.96
 Prob > chi2 = 0.0599
```

We find only mild evidence that employers look for different qualities in candidates according to the job for which they are being considered.

#### □ Technical note

Allison (1999) stressed that the comparison between groups of the coefficients of logistic regression is problematic, especially in its latent-variable interpretation. In many common latent-variable models, only the regression coefficients divided by the scale of the latent variable are identified. Thus a comparison of logit regression coefficients between, say, men and women is meaningful only if one is willing to argue that the standard deviation of the latent residual does not differ between the sexes. The rank-ordered logit model is also affected by this problem. While we formulated the model with a scale-free residual, we can actually think of the model for the value of an alternative as being scaled by the standard deviation of the random term, representing other relevant attributes of alternatives. Again comparing attribute weights between jobs is meaningful to the extent that we are willing to defend the proposition that “all omitted attributes” are equally important for different kinds of jobs. □

## Comparison of rologit and clogit

The rank-ordered logit model also has a sequential interpretation. A subject first chooses the best among the alternatives. Next he or she selects the best alternative among the remaining alternatives, etc. The decisions at each of the subsequent stages are described by a conditional logit model, and a subject is assumed to apply the same decision weights at each stage. Some authors have expressed concern that later choices may well be made more randomly than the first few decisions. A formalization of this idea is a heteroskedastic version of the rank-ordered logit model in which the scale of the random term increases with the number of decisions made (e.g., Hausman and Ruud [1987]). This extended model is currently not supported by `rologit`. However, the hypothesis that the same decision weights are applied at the first stage and at later stages can be tested by applying a Hausman test.

First, we fit the rank-ordered logit model on the full ranking data for the first type of job,

```
. rologit pref age female edufit grades workexp boardexp if job==1, group(caseid)
> nolog

Rank-ordered logistic regression Number of obs = 80
Group variable: caseid Number of groups = 8
No ties in data Obs per group: min = 10
 avg = 10.00
 max = 10
 LR chi2(6) = 54.13
 Prob > chi2 = 0.0000

Log likelihood = -68.34539
```

| pref     | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|----------|-----------|-----------|-------|-------|----------------------|----------|
| age      | -.0984926 | .0820473  | -1.20 | 0.230 | -.2593024            | .0623172 |
| female   | -.4487287 | .3671307  | -1.22 | 0.222 | -1.168292            | .2708343 |
| edufit   | .7658064  | .3602366  | 2.13  | 0.034 | .0597556             | 1.471857 |
| grades   | 3.064534  | .6148245  | 4.98  | 0.000 | 1.8595               | 4.269568 |
| workexp  | 1.386427  | .292553   | 4.74  | 0.000 | .8130341             | 1.959821 |
| boardexp | .6944377  | .3762596  | 1.85  | 0.065 | -.0430176            | 1.431893 |

and we save the estimates for later use with the `estimates` command.

```
. estimates store Ranking
```

To estimate the decision weights on the basis of the most preferred alternatives only, we create a variable, `best`, that is 1 for the best alternatives, and 0 otherwise. The `by` prefix is useful here.

```
. by caseid (pref), sort: gen best = pref == pref[_N] if job==1
(210 missing values generated)
```

By specifying `(pref)` with `by caseid`, we ensured that the data were sorted in increasing order on `pref` within `caseid`. Hence, the most preferred alternatives are last in the sort order. The expression `pref == pref[_N]` is true (1) for the most preferred alternatives, even if the alternative is not unique, and false (0) otherwise. If the most preferred alternatives were sometimes tied, we could still fit the model for the based-alternatives-only data via `rologit`, but `clogit` would yield different results because it deals with ties in a less appropriate way for continuous valuations. To ascertain whether there are ties in the selected data regarding applicants for research positions, we can combine `by` with `assert`:

```
. by caseid (pref), sort: assert pref[_N-1] != pref[_N] if job==1
```

There are no ties. We can now fit the model on the choice data by using either `clogit` or `rologit`.

*(Continued on next page)*

```
. rologit best age edufit grades workexp boardexp if job==1, group(caseid) nolog
Rank-ordered logistic regression Number of obs = 80
Group variable: caseid Number of groups = 8
No ties in data Obs per group: min = 10
 avg = 10.00
 max = 10
 LR chi2(5) = 17.27
 Prob > chi2 = 0.0040

Log likelihood = -9.783205
```

| best     | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|----------|-----------|-----------|-------|-------|----------------------|----------|
| age      | -.1048959 | .2017068  | -0.52 | 0.603 | -.5002339            | .2904421 |
| edufit   | .4558387  | .9336775  | 0.49  | 0.625 | -1.374136            | 2.285813 |
| grades   | 3.443851  | 1.969002  | 1.75  | 0.080 | -.4153223            | 7.303025 |
| workexp  | 2.545648  | 1.099513  | 2.32  | 0.021 | .3906422             | 4.700655 |
| boardexp | 1.765176  | 1.112763  | 1.59  | 0.113 | -.4157988            | 3.946152 |

```
. estimates store Choice
```

The same results, though with a slightly different formatted header, would have been obtained by using `clogit` on these data.

```
. clogit best age edufit grades workexp boardexp if job==1, group(caseid) nolog
Conditional (fixed-effects) logistic regression Number of obs = 80
 LR chi2(5) = 17.27
 Prob > chi2 = 0.0040
Log likelihood = -9.7832046 Pseudo R2 = 0.4689
```

| best     | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|----------|-----------|-----------|-------|-------|----------------------|----------|
| age      | -.1048959 | .2017068  | -0.52 | 0.603 | -.5002339            | .2904421 |
| edufit   | .4558387  | .9336775  | 0.49  | 0.625 | -1.374136            | 2.285813 |
| grades   | 3.443851  | 1.969002  | 1.75  | 0.080 | -.4153223            | 7.303025 |
| workexp  | 2.545648  | 1.099513  | 2.32  | 0.021 | .3906422             | 4.700655 |
| boardexp | 1.765176  | 1.112763  | 1.59  | 0.113 | -.4157988            | 3.946152 |

The parameters of the ranking and choice models look different, but the standard errors based on the choice data are much larger. Are we estimating parameters with the ranking data that are different from those with the choice data? A Hausman test compares two estimators of a parameter. One of the estimators should be efficient under the null hypothesis, namely, that choosing the second-best alternative is determined with the same decision weights as the best, etc. In our case, the efficient estimator of the decision weights uses the ranking information. The other estimator should be consistent, even if the null hypothesis is false. In our application, this is the estimator that uses the first-choice data only.

```
. hausman Choice Ranking
```

|          | Coefficients  |                | (b-B)<br>Difference | sqrt(diag(V_b-V_B))<br>S.E. |
|----------|---------------|----------------|---------------------|-----------------------------|
|          | (b)<br>Choice | (B)<br>Ranking |                     |                             |
| age      | -.1048959     | -.0984926      | -.0064033           | .1842657                    |
| edufit   | .4558387      | .7658064       | -.3099676           | .8613846                    |
| grades   | 3.443851      | 3.064534       | .3793169            | 1.870551                    |
| workexp  | 2.545648      | 1.386427       | 1.159221            | 1.059878                    |
| boardexp | 1.765176      | .6944377       | 1.070739            | 1.04722                     |

```

 b = consistent under Ho and Ha; obtained from rologit
 B = inconsistent under Ha, efficient under Ho; obtained from rologit
Test: Ho: difference in coefficients not systematic
 chi2(5) = (b-B)'[(V_b-V_B)^(-1)](b-B)
 = 3.05
 Prob>chi2 = 0.6918

```

We do not find evidence for misspecification. We have to be cautious, though, because Hausman-type tests are often not powerful, and the number of observations in our example is very small, which makes the quality of the approximation of the null distribution by a chi-squared test rather uncertain.

## On reversals of rankings

The rank-ordered logit model has a property that you may find unexpected and even unfortunate. Compare two analyses with the rank-ordered logit model, one in which alternatives are ranked from “most attractive” to “least attractive”, the other a reversed analysis in which these alternatives are ranked from “most unattractive” to “least unattractive”. By unattractiveness, you probably mean just the opposite of attractiveness, and you expect that the weights of the attributes in predicting “attractiveness” to be minus the weights in predicting “unattractiveness”. This is, however, *not* true for the rank-ordered logit model. The assumed distribution of the random residual takes the form  $F(\epsilon) = 1 - \exp\{\exp(-\epsilon)\}$ . This distribution is right-skewed. Therefore, slightly different models result from adding and subtracting the random residual, corresponding with high-to-low and low-to-high rankings. Thus the estimated coefficients will differ between the two specifications, though usually not in an important way. You may observe the difference by specifying the `reverse` option of `rologit`. Reversing the rank order makes rankings that are incomplete at the bottom become incomplete at the top. Only the first kind of incompleteness is supported by `rologit`. Thus, for this comparison, we exclude the alternatives that are not ranked, omitting the information that ranked alternatives are preferred over excluded ones.

```

. rologit pref grades edufit workexp boardexp if job==1 & pref!=0, group(caseid)
 (output omitted)
. estimates store Original
. rologit pref grades edufit workexp boardexp if job==1 & pref!=0, group(caseid)
> reverse
 (output omitted)
. estimates store Reversed

```

(Continued on next page)

```
. estimates table Original Reversed, stats(aic bic)
```

| Variable | Original   | Reversed   |
|----------|------------|------------|
| grades   | 2.0032332  | -1.0955335 |
| edufit   | -.13111006 | -.05710681 |
| workexp  | 1.2805373  | -1.2096383 |
| boardexp | .46213212  | -.27200317 |
| aic      | 96.750452  | 99.665642  |
| bic      | 104.23526  | 107.15045  |

Thus, although the weights of the attributes for reversed rankings are indeed mostly of opposite signs, the magnitudes of the weights and their standard errors differ. Which one is more appropriate? We have no advice to offer here. The specific science of the problem will determine what is appropriate, though we would be surprised indeed if this helps here. Formal testing does not help much either, as the models for the original and reversed rankings are not nested. The model-selection indices, such as the AIC and BIC, however, suggest that you stick to the rank-ordered logit model applied to the original ranking rather than to the reversed ranking.

## Saved results

rologit saves the following in `e()`:

|                          |                                                                    |
|--------------------------|--------------------------------------------------------------------|
| Scalars                  |                                                                    |
| <code>e(N)</code>        | number of observations                                             |
| <code>e(ll_0)</code>     | log likelihood of the null model (“all rankings are equiprobable”) |
| <code>e(ll)</code>       | log likelihood                                                     |
| <code>e(df_m)</code>     | model degrees of freedom                                           |
| <code>e(chi2)</code>     | $\chi^2$                                                           |
| <code>e(r2_p)</code>     | pseudo- $R^2$                                                      |
| <code>e(N_g)</code>      | number of groups                                                   |
| <code>e(g_min)</code>    | minimum group size                                                 |
| <code>e(g_avg)</code>    | average group size                                                 |
| <code>e(g_max)</code>    | maximum group size                                                 |
| <code>e(code_inc)</code> | value for incomplete preferences                                   |
| <code>e(N_clust)</code>  | number of clusters                                                 |
| <code>e(rank)</code>     | rank of <code>e(V)</code>                                          |

## Macros

|                              |                                                                 |
|------------------------------|-----------------------------------------------------------------|
| <code>e(cmd)</code>          | <code>rologit</code>                                            |
| <code>e(cmdline)</code>      | command as typed                                                |
| <code>e(depvar)</code>       | name of dependent variable                                      |
| <code>e(group)</code>        | name of <code>group()</code> variable                           |
| <code>e(wtype)</code>        | weight type                                                     |
| <code>e(wexp)</code>         | weight expression                                               |
| <code>e(title)</code>        | title in estimation output                                      |
| <code>e(clustvar)</code>     | name of cluster variable                                        |
| <code>e(offset)</code>       | offset                                                          |
| <code>e(chi2type)</code>     | Wald or LR; type of model $\chi^2$ test                         |
| <code>e(ties)</code>         | <code>breslow</code> , <code>efron</code> , <code>exactm</code> |
| <code>e(vce)</code>          | <code>vcetype</code> specified in <code>vce()</code>            |
| <code>e(vcetype)</code>      | title used to label Std. Err.                                   |
| <code>e(crittype)</code>     | optimization criterion                                          |
| <code>e(properties)</code>   | <code>b V</code>                                                |
| <code>e(predict)</code>      | program used to implement <code>predict</code>                  |
| <code>e(marginsok)</code>    | predictions allowed by <code>margins</code>                     |
| <code>e(marginsnotok)</code> | predictions disallowed by <code>margins</code>                  |
| <code>e(asbalanced)</code>   | factor variables <code>fvset</code> as <code>asbalanced</code>  |
| <code>e(asobserved)</code>   | factor variables <code>fvset</code> as <code>asobserved</code>  |

## Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`rologit` is implemented as an ado-file.

Allison and Christakis (1994) demonstrate that maximum likelihood estimates for the rank-ordered logit model can be obtained as the maximum partial-likelihood estimates of an appropriately specified Cox regression model for waiting time ([ST] `stcox`). In this analogy, a higher value for an alternative is formally equivalent to a higher hazard rate of failure. `rologit` uses `stcox` to fit the rank-ordered logit model based on such a specification of the data in Cox terms. A higher stated preference is represented by a shorter waiting time until failure. Incomplete rankings are dealt with via censoring. Moreover, decision situations (subjects) are to be treated as strata. Finally, as proposed by Allison and Christakis, ties in rankings are handled by the marginal-likelihood method, specifying that all strict preference orderings consistent with the stated weak preference ordering are equally likely. The marginal-likelihood estimator is available in `stcox` via the `exactm` option. The approximations of the marginal likelihood due to Breslow and Efron are also appropriate for the analysis of rank-ordered logit models. Because in most applications the number of ranked alternatives by one subject will be fairly small (at most, say, 20), the number of ties is small as well, and so you rarely will need to turn to approximations to restrict computer time. Because the marginal-likelihood estimator in `stcox` does not support the cluster adjustment or `pweights`, you should use the Efron approximation in such cases.

This command supports the clustered version of the Huber/White/sandwich estimator of the variance using `vce(robust)` and `vce(cluster clustvar)`. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#). Specifying `vce(robust)` is equivalent to specifying `vce(cluster groupvar)`, where `groupvar` is the identifier variable that links the alternatives.

## Acknowledgment

The `rologit` command was written by Jeroen Weesie, Department of Sociology, Utrecht University, The Netherlands.

## References

- Allison, P. D. 1999. Comparing logit and probit coefficients across groups. *Sociological Methods and Research* 28: 186–208.
- Allison, P. D., and N. Christakis. 1994. Logit models for sets of ranked items. In Vol. 24 of *Sociological Methodology*, ed. P. V. Marsden, 123–126. Oxford: Blackwell.
- Beggs, S., S. Cardell, and J. A. Hausman. 1981. Assessing the potential demand for electric cars. *Journal of Econometrics* 17: 1–19.
- de Wolf, I. 2000. *Opleidingsspecialisatie en arbeidsmarktsucces van sociale wetenschappers*. Amsterdam: ThelaThesis.
- Hair Jr., J. F., W. C. Black, B. J. Babin, and R. E. Anderson. 2010. *Multivariate Data Analysis*. 7th ed. Upper Saddle River, NJ: Pearson.
- Hausman, J. A., and P. A. Ruud. 1987. Specifying and testing econometric models for rank-ordered data. *Journal of Econometrics* 34: 83–104.
- Luce, R. D. 1959. *Individual Choice Behavior: A Theoretical Analysis*. New York: Dover.
- Marden, J. I. 1995. *Analyzing and Modeling Rank Data*. London: Chapman & Hall.
- McCullagh, P. 1993. Permutations and regression models. In *Probability Models and Statistical Analysis for Ranking Data*, ed. M. A. Fligner and J. S. Verducci, 196–215. New York: Springer.
- Plackett, R. L. 1975. The analysis of permutations. *Applied Statistics* 24: 193–202.
- Punj, G. N., and R. Staelin. 1978. The choice process for graduate business schools. *Journal of Marketing Research* 15: 588–598.
- Thurstone, L. L. 1927. A law of comparative judgment. *Psychological Reviews* 34: 273–286.
- Yellott Jr., J. I. 1977. The relationship between Luce’s choice axiom, Thurstone’s theory of comparative judgment, and the double exponential distribution. *Journal of Mathematical Psychology* 15: 109–144.

## Also see

- [R] [rologit postestimation](#) — Postestimation tools for `rologit`
- [R] [clogit](#) — Conditional (fixed-effects) logistic regression
- [R] [logistic](#) — Logistic regression, reporting odds ratios
- [R] [mlogit](#) — Multinomial (polytomous) logistic regression
- [R] [nlogit](#) — Nested logit regression
- [R] [slogit](#) — Stereotype logistic regression
- [U] [20 Estimation and postestimation commands](#)



Description

The following postestimation commands are available for `rologit`:

| command                           | description                                                                                         |
|-----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>                | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estimates</code>            | cataloging estimation results                                                                       |
| <code>hausman</code>              | Hausman's specification test                                                                        |
| <code>lincom</code>               | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>             | link test for model specification                                                                   |
| <code>lrtest</code>               | likelihood-ratio test                                                                               |
| <code>margins</code> <sup>1</sup> | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>                | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>              | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>            | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>                 | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>               | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> The default prediction statistic `pr` cannot be correctly handled by `margins`; however, `margins` can be used after `rologit` with the `predict(xb)` option.

See the corresponding entries in the *Base Reference Manual* for details.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic nooffset]
```

| statistic         | description                                                 |
|-------------------|-------------------------------------------------------------|
| <code>pr</code>   | probability that alternatives are ranked first; the default |
| <code>xb</code>   | linear prediction                                           |
| <code>stdp</code> | standard error of the linear prediction                     |

These statistics are available both in and out of sample; type `predict ... if esample() ...` if wanted only for the estimation sample.

Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

`pr`, the default, calculates the probability that alternatives are ranked first.

`xb` calculates the linear prediction.

`stdp` calculates the standard error of the linear prediction.

`nooffset` is relevant only if you specified `offset(varname)` for `rologit`. It modifies the calculations made by `predict` so that they ignore the offset variable; the linear prediction is treated as  $\mathbf{x}_j\mathbf{b}$  rather than as  $\mathbf{x}_j\mathbf{b} + \text{offset}_j$ .

## Remarks

See *Comparing respondents* and *Clustered choice data* in [R] **rologit** for examples of the use of `testparm`, an alternative to the `test` command.

See *Comparison of rologit and clogit* and *On reversals of rankings* in [R] **rologit** for examples of the use of `estimates`.

See *Comparison of rologit and clogit* in [R] **rologit** for an example of the use of `hausman`.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] **rologit** — Rank-ordered logistic regression

[U] **20 Estimation and postestimation commands**

Syntax

```
rreg depvar [indepvars] [if] [in] [, options]
```

| <i>options</i>                 | description                                                               |
|--------------------------------|---------------------------------------------------------------------------|
| Model                          |                                                                           |
| <u>tune</u> (#)                | use # as the biweight tuning constant; default is <code>tune(7)</code>    |
| Reporting                      |                                                                           |
| <u>level</u> (#)               | set confidence level; default is <code>level(95)</code>                   |
| <u>genwt</u> ( <i>newvar</i> ) | create <i>newvar</i> containing the weights assigned to each observation  |
| <u>display_options</u>         | control spacing and display of omitted variables and base and empty cells |
| Optimization                   |                                                                           |
| <u>optimization_options</u>    | control the optimization process; seldom used                             |
| <u>graph</u>                   | graph weights during convergence                                          |
| <sup>†</sup> <u>coeflegend</u> | display coefficients' legend instead of coefficient table                 |

<sup>†</sup>`coeflegend` does not appear in the dialog box.  
*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.  
*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.  
`by`, `fracpoly`, `mfp`, `mi estimate`, `rolling`, and `statsby` are allowed; see [U] 11.1.10 Prefix commands.  
See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Linear models and related > Other > Robust regression

Description

`rreg` performs one version of robust regression of *depvar* on *indepvars*.

Also see *Robust standard errors* in [R] `regress` for standard regression with robust variance estimates and [R] `qreg` for quantile (including median or least-absolute-residual) regression.

Options

Model

`tune(#)` is the biweight tuning constant. The default is 7, meaning seven times the median absolute deviation (MAD) from the median residual; see *Methods and formulas*. Lower tuning constants downweight outliers rapidly but may lead to unstable estimates (less than 6 is not recommended). Higher tuning constants produce milder downweighting.

Reporting

`level(#)`; see [R] [estimation options](#).  
`genwt(newvar)` creates the new variable `newvar` containing the weights assigned to each observation.  
`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

Optimization

`optimization_options`: `iterate(#)`, `tolerance(#)`, `[no]log`. `iterate()` specifies the maximum number of iterations; iterations stop when the maximum change in weights drops below `tolerance()`; and `log/nolog` specifies whether to show the iteration log. These options are seldom used.  
`graph` allows you to graphically watch the convergence of the iterative technique. The weights obtained from the most recent round of estimation are graphed against the weights obtained from the previous round.

The following option is available with `rreg` but is not shown in the dialog box:  
`coeflegend`; see [R] [estimation options](#).

Remarks

`rreg` first performs an initial screening based on Cook’s distance > 1 to eliminate gross outliers before calculating starting values and then performs Huber iterations followed by biweight iterations, as suggested by Li (1985).

Example 1

We wish to examine the relationship between mileage rating, weight, and location of manufacture for the 74 cars in our automobile data. As a point of comparison, we begin by fitting an ordinary regression:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. regress mpg weight foreign
```

| Source   | SS         | df | MS         |                 |        |  |
|----------|------------|----|------------|-----------------|--------|--|
| Model    | 1619.2877  | 2  | 809.643849 | Number of obs = | 74     |  |
| Residual | 824.171761 | 71 | 11.608053  | F( 2, 71) =     | 69.75  |  |
|          |            |    |            | Prob > F =      | 0.0000 |  |
|          |            |    |            | R-squared =     | 0.6627 |  |
|          |            |    |            | Adj R-squared = | 0.6532 |  |
| Total    | 2443.45946 | 73 | 33.4720474 | Root MSE =      | 3.4071 |  |

| mpg     | Coef.     | Std. Err. | t      | P> t  | [95% Conf. Interval] |
|---------|-----------|-----------|--------|-------|----------------------|
| weight  | -.0065879 | .0006371  | -10.34 | 0.000 | -.0078583 -.0053175  |
| foreign | -1.650029 | 1.075994  | -1.53  | 0.130 | -3.7955 .4954422     |
| _cons   | 41.6797   | 2.165547  | 19.25  | 0.000 | 37.36172 45.99768    |

We now compare this with the results from rreg:

```
. rreg mpg weight foreign
 Huber iteration 1: maximum difference in weights = .80280176
 Huber iteration 2: maximum difference in weights = .2915438
 Huber iteration 3: maximum difference in weights = .08911171
 Huber iteration 4: maximum difference in weights = .02697328
 Biweight iteration 5: maximum difference in weights = .29186818
 Biweight iteration 6: maximum difference in weights = .11988101
 Biweight iteration 7: maximum difference in weights = .03315872
 Biweight iteration 8: maximum difference in weights = .00721325
Robust regression Number of obs = 74
 F(2, 71) = 168.32
 Prob > F = 0.0000
```

| mpg     | Coef.     | Std. Err. | t      | P> t  | [95% Conf. Interval] |           |
|---------|-----------|-----------|--------|-------|----------------------|-----------|
| weight  | -.0063976 | .0003718  | -17.21 | 0.000 | -.007139             | -.0056562 |
| foreign | -3.182639 | .627964   | -5.07  | 0.000 | -4.434763            | -1.930514 |
| _cons   | 40.64022  | 1.263841  | 32.16  | 0.000 | 38.1202              | 43.16025  |

Note the large change in the foreign coefficient.



□ Technical note

It would have been better if we had fit the previous robust regression by typing `rreg mpg weight foreign, genwt(w)`. The new variable, `w`, would then contain the estimated weights. Let's pretend that we did this:

```
. rreg mpg weight foreign, genwt(w)
(output omitted)
. summarize w, detail

 Robust Regression Weight
+-----+-----+-----+-----+
Percentiles Smallest
1% 0
5% .0442957
10% .4674935
25% .8894815 .0442957 Obs 74
50% .9690193
75% .9949395 Largest Sum of Wgt. 74
90% .9989245 .9996715 Mean .8509966
95% .9996715 .9996953 Std. Dev. .2746451
99% .9998585 .9997343 Variance .0754299
 .9998585 Skewness -2.287952
 Kurtosis 6.874605
```

(Continued on next page)

We discover that 3 observations in our data were dropped altogether (they have weight 0). We could further explore our data:

```
. sort w
. list make mpg weight w if w<.467, sep(0)
```

|    | make           | mpg | weight | w         |
|----|----------------|-----|--------|-----------|
| 1. | Datsun 210     | 35  | 2,020  | 0         |
| 2. | Subaru         | 35  | 2,050  | 0         |
| 3. | VW Diesel      | 41  | 2,040  | 0         |
| 4. | Plym. Arrow    | 28  | 3,260  | .04429567 |
| 5. | Cad. Seville   | 21  | 4,290  | .08241943 |
| 6. | Toyota Corolla | 31  | 2,200  | .10443129 |
| 7. | Olds 98        | 21  | 4,060  | .28141296 |

Being familiar with the automobile data, we immediately spotted two things: the VW is the only diesel car in our data, and the weight recorded for the Plymouth Arrow is incorrect. □

➤ Example 2

If we specify no explanatory variables, rreg produces a robust estimate of the mean:

```
. rreg mpg
 Huber iteration 1: maximum difference in weights = .64471879
 Huber iteration 2: maximum difference in weights = .05098336
 Huber iteration 3: maximum difference in weights = .0099887
 Biweight iteration 4: maximum difference in weights = .25197391
 Biweight iteration 5: maximum difference in weights = .00358606
Robust regression Number of obs = 74
 F(0, 73) = 0.00
 Prob > F = .
```

| mpg   | Coef.    | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|-------|----------|-----------|-------|-------|----------------------|----------|
| _cons | 20.68825 | .641813   | 32.23 | 0.000 | 19.40912             | 21.96738 |

The estimate is given by the coefficient on \_cons. The mean is 20.69 with an estimated standard error of 0.6418. The 95% confidence interval is [ 19.4, 22.0]. By comparison, ci gives us the standard calculation:

```
. ci mpg
```

| Variable | Obs | Mean    | Std. Err. | [95% Conf. Interval] |          |
|----------|-----|---------|-----------|----------------------|----------|
| mpg      | 74  | 21.2973 | .6725511  | 19.9569              | 22.63769 |

## Saved results

`rreg` saves the following in `e()`:

### Scalars

|                      |                             |
|----------------------|-----------------------------|
| <code>e(N)</code>    | number of observations      |
| <code>e(mss)</code>  | model sum of squares        |
| <code>e(df_m)</code> | model degrees of freedom    |
| <code>e(rss)</code>  | residual sum of squares     |
| <code>e(df_r)</code> | residual degrees of freedom |
| <code>e(r2)</code>   | <i>R</i> -squared           |
| <code>e(r2_a)</code> | adjusted <i>R</i> -squared  |
| <code>e(F)</code>    | <i>F</i> statistic          |
| <code>e(rmse)</code> | root mean squared error     |
| <code>e(rank)</code> | rank of <code>e(V)</code>   |

### Macros

|                            |                                                                |
|----------------------------|----------------------------------------------------------------|
| <code>e(cmd)</code>        | <code>rreg</code>                                              |
| <code>e(cmdline)</code>    | command as typed                                               |
| <code>e(depvar)</code>     | name of dependent variable                                     |
| <code>e(genwt)</code>      | variable containing the weights                                |
| <code>e(title)</code>      | title in estimation output                                     |
| <code>e(model)</code>      | ols                                                            |
| <code>e(vce)</code>        | ols                                                            |
| <code>e(properties)</code> | <code>b V</code>                                               |
| <code>e(predict)</code>    | program used to implement <code>predict</code>                 |
| <code>e(marginsok)</code>  | predictions allowed by <code>margins</code>                    |
| <code>e(asbalanced)</code> | factor variables <code>fvset</code> as <code>asbalanced</code> |
| <code>e(asobserved)</code> | factor variables <code>fvset</code> as <code>asobserved</code> |

### Matrices

|                   |                                              |
|-------------------|----------------------------------------------|
| <code>e(b)</code> | coefficient vector                           |
| <code>e(V)</code> | variance–covariance matrix of the estimators |

### Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`rreg` is implemented as an ado-file.

See [Berk \(1990\)](#), [Goodall \(1983\)](#), and [Rousseeuw and Leroy \(1987\)](#) for a general description of the issues and methods. [Hamilton \(1991a, 1992\)](#) provides a more detailed description of `rreg` and some Monte Carlo evaluations.

`rreg` begins by fitting the regression (see [\[R\] regress](#)), calculating Cook’s *D* (see [\[R\] predict](#) and [\[R\] regress postestimation](#)), and excluding any observation for which  $D > 1$ .

Thereafter `rreg` works iteratively: it performs a regression, calculates case weights from absolute residuals, and regresses again using those weights. Iterations stop when the maximum change in weights drops below `tolerance()`. Weights derive from one of two weight functions, Huber weights and biweights. Huber weights ([Huber 1964](#)) are used until convergence, and then, from that result, biweights are used until convergence. The biweight was proposed by [Beaton and Tukey \(1974, 151–152\)](#) after the Princeton robustness study ([Andrews et al. 1972](#)) had compared various estimators.

Both weighting functions are used because Huber weights have problems dealing with severe outliers, whereas biweights sometimes fail to converge or have multiple solutions. The initial Huber weighting should improve the behavior of the biweight estimator.

In Huber weighting, cases with small residuals receive weights of 1; cases with larger residuals receive gradually smaller weights. Let  $e_i = y_i - \mathbf{X}_i\mathbf{b}$  represent the  $i$ th-case residual. The  $i$ th scaled residual  $u_i = e_i/s$  is calculated, where  $s = M/0.6745$  is the residual scale estimate and  $M = \text{med}(|e_i - \text{med}(e_i)|)$  is the median absolute deviation from the median residual. Huber estimation obtains case weights:

$$w_i = \begin{cases} 1 & \text{if } |u_i| \leq c_h \\ c_h/|u_i| & \text{otherwise} \end{cases}$$

**rreg** defines  $c_h = 1.345$ , so downweighting begins with cases whose absolute residual exceeds  $(1.345/0.6745)M \approx 2M$ .

With biweights, all cases with nonzero residuals receive some downweighting, according to the smoothly decreasing biweight function

$$w_i = \begin{cases} \{1 - (u_i/c_b)^2\}^2 & \text{if } |u_i| \leq c_b \\ 0 & \text{otherwise} \end{cases}$$

where  $c_b = 4.685 \times \text{tune}()/7$ . Thus when  $\text{tune}() = 7$ , cases with absolute residuals of  $(4.685/0.6745)M \approx 7M$  or more are assigned 0 weight and thus are effectively dropped. Goodall (1983, 377) suggests using a value between 6 and 9, inclusive, for  $\text{tune}()$  in the biweight case and states that performance is good between 6 and 12, inclusive.

The tuning constants  $c_h = 1.345$  and  $c_b = 4.685$  (assuming  $\text{tune}()$  is set at the default 7) give **rreg** about 95% of the efficiency of OLS when applied to data with normally distributed errors (Hamilton 1991b). Lower tuning constants downweight outliers more drastically (but give up Gaussian efficiency); higher tuning constants make the estimator more like OLS.

Standard errors are calculated using the pseudovalues approach described in Street, Carroll, and Ruppert (1988).

## Acknowledgment

The current version of **rreg** is due to the work of Lawrence Hamilton, Department of Sociology, University of New Hampshire.

## References

- Andrews, D. F., P. J. Bickel, F. R. Hampel, P. J. Huber, W. H. Rogers, and J. W. Tukey. 1972. *Robust Estimates of Location: Survey and Advances*. Princeton: Princeton University Press.
- Beaton, A. E., and J. W. Tukey. 1974. The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data. *Technometrics* 16: 147–185.
- Berk, R. A. 1990. A primer on robust regression. In *Modern Methods of Data Analysis*, ed. J. Fox and J. S. Long, 292–324. Newbury Park, CA: Sage.
- Goodall, C. 1983. M-estimators of location: An outline of the theory. In *Understanding Robust and Exploratory Data Analysis*, ed. D. C. Hoaglin, F. Mosteller, and J. W. Tukey, 339–431. New York: Wiley.
- Gould, W. W., and W. H. Rogers. 1994. Quantile regression as an alternative to robust regression. In *1994 Proceedings of the Statistical Computing Section*. Alexandria, VA: American Statistical Association.
- Hamilton, L. C. 1991a. [srd1: How robust is robust regression?](#) *Stata Technical Bulletin* 2: 21–26. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 169–175. College Station, TX: Stata Press.



- . 1991b. [ssi2: Bootstrap programming](#). *Stata Technical Bulletin* 4: 18–27. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 208–220. College Station, TX: Stata Press.
- . 1992. *Regression with Graphics: A Second Course in Applied Statistics*. Belmont, CA: Duxbury.
- . 2009. *Statistics with Stata (Updated for Version 10)*. Belmont, CA: Brooks/Cole.
- Huber, P. J. 1964. Robust estimation of a location parameter. *Annals of Mathematical Statistics* 35: 73–101.
- Li, G. 1985. Robust regression. In *Exploring Data Tables, Trends, and Shapes*, ed. D. C. Hoaglin, F. Mosteller, and J. W. Tukey, 281–340. New York: Wiley.
- Mosteller, F., and J. W. Tukey. 1977. *Data Analysis and Regression: A Second Course in Statistics*. Reading, MA: Addison–Wesley.
- Relles, D. A., and W. H. Rogers. 1977. Statisticians are fairly robust estimators of location. *Journal of the American Statistical Association* 72: 107–111.
- Rousseeuw, P. J., and A. M. Leroy. 1987. *Robust Regression and Outlier Detection*. New York: Wiley.
- Street, J. O., R. J. Carroll, and D. Ruppert. 1988. A note on computing robust regression estimates via iteratively reweighted least squares. *American Statistician* 42: 152–154.

## Also see

- [R] [rreg postestimation](#) — Postestimation tools for rreg
- [R] [qreg](#) — Quantile regression
- [R] [regress](#) — Linear regression
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `rreg`:

| command                | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>     | VCE and estimation sample summary                                                                   |
| <code>estimates</code> | cataloging estimation results                                                                       |
| <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>margins</code>   | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>   | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code> | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic]
```

| statistic              | description                             |
|------------------------|-----------------------------------------|
| <code>xb</code>        | linear prediction; the default          |
| <code>stdp</code>      | standard error of the linear prediction |
| <code>residuals</code> | residuals                               |
| <code>hat</code>       | diagonal elements of the hat matrix     |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

Menu

Statistics > Postestimation > Predictions, residuals, etc.

Options for predict

- `xb`, the default, calculates the linear prediction.
- `stdp` calculates the standard error of the linear prediction.
- `residuals` calculates the residuals.

`hat` calculates the diagonal elements of the hat matrix. You must have run the `rreg` command with the `genwt()` option.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [rreg](#) — Robust regression

[U] [20 Estimation and postestimation commands](#)

Syntax

```
runtest varname [in] [, options]
```

| options             | description                                                                                                |
|---------------------|------------------------------------------------------------------------------------------------------------|
| <u>continuity</u>   | continuity correction                                                                                      |
| <u>drop</u>         | ignore values equal to the threshold                                                                       |
| <u>split</u>        | randomly split values equal to the threshold as above or below the threshold; default is to count as below |
| <u>mean</u>         | use mean as threshold; default is median                                                                   |
| <u>threshold(#)</u> | assign arbitrary threshold; default is median                                                              |

Menu

Statistics > Nonparametric analysis > Tests of hypotheses > Test for random order

Description

`runtest` tests whether the observations of *varname* are serially independent—that is, whether they occur in a random order—by counting how many runs there are above and below a threshold. By default, the median is used as the threshold. A small number of runs indicates positive serial correlation; a large number indicates negative serial correlation.

Options

- `continuity` specifies a continuity correction that may be helpful in small samples. If there are fewer than 10 observations either above or below the threshold, however, the tables in [Swed and Eisenhart \(1943\)](#) provide more reliable critical values. By default, no continuity correction is used.
- `drop` directs `runtest` to ignore any values of *varname* that are equal to the threshold value when counting runs and tabulating observations. By default, `runtest` counts a value as being above the threshold when it is strictly above the threshold and as being below the threshold when it is less than or equal to the threshold.
- `split` directs `runtest` to randomly split values of *varname* that are equal to the threshold. In other words, when *varname* is equal to threshold, a “coin” is flipped. If it comes up heads, the value is counted as above the threshold. If it comes up tails, the value is counted as below the threshold.
- `mean` directs `runtest` to tabulate runs above and below the mean rather than the median.
- `threshold(#)` specifies an arbitrary threshold to use in counting runs. For example, if *varname* has already been coded as a 0/1 variable, the median generally will not be a meaningful separating value.

## Remarks

`runtest` performs a nonparametric test of the hypothesis that the observations of *varname* occur in a random order by counting how many runs there are above and below a threshold. If *varname* is positively serially correlated, it will tend to remain above or below its median for several observations in a row; that is, there will be relatively few runs. If, on the other hand, *varname* is negatively serially correlated, observations above the median will tend to be followed by observations below the median and vice versa; that is, there will be relatively many runs.

By default, `runtest` uses the median for the threshold, and this is not necessarily the best choice. If `mean` is specified, the mean is used instead of the median. If `threshold(#)` is specified, `#` is used. Because `runtest` divides the data into two states—above and below the threshold—it is appropriate for data that are already binary; for example, win or lose, live or die, rich or poor, etc. Such variables are often coded as 0 for one state and 1 for the other. Here you should specify `threshold(0)` because, by default, `runtest` separates the observations into those that are greater than the threshold and those that are less than or equal to the threshold.

As with most nonparametric procedures, the treatment of ties complicates the test. Observations equal to the threshold value are ties and can be treated in one of three ways. By default, they are treated as if they were below the threshold. If `drop` is specified, they are omitted from the calculation and the total number of observations is adjusted. If `split` is specified, each is randomly assigned to the above- and below-threshold groups. The random assignment is different each time the procedure is run unless you specify the random-number seed; see [\[R\] set seed](#).

### ► Example 1

We can use `runtest` to check regression residuals for serial correlation.

```
. use http://www.stata-press.com/data/r11/run1
. scatter resid year, connect(1) yline(0) title(Regression residuals)
```



The graph gives the impression that these residuals are positively correlated. Excursions above or below zero—the natural threshold for regression residuals—tend to last for several observations. `runtest` can evaluate the statistical significance of this impression.

```
. runtest resid, thresh(0)
N(resid <= 0) = 8
N(resid > 0) = 8
 obs = 16
 N(runs) = 5
 z = -2.07
 Prob>|z| = .04
```

There are five runs in these 16 observations. Using the normal approximation to the true distribution of the number of runs, the five runs in this series are fewer than would be expected if the residuals were serially independent. The  $p$ -value is 0.04, indicating a two-sided significant result at the 5% level. If the alternative hypothesis is positive serial correlation, rather than any deviation from randomness, then the one-sided  $p$ -value is  $0.04/2 = 0.015$ . With so few observations, however, the normal approximation may be inaccurate. (Tables compiled by Swed and Eisenhart list five runs as the 5% critical value for a one-sided test.)

`runtest` is a nonparametric test. It ignores the magnitudes of the observations and notes only whether the values are above or below the threshold. We can demonstrate this feature by reducing the information about the regression residuals in this example to a 0/1 variable that indicates only whether a residual is positive or negative.

```
. generate byte sign = resid>0
. runtest sign, thresh(0)
N(sign <= 0) = 8
N(sign > 0) = 8
 obs = 16
 N(runs) = 5
 z = -2.07
 Prob>|z| = .04
```

As expected, `runtest` produces the same answer as before.

◀

## □ Technical note

The run test can also be used to test the null hypothesis that two samples are drawn from the same underlying distribution. The run test is sensitive to differences in the shapes, as well as the locations, of the empirical distributions.

Suppose, for example, that two different additives are added to the oil in 10 different cars during an oil change. The cars are run until a viscosity test determines that another oil change is needed, and the number of miles traveled between oil changes is recorded. The data are

```
. use http://www.stata-press.com/data/r11/additive, clear
. list
```

|     | additive | miles |
|-----|----------|-------|
| 1.  | 1        | 4024  |
| 2.  | 1        | 4756  |
| 3.  | 1        | 7993  |
| 4.  | 1        | 5025  |
| 5.  | 1        | 4188  |
| 6.  | 2        | 3007  |
| 7.  | 2        | 1988  |
| 8.  | 2        | 1051  |
| 9.  | 2        | 4478  |
| 10. | 2        | 4232  |

To test whether the additives generate different distributions of miles between oil changes, we sort the data by `miles` and then use `runtest` to see whether the marker for each additive occurs in random order:

```
. sort miles
. runtest additive, thresh(1)
N(additive <= 1) = 5
N(additive > 1) = 5
 obs = 10
 N(runs) = 4
 z = -1.34
 Prob>|z| = .18
```

Here the additives do not produce statistically different results.



## □ Technical note

A test that is related to the run test is the runs up-and-down test. In the latter test, the data are classified not by whether they lie above or below a threshold but by whether they are steadily increasing or decreasing. Thus an unbroken string of increases in the variable of interest is counted as one run, as is an unbroken string of decreases. According to [Madansky \(1988\)](#), the run test is superior to the runs up-and-down test for detecting trends in the data, but the runs up-and-down test is superior for detecting autocorrelation.

`runtest` can be used to perform a runs up-and-down test. Using the regression residuals from the example above, we can perform a `runtest` on their first differences:

```
. use http://www.stata-press.com/data/r11/run1
. generate resid_D = resid - resid[_n-1]
(1 missing value generated)
. runtest resid_D, thresh(0)
N(resid_D <= 0) = 7
N(resid_D > 0) = 8
 obs = 15
 N(runs) = 6
 z = -1.33
 Prob>|z| = .18
```

Edgington (1961) has compiled a table of the small sample distribution of the runs up-and-down statistic, and this table is reprinted in Madansky (1988). For large samples, the  $z$  statistic reported by `runtest` is incorrect for the runs up-and-down test. Let  $N$  be the number of observations (15 here), and let  $r$  be the number of runs (6). The expected number of runs in the runs up-and-down test is

$$\mu_r = \frac{2N - 1}{3}$$

the variance is

$$\sigma_r^2 = \frac{16N - 29}{90}$$

and the correct  $z$  statistic is

$$\hat{z} = \frac{r - \mu_r}{\sigma_r}$$



□ Technical note

`runtest` will tolerate missing values at the beginning or end of a series, as occurred in the technical note above (generating first differences resulted in a missing value for the first observation). `runtest`, however, will issue an error message if there are any missing observations in the interior of the series (in the portion covered by the `in range` modifier). To perform the test anyway, simply drop the missing observations before using `runtest`.



Saved results

`runtest` saves the following in `r()`:

|                         |                            |                        |                                |
|-------------------------|----------------------------|------------------------|--------------------------------|
| Scalars                 |                            |                        |                                |
| <code>r(N)</code>       | number of observations     | <code>r(p)</code>      | $p$ -value of $z$              |
| <code>r(N_below)</code> | number below the threshold | <code>r(z)</code>      | $z$ statistic                  |
| <code>r(N_above)</code> | number above the threshold | <code>r(n_runs)</code> | number of runs                 |
| <code>r(mean)</code>    | expected number of runs    | <code>r(Var)</code>    | variance of the number of runs |

Methods and formulas

`runtest` is implemented as an ado-file.

`runtest` begins by calculating the number of observations below the threshold,  $n_0$ ; the number of observations above the threshold,  $n_1$ ; the total number of observations,  $N = n_0 + n_1$ ; and the number of runs,  $r$ . These statistics are always reported, so the exact tables of critical values in Swed and Eisenhart (1943) may be consulted if necessary.

The expected number of runs under the null is

$$\mu_r = \frac{2n_0n_1}{N} + 1$$



the variance is

$$\sigma_r^2 = \frac{2n_0n_1(2n_0n_1 - N)}{N^2(N - 1)}$$

and the normal approximation test statistic is

$$\hat{z} = \frac{r - \mu_r}{\sigma_r}$$

## Acknowledgment

`runtest` was written by Sean Beckett, a past editor of the *Stata Technical Bulletin*.

## References

- Edgington, E. S. 1961. Probability table for number of runs of signs of first differences in ordered series. *Journal of the American Statistical Association* 56: 156–159.
- Madansky, A. 1988. *Prescriptions for Working Statisticians*. New York: Springer.
- Swed, F. S., and C. Eisenhart. 1943. Tables for testing randomness of grouping in a sequence of alternatives. *Annals of Mathematical Statistics* 14: 66–87.

Syntax

```
sampsi #1 #2 [, options]
```

| options                         | description                                                                                                                                                         |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Main                            |                                                                                                                                                                     |
| <u>onesample</u>                | one-sample test; default is two-sample                                                                                                                              |
| <u>sd1</u> (#)                  | standard deviation of sample 1                                                                                                                                      |
| <u>sd2</u> (#)                  | standard deviation of sample 2                                                                                                                                      |
| Options                         |                                                                                                                                                                     |
| <u>alpha</u> (#)                | significance level of test; default is <code>alpha(0.05)</code>                                                                                                     |
| <u>power</u> (#)                | power of test; default is <code>power(0.90)</code>                                                                                                                  |
| <u>n1</u> (#)                   | size of sample 1                                                                                                                                                    |
| <u>n2</u> (#)                   | size of sample 2                                                                                                                                                    |
| <u>ratio</u> (#)                | ratio of sample sizes; default is <code>ratio(1)</code>                                                                                                             |
| <u>pre</u> (#)                  | number of baseline measurements; default is <code>pre(0)</code>                                                                                                     |
| <u>post</u> (#)                 | number of follow-up measurements; default is <code>post(1)</code>                                                                                                   |
| <u>nocontinuity</u>             | do not use continuity correction for two-sample test on proportions                                                                                                 |
| <u>r0</u> (#)                   | correlation between baseline measurements; default is <code>r0()=r1()</code>                                                                                        |
| <u>r1</u> (#)                   | correlation between follow-up measurements                                                                                                                          |
| <u>r01</u> (#)                  | correlation between baseline and follow-up measurements                                                                                                             |
| <u>onesided</u>                 | one-sided test; default is two-sided                                                                                                                                |
| <u>method</u> ( <i>method</i> ) | analysis method where <i>method</i> is <code>post</code> , <code>change</code> , <code>ancova</code> , or <code>all</code> ;<br>default is <code>method(all)</code> |

Menu

sampsi

Statistics > Power and sample size > Tests of means and proportions

sampsi with repeated measures

Statistics > Power and sample size > Tests of means with repeated measures

Description

sampsi estimates require sample size or power of tests for studies comparing two groups. sampsi can be used when comparing means or proportions for simple studies where only one measurement of the outcome is planned and for comparing mean summary statistics for more complex studies where repeated measurements of the outcome on each experimental unit are planned.

If `n1(#)` or `n2(#)` is specified, `samps1` computes power; otherwise, it computes sample size. For simple studies, if `sd1(#)` or `sd2(#)` is specified, `samps1` assumes a comparison of means; otherwise, it assumes a comparison of proportions. For repeated measurements, `sd1(#)` or `sd2(#)` must be specified. `samps1` is an immediate command; all its arguments are numbers.

## Options

### Main

`onesample` indicates a one-sample test. The default is two-sample.

`sd1(#)` and `sd2(#)` are the standard deviations of population 1 and population 2, respectively. One or both must be specified when doing a comparison of means. When the `onesample` option is used, `sd1(#)` is the standard deviation of the single sample (it can be abbreviated as `sd(#)`). If only one of `sd1(#)` or `sd2(#)` is specified, `samps1` assumes that `sd1() = sd2()`. If neither `sd1(#)` nor `sd2(#)` is specified, `samps1` assumes a test of proportions. For repeated measurements, `sd1(#)` or `sd2(#)` must be specified.

### Options

`alpha(#)` is the significance level of the test. The default is `alpha(0.05)` unless `set level` has been used to reset the default significance level for confidence intervals. If a `set level #lev` command has been issued, the default value is `alpha(1 - #lev/100)`. See [\[R\] level](#).

`power(#)` =  $1 - \beta$  is the power of the test. The default is `power(0.90)`.

`n1(#)` and `n2(#)` are the sizes of sample 1 and sample 2, respectively. One or both must be specified when computing power. If neither `n1(#)` nor `n2(#)` is specified, `samps1` computes sample size. When the `onesample` option is used, `n1(#)` is the size of the single sample (it can be abbreviated as `n(#)`). If only one of `n1(#)` or `n2(#)` is specified, the unspecified one is computed using the formula `ratio = n2()/n1()`.

`ratio(#)` is the ratio of sample sizes for two-sample tests: `ratio = n2()/n1()`. The default is `ratio(1)`.

`pre(#)` specifies the number of baseline measurements (prerandomization) planned in a repeated-measure study. The default is `pre(0)`.

`post(#)` specifies the number of follow-up measurements (postrandomization) planned in a repeated-measure study. The default is `post(1)`.

`nocontinuity` requests power and sample size calculations without the continuity correction for the two-sample test on proportions. If not specified, the continuity correction is used.

`r0(#)` specifies the correlation between baseline measurements in a repeated-measure study. If `r0(#)` is not specified, `samps1` assumes that `r0() = r1()`.

`r1(#)` specifies the correlation between follow-up measurements in a repeated-measure study. For a repeated-measure study, either `r1(#)` or `r01(#)` must be specified. If `r1(#)` is not specified, `samps1` assumes that `r1() = r01()`.

`r01(#)` specifies the correlation between baseline and follow-up measurements in a repeated-measure study. For a repeated-measure study, either `r01(#)` or `r1(#)` must be specified. If `r01(#)` is not specified, `samps1` assumes that `r01() = r1()`.

`onesided` indicates a one-sided test. The default is two-sided.

`method(post | change | ancova | all)` specifies the analysis method to be used with repeated measures. `change` and `ancova` can be used only if baseline measurements are planned. The default is `method(all)`, which means to use all three methods. Each method is described in [Methods and formulas](#).

## Remarks

Remarks are presented under the following headings:

*Studies with one measurement of the outcome*  
*Two-sample test of equality of means*  
*One-sample test of mean*  
*Two-sample test of equality of proportions*  
*One-sample test of proportion*  
*Clinical trials with repeated measures*

## Studies with one measurement of the outcome

For simple studies, where only one measurement of the outcome is planned, `sampsi` computes sample size or power for four types of tests:

1. Two-sample comparison of mean  $\mu_1$  of population 1 with mean  $\mu_2$  of population 2. The null hypothesis is  $\mu_1 = \mu_2$ , and normality is assumed. The postulated values of the means are  $\mu_1 = \#_1$  and  $\mu_2 = \#_2$ , and the postulated standard deviations are `sd1(#)` and `sd2(#)`.
2. One-sample comparison of the mean  $\mu$  of a population with a hypothesized value  $\mu_0$ . The null hypothesis is  $\mu = \mu_0$ , and normality is assumed. The first argument, `#1`, to `sampsi` is  $\mu_0$ . The second argument, `#2`, is the postulated value of  $\mu$ , i.e., the alternative hypothesis is  $\mu = \#_2$ . The postulated standard deviation is `sd1(#)`. To get this test, the `onesample` option must be specified.
3. Two-sample comparison of proportion  $p_1$  with proportion  $p_2$ . The null hypothesis is  $p_1 = p_2$ , and the postulated values are  $p_1 = \#_1$  and  $p_2 = \#_2$ .
4. One-sample comparison of a proportion  $p$  with a hypothesized value  $p_0$ . The null hypothesis is  $p = p_0$ , where  $p_0 = \#_1$ . The alternative hypothesis is  $p = \#_2$ . To get this test, the `onesample` option must be specified.

Examples of these follow.

## Two-sample test of equality of means

### ► Example 1

We are doing a study of the relationship of oral contraceptives (OC) and blood pressure (BP) level for women ages 35–39 ([Rosner 2006](#), 331–333). From a pilot study, it was determined that the mean and standard deviation of BP of OC users were 132.86 and 15.34, respectively. The mean and standard deviation of OC nonusers in the pilot study were found to be 127.44 and 18.23. Because it is easier to find OC nonusers than users, we decide that  $n_2$ , the size of the sample of OC nonusers, should be twice  $n_1$ , the size of the sample of OC users; that is,  $r = n_2/n_1 = 2$ . To compute the sample sizes for  $\alpha = 0.05$  (two-sided) and the power of 0.80, we issue the following command:

```
. samps1 132.86 127.44, p(0.8) r(2) sd1(15.34) sd2(18.23)
Estimated sample size for two-sample comparison of means
Test Ho: m1 = m2, where m1 is the mean in population 1
 and m2 is the mean in population 2
Assumptions:
 alpha = 0.0500 (two-sided)
 power = 0.8000
 m1 = 132.86
 m2 = 127.44
 sd1 = 15.34
 sd2 = 18.23
 n2/n1 = 2.00
Estimated required sample sizes:
 n1 = 108
 n2 = 216
```

We now find out that we have only enough money to study 100 subjects from each group. We can compute the power for  $n_1 = n_2 = 100$  by typing

```
. samps1 132.86 127.44, n1(100) sd1(15.34) sd2(18.23)
Estimated power for two-sample comparison of means
Test Ho: m1 = m2, where m1 is the mean in population 1
 and m2 is the mean in population 2
Assumptions:
 alpha = 0.0500 (two-sided)
 m1 = 132.86
 m2 = 127.44
 sd1 = 15.34
 sd2 = 18.23
sample size n1 = 100
 n2 = 100
 n2/n1 = 1.00
Estimated power:
 power = 0.6236
```

We did not have to specify `n2(#)` or `ratio(#)` because `ratio(1)` is the default.

◀

## One-sample test of mean

### ► Example 2

Suppose that we wish to test the effects of a low-fat diet on serum cholesterol levels. We will measure the difference in cholesterol levels for each subject before and after being on the diet. Because there is only one group of subjects, all on the diet, this is a one-sample test, and we must use the `onesample` option with `samps1`.

Our null hypothesis is that the mean of individual differences in cholesterol level will be zero; i.e.,  $\mu = 0$  mg/100 mL. If the effect of the diet is as large as a mean difference of  $-10$  mg/100 mL, then we wish to have power of 0.95 for rejecting the null hypothesis. Because we expect a reduction in level, we want to use a one-sided test with  $\alpha = 0.025$ . From past studies, we estimate that the standard deviation of the difference in cholesterol levels will be about 20 mg/100 mL. To compute the required sample size, we type

```
. sampsi 0 -10, sd(20) onesam a(0.025) onesided p(0.95)
Estimated sample size for one-sample comparison of mean
to hypothesized value
Test Ho: m = 0, where m is the mean in the population
Assumptions:
 alpha = 0.0250 (one-sided)
 power = 0.9500
 alternative m = -10
 sd = 20
Estimated required sample size:
 n = 52
```

We decide to conduct the study with  $n = 60$  subjects, and we wonder what the power will be at a one-sided significance level of  $\alpha = 0.01$ :

```
. sampsi 0 -10, sd(20) onesam a(0.01) onesided n(60)
Estimated power for one-sample comparison of mean
to hypothesized value
Test Ho: m = 0, where m is the mean in the population
Assumptions:
 alpha = 0.0100 (one-sided)
 alternative m = -10
 sd = 20
 sample size n = 60
Estimated power:
 power = 0.9390
```

◀

## Two-sample test of equality of proportions

### ► Example 3

We want to conduct a survey on people's opinions of the president's performance. Specifically, we want to determine whether members of the president's party have a different opinion from people with another party affiliation. Using past surveys as a guide, we estimate that only 25% of members of the president's party will say that the president is doing a poor job, whereas 40% of members of other parties will rate the president's performance as poor. We compute the required sample sizes for  $\alpha = 0.05$  (two-sided) and the power of 0.90 by typing

```
. sampsi 0.25 0.4
Estimated sample size for two-sample comparison of proportions
Test Ho: p1 = p2, where p1 is the proportion in population 1
 and p2 is the proportion in population 2
Assumptions:
 alpha = 0.0500 (two-sided)
 power = 0.9000
 p1 = 0.2500
 p2 = 0.4000
 n2/n1 = 1.00
Estimated required sample sizes:
 n1 = 216
 n2 = 216
```

To compute the power for a survey with a sample of  $n_1 = 300$  members of the president's party and a sample of  $n_2 = 150$  members of other parties, we type

```
. samps_i 0.25 0.4, n1(300) r(0.5)
Estimated power for two-sample comparison of proportions
Test Ho: p1 = p2, where p1 is the proportion in population 1
 and p2 is the proportion in population 2
Assumptions:
 alpha = 0.0500 (two-sided)
 p1 = 0.2500
 p2 = 0.4000
sample size n1 = 300
 n2 = 150
 n2/n1 = 0.50
Estimated power:
 power = 0.8790
```

◀

## One-sample test of proportion

### ► Example 4

Someone claims that females are more likely than males to study French. Our null hypothesis is that the proportion of female French students is 0.5. We wish to compute the sample size that will give us 80% power to reject the null hypothesis if the true proportion of female French students is 0.75:

```
. samps_i 0.5 0.75, power(0.8) onesample
Estimated sample size for one-sample comparison of proportion
to hypothesized value
Test Ho: p = 0.5000, where p is the proportion in the population
Assumptions:
 alpha = 0.0500 (two-sided)
 power = 0.8000
 alternative p = 0.7500
Estimated required sample size:
 n = 29
```

What is the power if the true proportion of female French students is only 0.6 and the biggest sample of French students we can survey is  $n = 200$ ?

```
. samps_i 0.5 0.6, n(200) onesample
Estimated power for one-sample comparison of proportion
to hypothesized value
Test Ho: p = 0.5000, where p is the proportion in the population
Assumptions:
 alpha = 0.0500 (two-sided)
 alternative p = 0.6000
 sample size n = 200
Estimated power:
 power = 0.8123
```

◀

## □ Technical note

`r(warning)` is saved only for power calculations for one- and two-sample tests on proportions. If sample sizes are not large enough (Tamhane and Dunlop 2000, 300, 307) for sample proportions to be approximately normally distributed, `r(warning)` is set to 1. Otherwise, a note is displayed in the output, and `r(warning)` is set to 0.

□

## Clinical trials with repeated measures

In randomized controlled trials (RCTs), when comparing a standard treatment with an experimental therapy, it is not unusual for the study design to allow for repeated measurements of the outcome. Typically, one or more measurements are taken at baseline immediately before randomization, and additional measurements are taken at regular intervals during follow-up. Depending on the analysis method planned and the correlations between measurements at different time points, there can be a great increase in efficiency (variance reduction) from such designs over a simple study with only one measurement of the outcome.

Frison and Pocock (1992) discuss three methods used in RCTs to compare two treatments by using a continuous outcome measured at different times on each patient.

Posttreatment means (POST) uses the mean of each patient's follow-up measurements as the summary measurement. It compares the two groups by using a simple  $t$  test. This method ignores any baseline measurements.

Mean changes (CHANGE) uses each patient's difference between the mean of the follow-up measurements and the mean of baseline measurements as the summary measurement. It compares the two groups by using a simple  $t$  test.

Analysis of covariance (ANCOVA) uses the mean baseline measurement for each patient as a covariate in a linear model for treatment comparisons of follow-up means.

`method()` specifies which of these three analyses is planned to be used. `samps1` will calculate the decrease in variance of the estimate of treatment effect from the number of measurements at baseline, the number of measurements during follow-up, and the correlations between measurements at different times, and use the calculation to estimate power or sample size, or both.

## ▷ Example 5

We are designing a clinical trial comparing a new medication for the treatment of angina to a placebo. We are planning on performing an exercise stress test on each patient four times during the study: once at time of treatment randomization and three more times at 4, 6, and 8 weeks after randomization. From each test, we will measure the time in seconds from the beginning of the test until the patient is unable to continue because of angina pain. From a previous pilot study, we estimated the means (`sd#s`) for the new drug and the placebo group to be 498 seconds (20.2) and 485 seconds (19.5), respectively, and an overall correlation at follow-up of 0.7. We will analyze these data by comparing each patient's difference between the mean of posttreatment measurements and the mean of baseline measurements, i.e., the change method. To compute the number of subjects needed for allocation to each treatment group for  $\alpha = 0.05$  (two-sided) and power of 90%, we issue the following command:



```
. sampsī 498 485, sd1(20.2) sd2(19.5) method(change) pre(1) post(3) r1(.7)
```

Estimated sample size for two samples with repeated measures

Assumptions:

```

alpha = 0.0500 (two-sided)
power = 0.9000
m1 = 498
m2 = 485
sd1 = 20.2
sd2 = 19.5
n2/n1 = 1.00
number of follow-up measurements = 3
correlation between follow-up measurements = 0.700
number of baseline measurements = 1
correlation between baseline & follow-up = 0.700
```

Method: CHANGE

```

relative efficiency = 2.500
adjustment to sd = 0.632
adjusted sd1 = 12.776
adjusted sd2 = 12.333
```

Estimated required sample sizes:

```

n1 = 20
n2 = 20
```

The output from `sampsī` for repeated measurements includes the specified parameters used to estimate the sample sizes or power, the relative efficiency of the design, and the adjustment to the standard deviation. These last two are the inverse and the square root of the calculated improvement in the variance compared with a similar study where only one measurement is planned.

We see that we need to allocate 20 subjects to each treatment group. Assume that we have funds to enroll only 30 patients into our study. If we randomly assigned 15 patients to each treatment group, what would be the expected power of our study, assuming that all other parameters remain the same?

```
. sampsī 498 485, sd1(20.2) sd2(19.5) meth(change) pre(1) post(3) r1(.7) n1(15)
> n2(15)
```

Estimated power for two samples with repeated measures

Assumptions:

```

alpha = 0.0500 (two-sided)
m1 = 498
m2 = 485
sd1 = 20.2
sd2 = 19.5
sample size n1 = 15
n2 = 15
n2/n1 = 1.00
number of follow-up measurements = 3
correlation between follow-up measurements = 0.700
number of baseline measurements = 1
correlation between baseline & follow-up = 0.700
```

Method: CHANGE

```

relative efficiency = 2.500
adjustment to sd = 0.632
adjusted sd1 = 12.776
adjusted sd2 = 12.333
```

Estimated power:

```
power = 0.809
```

If we enroll 30 patients into our study instead of the recommended 40, the power of the study decreases from 90% to approximately 81%.

## Saved results

**samps** saves the following in `r()`:

Scalars

|                         |                                                |
|-------------------------|------------------------------------------------|
| <code>r(N_1)</code>     | sample size $n_1$                              |
| <code>r(N_2)</code>     | sample size $n_2$                              |
| <code>r(power)</code>   | power                                          |
| <code>r(adj)</code>     | adjustment to the SE                           |
| <code>r(warning)</code> | 0 if assumptions are satisfied and 1 otherwise |

## Methods and formulas

**samps** is implemented as an ado-file.

In the following formulas,  $\alpha$  is the significance level,  $1 - \beta$  is the power,  $z_{1-\alpha/2}$  is the  $(1 - \alpha/2)$  quantile of the normal distribution, and  $r = n_2/n_1$  is the ratio of sample sizes. The formulas below are for two-sided tests. The formulas for one-sided tests can be obtained by replacing  $z_{1-\alpha/2}$  with  $z_{1-\alpha}$ .

1. The required sample sizes for a two-sample test of equality of means (assuming normality) are

$$n_1 = \frac{(\sigma_1^2 + \sigma_2^2/r)(z_{1-\alpha/2} + z_{1-\beta})^2}{(\mu_1 - \mu_2)^2}$$

and  $n_2 = rn_1$  (Rosner 2006, 332).

2. For a one-sample test of a mean where the null hypothesis is  $\mu = \mu_0$  and the alternative hypothesis is  $\mu = \mu_A$ , the required sample size (assuming normality) is

$$n = \left\{ \frac{(z_{1-\alpha/2} + z_{1-\beta})\sigma}{\mu_A - \mu_0} \right\}^2$$

(Pagano and Gauvreau 2000, 247–248).

3. The required sample sizes for a two-sample test of equality of proportions (using a normal approximation with a continuity correction) are

$$n_1 = \frac{n'}{4} \left[ 1 + \left\{ 1 + \frac{2(r+1)}{n'r |p_1 - p_2|} \right\}^{1/2} \right]^2$$

$$n_2 = rn_1$$

where

$$n' = \frac{\left[ z_{1-\alpha/2} \{ (r+1)\bar{p}\bar{q} \}^{1/2} + z_{1-\beta} (rp_1q_1 + p_2q_2)^{1/2} \right]^2}{r(p_1 - p_2)^2}$$

and  $\bar{p} = (p_1 + rp_2)/(r+1)$  and  $\bar{q} = 1 - \bar{p}$  (Fleiss, Levin, and Paik 2003, 76).

Without a continuity correction, the sample sizes are

$$n_1 = n'$$

$$n_2 = rn_1$$

where  $n'$  is defined above.

4. For a one-sample test of proportion where the null hypothesis is  $p = p_0$  and the alternative hypothesis is  $p = p_A$ , the required sample size (using a normal approximation) is

$$n = \left[ \frac{z_{1-\alpha/2} \{p_0(1-p_0)\}^{1/2} + z_{1-\beta} \{p_A(1-p_A)\}^{1/2}}{p_A - p_0} \right]^2$$

(Pagano and Gauvreau 2000, 332).

5. For repeated measurements, Frison and Pocock (1992) discuss three methods for use in randomized clinical trials to compare two treatments using a continuous outcome measured at different times on each patient. Each uses the average of baseline measurements,  $\bar{x}_0$ , and follow-up measurements,  $\bar{x}_1$ :

POST outcome is  $\bar{x}_1$ , where the analysis is by simple  $t$  test.

CHANGE outcome is  $\bar{x}_1 - \bar{x}_0$ , where the analysis is by simple  $t$  test.

ANCOVA outcome is  $\bar{x}_1 - \beta \bar{x}_0$ , where the  $\beta$  is estimated by analysis of covariance, correcting for the average at baseline.

ANCOVA will always be the most efficient of the three approaches.  $\beta$  is set so that  $\beta \bar{x}_0$  accounts for the largest possible variation of  $\bar{x}_1$ .

For a study with one measurement each at baseline and follow-up, CHANGE will be more efficient than POST, provided that the correlation between measurements at baseline and measurements at follow-up is more than 0.5. POST ignores all baseline measurements, which tends to make it unpopular. CHANGE is the method most commonly used. With more than one baseline measurement, CHANGE and ANCOVA tend to produce similar sample sizes and power.

The improvements in variance of the estimate of treatment effect over a study with only one measurement depend on the number of measurements  $p$  at baseline; the number of measurements during follow-up; and the correlations between measurements at baseline  $\bar{\rho}_{\text{pre}}$ , between measurements at follow-up  $\bar{\rho}_{\text{post}}$ , and between measurements at baseline and measurements at follow-up  $\bar{\rho}_{\text{mix}}$ . The improvements in variance for the POST method are given by

$$\frac{1 + (r - 1)\bar{\rho}_{\text{post}}}{r}$$

for the CHANGE method by

$$\frac{1 + (r - 1)\bar{\rho}_{\text{post}}}{r} + \frac{1 + (p - 1)\bar{\rho}_{\text{pre}}}{p} - 2\bar{\rho}_{\text{mix}}$$

and for the ANCOVA method by

$$\frac{1 + (r - 1)\bar{\rho}_{\text{post}}}{r} - \frac{\bar{\rho}_{\text{mix}}^2 p}{1 + (p - 1)\bar{\rho}_{\text{pre}}}$$

Often the three correlations are assumed equal. In data from several trials, Frison and Pocock found that  $\bar{\rho}_{\text{pre}}$  and  $\bar{\rho}_{\text{post}}$  typically had values around 0.7, whereas  $\bar{\rho}_{\text{mix}}$  was nearer 0.5. This finding is consistent with the common finding that measurements closer in time are more strongly correlated.

Power calculations are based on estimates of one variance at all time points.

## Acknowledgments

`samps` is based on the `sampsiz` command written by Joseph Hilbe of Arizona State University (Hilbe 1993). Paul Seed of Maternal and Fetal Research Unit, St. Thomas' Hospital (Seed 1997, 1998), expanded the command to allow for repeated measurements.

## References

- Fleiss, J. L., B. Levin, and M. C. Paik. 2003. *Statistical Methods for Rates and Proportions*. 3rd ed. New York: Wiley.
- Frison, L., and S. J. Pocock. 1992. Repeated measures in clinical trials: Analysis using mean summary statistics and its implications for design. *Statistics in Medicine* 11: 1685–1704.
- Hilbe, J. M. 1993. `sg15`: Sample size determination for means and proportions. *Stata Technical Bulletin* 11: 17–20. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 145–149. College Station, TX: Stata Press.
- Pagano, M., and K. Gauvreau. 2000. *Principles of Biostatistics*. 2nd ed. Belmont, CA: Duxbury.
- Rosner, B. 2006. *Fundamentals of Biostatistics*. 6th ed. Belmont, CA: Duxbury.
- Royston, P., and A. Babiker. 2002. A menu-driven facility for complex sample size calculation in randomized controlled trials with a survival or a binary outcome. *Stata Journal* 2: 151–163.
- Seed, P. T. 1997. `sbe18`: Sample size calculations for clinical trials with repeated measures data. *Stata Technical Bulletin* 40: 16–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 121–125. College Station, TX: Stata Press.
- . 1998. `sbe18.1`: Update of `samps`. *Stata Technical Bulletin* 45: 21. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, p. 84. College Station, TX: Stata Press.
- Tamhane, A. C., and D. D. Dunlop. 2000. *Statistics and Data Analysis: From Elementary to Intermediate*. Upper Saddle River, NJ: Prentice–Hall.

## Also see

[ST] `stpower` — Sample-size, power, and effect-size determination for survival analysis

## Syntax

*List results stored in `r()`*

`return list`

*List results stored in `e()`*

`ereturn list`

*List results stored in `s()`*

`sreturn list`

## Description

Results of calculations are saved by many Stata commands so that they can be easily accessed and substituted into later commands.

`return list` lists results stored in `r()`.

`ereturn list` lists results stored in `e()`.

`sreturn list` lists results stored in `s()`.

This entry discusses using saved results. Programmers wishing to save results should see [\[P\] `return`](#) and [\[P\] `ereturn`](#).

## Remarks

Stata commands are classified as being

|         |                                                                                        |
|---------|----------------------------------------------------------------------------------------|
| r-class | general commands that save results in <code>r()</code>                                 |
| e-class | estimation commands that save results in <code>e()</code>                              |
| s-class | parsing commands that save results in <code>s()</code>                                 |
| n-class | commands that do not save in <code>r()</code> , <code>e()</code> , or <code>s()</code> |

There is also a c-class, `c()`, containing the values of system parameters and settings, along with certain constants, such as the value of pi; see [\[P\] `return`](#). A program, however, cannot be c-class.

You can look at the *Saved results* section of the manual entry of a command to determine whether it is r-, e-, s-, or n-class, but it is easy enough to guess.

Commands producing statistical results are either r-class or e-class. They are e-class if they present estimation results and r-class otherwise. s-class is a class used by programmers and is primarily used in subprograms performing parsing. n-class commands explicitly state where the result is to go. For instance, `generate` and `replace` are n-class because their syntax is `generate varname = ...` and `replace varname = ...`.

After executing a command, you can type `return list`, `ereturn list`, or `sreturn list` to see what has been saved:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. summarize mpg
```

| Variable | Obs | Mean    | Std. Dev. | Min | Max |
|----------|-----|---------|-----------|-----|-----|
| mpg      | 74  | 21.2973 | 5.785503  | 12  | 41  |

```
. return list
scalars:
 r(N) = 74
 r(sum_w) = 74
 r(mean) = 21.2972972972973
 r(Var) = 33.47204738985561
 r(sd) = 5.785503209735141
 r(min) = 12
 r(max) = 41
 r(sum) = 1576
```

After `summarize`, you can use `r(N)`, `r(mean)`, `r(sd)`, etc., in expressions:

```
. gen double mpgstd = (mpg-r(mean))/r(sd)
. summarize mpgstd
```

| Variable | Obs | Mean      | Std. Dev. | Min       | Max     |
|----------|-----|-----------|-----------|-----------|---------|
| mpgstd   | 74  | -1.64e-16 | 1         | -1.606999 | 3.40553 |

Be careful to use results stored in `r()` soon because they will be replaced the next time you execute another `r-class` command. For instance, although `r(mean)` was 21.3 (approximately) after `summarize mpg`, it is `-1e-16` now because you just ran `summarize` again.

`e-class` is really no different from `r-class`, except for where results are stored and that, when an estimation command stores results, it tends to store a lot of them:

```
. regress mpg weight displ
(output omitted)

. ereturn list
scalars:
 e(N) = 74
 e(df_m) = 2
 e(df_r) = 71
 e(F) = 66.78504752026517
 e(r2) = .6529306984682528
 e(rmse) = 3.45606176570828
 e(mss) = 1595.409691543724
 e(rss) = 848.0497679157352
 e(r2_a) = .643154098425105
 e(ll) = -195.2397979466294
 e(ll_0) = -234.3943376482347
 e(rank) = 3

macros:
 e(cmdline) : "regress mpg weight displ"
 e(title) : "Linear regression"
 e(marginsok) : "XB default"
 e(vce) : "ols"
 e(depvar) : "mpg"
 e(cmd) : "regress"
 e(properties) : "b V"
 e(predict) : "regres_p"
 e(model) : "ols"
 e(estat_cmd) : "regress_estat"
```

```
matrices:
 e(b) : 1 x 3
 e(V) : 3 x 3

functions:
 e(sample)
```

These e-class results will stick around until you run another estimation command. Typing `return list` and `ereturn list` is the easy way to find out what a command stores.

Both r- and e-class results come in four flavors: scalars, macros, matrices, and functions. (s-class results come in only one flavor—macros—and as earlier noted, s-class is used solely by programmers, so ignore it.)

Scalars are just that—numbers by any other name. You can subsequently refer to `r(mean)` or `e(rmse)` in numeric expressions and obtain the result to full precision.

Macros are strings. For instance, `e(depvar)` contains “mpg”. You can refer to it, too, in subsequent expressions, but really that would be of most use to programmers, who will refer to it using constructs like “`‘e(depvar)’`”. In any case, macros are macros, and you obtain their contents just as you would a local macro, by enclosing their name in single quotes. The name here is the full name, so `‘e(depvar)’` is mpg.

Matrices are matrices, and all estimation commands store `e(b)` and `e(V)` containing the coefficient vector and variance–covariance matrix of the estimates (VCE).

Functions are saved by e-class commands only, and the only function existing is `e(sample)`. `e(sample)` evaluates to 1 (meaning true) if the observation was used in the previous estimation and to 0 (meaning false) otherwise.

## □ Technical note

Say that some command set `r(scalar)` and `r(macro)`, the first being stored as a scalar and the second as a macro. In theory, in subsequent use you are supposed to refer to `r(scalar)` and `‘r(macro)’`. In fact, however, you can refer to either one with or without quotes, so you could refer to `‘r(scalar)’` and `r(macro)`. Programmers sometimes do this.

When you refer to `r(scalar)`, you are referring to the full double-precision saved result. Think of `r(scalar)` without quotes as a function returning the value of the saved result `scalar`. When you refer to `r(scalar)` in quotes, Stata understands `‘r(scalar)’` to mean “substitute the printed result of evaluating `r(scalar)`”. Pretend that `r(scalar)` equals the number 23. Then `‘r(scalar)’` is 23, the character 2 followed by 3.

Referring to `r(scalar)` in quotes is sometimes useful. Say that you want to use the immediate command `ci` with `r(scalar)`. The immediate command `ci` requires its arguments to be numbers—numeric literals in programmer’s jargon—and it will not take an expression. Thus you could not type `‘ci r(scalar) ...’`. You could, however, type `‘ci ‘r(scalar)’ ...’` because `‘r(scalar)’` is just a numeric literal.

For `r(macro)`, you are supposed to refer to it in quotes: `‘r(macro)’`. If, however, you omit the quotes in an expression context, Stata evaluates the macro and then pretends that it is the result of function-returning-string. There are side effects of this, the most important being that the result is trimmed to 80 characters.

Referring to `r(macro)` without quotes is never a good idea; the feature was included merely for completeness.

You can even refer to `r(matrix)` in quotes (assume that `r(matrix)` is a matrix). `'r(matrix)'` does not result in the matrix being substituted; it returns the word `matrix`. Programmers sometimes find that useful.



## References

- Jann, B. 2005. [Making regression tables from stored estimates](#). *Stata Journal* 5: 288–308.
- . 2007. [Making regression tables simplified](#). *Stata Journal* 7: 227–244.

## Also see

- [P] [ereturn](#) — Post the estimation results
- [P] [return](#) — Return saved results
- [U] [18.8 Accessing results calculated by other programs](#)
- [U] [18.9 Accessing results calculated by estimation commands](#)



Syntax

```
scoibit depvar [indepvars] [if] [in] [weight] [, options]
```

| <i>options</i>                            | description                                                                                                                             |
|-------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Model                                     |                                                                                                                                         |
| <u>noconstant</u>                         | suppress constant term                                                                                                                  |
| <u>offset</u> ( <i>varname</i> )          | include <i>varname</i> in model with coefficient constrained to 1                                                                       |
| <u>asis</u>                               | retain perfect predictor variables                                                                                                      |
| <u>constraints</u> ( <i>constraints</i> ) | apply specified linear constraints                                                                                                      |
| <u>collinear</u>                          | keep collinear variables                                                                                                                |
| SE/Robust                                 |                                                                                                                                         |
| <u>vce</u> ( <i>vcetype</i> )             | <i>vcetype</i> may be <u>oim</u> , <u>robust</u> , <u>cluster</u> <i>clustvar</i> , <u>opg</u> , <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting                                 |                                                                                                                                         |
| <u>level</u> (#)                          | set confidence level; default is <u>level</u> (95)                                                                                      |
| <u>or</u>                                 | report odds ratios                                                                                                                      |
| <u>nocnsreport</u>                        | do not display constraints                                                                                                              |
| <u>display_options</u>                    | control spacing and display of omitted variables and base and empty cells                                                               |
| Maximization                              |                                                                                                                                         |
| <u>maximize_options</u>                   | control the maximization process                                                                                                        |
| <sup>†</sup> <u>coeflegend</u>            | display coefficients' legend instead of coefficient table                                                                               |

<sup>†</sup>coeflegend does not appear in the dialog box.  
*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.  
bootstrap, by, jackknife, nestreg, rolling, statsby, stepwise, and svy are allowed; see [U] 11.1.10 Prefix commands.  
Weights are not allowed with the bootstrap prefix.  
vce() and weights are not allowed with the svy prefix.  
fweights, iweights, and pweights are allowed; see [U] 11.1.6 weight.  
See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Binary outcomes > Skewed logit regression

## Description

`scobit` fits a maximum-likelihood skewed logit model.

See [\[R\] logistic](#) for a list of related estimation commands.

## Options

### Model

`noconstant`, `offset(varname)`, `constraints(constraints)`, `collinear`; see [\[R\] estimation options](#).

`asis` forces retention of perfect predictor variables and their associated perfectly predicted observations and may produce instabilities in maximization; see [\[R\] probit](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [\[R\] vce\\_option](#).

### Reporting

`level(#)`; see [\[R\] estimation options](#).

`or` reports the estimated coefficients transformed to odds ratios, i.e.,  $e^b$  rather than  $b$ . Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated. `or` may be specified at estimation or when replaying previously estimated results.

`nocnsreport`; see [\[R\] estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [\[R\] estimation options](#).

### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [\[R\] maximize](#).

Setting the optimization type to `technique(bhhh)` resets the default `vcetype` to `vce(opg)`.

The following option is available with `scobit` but is not shown in the dialog box:

`coeflegend`; see [\[R\] estimation options](#).

## Remarks

Remarks are presented under the following headings:

*Skewed logistic model*

*Robust standard errors*

## Skewed logistic model

`scobit` fits maximum likelihood models with dichotomous dependent variables coded as 0/1 (or, more precisely, coded as 0 and not 0).

### ► Example 1

We have data on the make, weight, and mileage rating of 22 foreign and 52 domestic automobiles. We wish to fit a model explaining whether a car is foreign based on its mileage. Here is an overview of our data:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
```

```
. keep make mpg weight foreign
```

```
. describe
```

```
Contains data from http://www.stata-press.com/data/r11/auto.dta
 obs: 74 1978 Automobile Data
 vars: 4 13 Apr 2009 17:45
 size: 1,998 (99.9% of memory free) (_dta has notes)
```

| variable name | storage<br>type | display<br>format | value<br>label | variable label |
|---------------|-----------------|-------------------|----------------|----------------|
| make          | str18           | %-18s             |                | Make and Model |
| mpg           | int             | %8.0g             |                | Mileage (mpg)  |
| weight        | int             | %8.0gc            |                | Weight (lbs.)  |
| foreign       | byte            | %8.0g             | origin         | Car type       |

```
Sorted by: foreign
```

```
Note: dataset has changed since last saved
```

```
. inspect foreign
```

```
foreign: Car type
```

|     |          | Number of Observations |          |             |
|-----|----------|------------------------|----------|-------------|
|     |          | Total                  | Integers | Nonintegers |
| #   | Negative | -                      | -        | -           |
| #   | Zero     | 52                     | 52       | -           |
| #   | Positive | 22                     | 22       | -           |
| #   |          |                        |          |             |
| # # | Total    | 74                     | 74       | -           |
| # # | Missing  | -                      |          |             |
|     |          | 74                     |          |             |

0 1

(2 unique values)

```
foreign is labeled and all values are documented in the label.
```

The variable `foreign` takes on two unique values, 0 and 1. The value 0 denotes a domestic car, and 1 denotes a foreign car.

The model that we wish to fit is

$$\Pr(\text{foreign} = 1) = F(\beta_0 + \beta_1 \text{mpg})$$

where  $F(z) = 1 - 1/\{1 + \exp(z)\}^\alpha$ .

To fit this model, we type

```
. scobit foreign mpg
Fitting logistic model:
Iteration 0: log likelihood = -45.03321
Iteration 1: log likelihood = -39.380959
Iteration 2: log likelihood = -39.288802
Iteration 3: log likelihood = -39.28864
Fitting full model:
Iteration 0: log likelihood = -39.28864
Iteration 1: log likelihood = -39.286393
Iteration 2: log likelihood = -39.284415
Iteration 3: log likelihood = -39.284234
Iteration 4: log likelihood = -39.284197
Iteration 5: log likelihood = -39.284196
Skewed logistic regression Number of obs = 74
 Zero outcomes = 52
Log likelihood = -39.2842 Nonzero outcomes = 22
```

| foreign  | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| mpg      | .1813879  | .2407362  | 0.75  | 0.451 | -.2904463            | .6532222  |
| _cons    | -4.274883 | 1.399305  | -3.06 | 0.002 | -7.017471            | -1.532295 |
| /lnalpha | -.4450405 | 3.879885  | -0.11 | 0.909 | -8.049476            | 7.159395  |
| alpha    | .6407983  | 2.486224  |       |       | .0003193             | 1286.133  |

Likelihood-ratio test of alpha=1: chi2(1) = 0.01 Prob > chi2 = 0.9249  
note: likelihood-ratio tests are recommended for inference with scobit models.

We find that cars yielding better gas mileage are less likely to be foreign. The likelihood-ratio test at the bottom of the output indicates that the model is not significantly different from a logit model. Therefore, we should use the more parsimonious model.



□ Technical note

Stata interprets a value of 0 as a negative outcome (failure) and treats all other values (except missing) as positive outcomes (successes). Thus if the dependent variable takes on the values 0 and 1, then 0 is interpreted as failure and 1 as success. If the dependent variable takes on the values 0, 1, and 2, then 0 is still interpreted as failure, but both 1 and 2 are treated as successes.

Formally, when we type `scobit y x`, Stata fits the model

$$\Pr(y_j \neq 0 \mid \mathbf{x}_j) = 1 - 1 / \left\{ 1 + \exp(\mathbf{x}_j \boldsymbol{\beta}) \right\}^\alpha$$



Robust standard errors

If you specify the `vce(robust)` option, `scobit` reports robust standard errors as described in [U] 20.16 Obtaining robust variance estimates. For the model of `foreign` on `mpg`, the robust calculation increases the standard error of the coefficient on `mpg` by around 25%:

```
. scobit foreign mpg, vce(robust) nolog
```

```
Skewed logistic regression Number of obs = 74
 Zero outcomes = 52
Log pseudolikelihood = -39.2842 Nonzero outcomes = 22
```

| foreign  | Coef.     | Robust<br>Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|---------------------|-------|-------|----------------------|-----------|
| mpg      | .1813879  | .3028487            | 0.60  | 0.549 | -.4121847            | .7749606  |
| _cons    | -4.274883 | 1.335521            | -3.20 | 0.001 | -6.892455            | -1.657311 |
| /lnalpha | -.4450405 | 4.71561             | -0.09 | 0.925 | -9.687466            | 8.797385  |
| alpha    | .6407983  | 3.021755            |       |       | .0000621             | 6616.919  |

```
Likelihood-ratio test of alpha=1: chi2(1) = 0.01 Prob > chi2 = 0.9249
```

```
note: likelihood-ratio tests are recommended for inference with scobit models.
```

Without `vce(robust)`, the standard error for the coefficient on `mpg` was reported to be 0.241, with a resulting confidence interval of  $[-0.29, 0.65]$ .

Specifying the `vce(cluster clustvar)` option relaxes the independence assumption required by the skewed logit estimator to being just independence between clusters. To demonstrate this, we will switch to a different dataset.

## ► Example 2

We are studying the unionization of women in the United States and have a dataset with 26,200 observations on 4,434 women between 1970 and 1988. For our purposes, we will use the variables `age` (the women were 14–26 in 1968 and the data thus span the age range of 16–46), `grade` (years of schooling completed, ranging from 0 to 18), `not_smsa` (28% of the person-time was spent living outside an SMSA—standard metropolitan statistical area), `south` (41% of the person-time was in the South), and `year`. Each of these variables is included in the regression as a covariate along with the interaction between `south` and `year`. This interaction, along with the `south` and `year` variables, is specified in the `scobit` command using factor-variables notation, `south##c.year`. We also have variable `union`. Overall, 22% of the person-time is marked as time under union membership and 44% of these women have belonged to a union.

We fit the following model, ignoring that women are observed an average of 5.9 times each in these data:

(Continued on next page)

```
. use http://www.stata-press.com/data/r11/union, clear
(NLS Women 14-24 in 1968)

. scobit union age grade not_smsa south##c.year, nrtol(1e-3)
(output omitted)

Skewed logistic regression Number of obs = 26200
 Zero outcomes = 20389
Log likelihood = -13540.61 Nonzero outcomes = 5811
```

| union        | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|--------------|-----------|-----------|-------|-------|----------------------|-----------|
| age          | .0185365  | .0043615  | 4.25  | 0.000 | .0099881             | .0270849  |
| grade        | .0452803  | .0057124  | 7.93  | 0.000 | .0340842             | .0564764  |
| not_smsa     | -.1886849 | .0317802  | -5.94 | 0.000 | -.250973             | -.1263968 |
| 1.south      | -1.422381 | .3949298  | -3.60 | 0.000 | -2.196429            | -.6483327 |
| year         | -.0133017 | .0049575  | -2.68 | 0.007 | -.0230182            | -.0035853 |
| south#c.year |           |           |       |       |                      |           |
| 1            | .0105663  | .0049233  | 2.15  | 0.032 | .0009168             | .0202158  |
| _cons        | -10.19247 | 63.69012  | -0.16 | 0.873 | -135.0228            | 114.6379  |
| /lnalpha     | 8.972796  | 63.68823  | 0.14  | 0.888 | -115.8538            | 133.7994  |
| alpha        | 7885.615  | 502220.9  |       |       | 4.85e-51             | 1.28e+58  |

Likelihood-ratio test of alpha=1:    chi2(1) =        3.76    Prob > chi2 = 0.0524  
Note: likelihood-ratio tests are recommended for inference with scobit models.

The reported standard errors in this model are probably meaningless. Women are observed repeatedly, so the observations are not independent. Looking at the coefficients, we find a large southern effect against unionization and a different time trend for the south. The `vce(cluster clustvar)` option provides a way to fit this model and obtains correct standard errors:

```
. scobit union age grade not_smsa south##c.year, vce(cluster id) nrtol(1e-3)
(output omitted)

Skewed logistic regression Number of obs = 26200
 Zero outcomes = 20389
Log pseudolikelihood = -13540.61 Nonzero outcomes = 5811
 (Std. Err. adjusted for 4434 clusters in idcode)
```

| union        | Coef.     | Robust Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|--------------|-----------|------------------|--------|-------|----------------------|-----------|
| age          | .0185365  | .0084867         | 2.18   | 0.029 | .0019029             | .0351701  |
| grade        | .0452803  | .0125764         | 3.60   | 0.000 | .0206311             | .0699296  |
| not_smsa     | -.1886849 | .0642035         | -2.94  | 0.003 | -.3145214            | -.0628484 |
| 1.south      | -1.422381 | .5064916         | -2.81  | 0.005 | -2.415086            | -.4296756 |
| year         | -.0133017 | .0090621         | -1.47  | 0.142 | -.0310632            | .0044597  |
| south#c.year |           |                  |        |       |                      |           |
| 1            | .0105663  | .0063172         | 1.67   | 0.094 | -.0018152            | .0229478  |
| _cons        | -10.19247 | .9433186         | -10.80 | 0.000 | -12.04134            | -8.343599 |
| /lnalpha     | 8.972796  | .7451728         | 12.04  | 0.000 | 7.512284             | 10.43331  |
| alpha        | 7885.615  | 5876.146         |        |       | 1830.389             | 33972.52  |

`scobit`, `vce(cluster clustvar)` is robust to assumptions about within-cluster correlation. That is, it inefficiently sums within cluster for the standard error calculation rather than attempting to exploit what might be assumed about the within-cluster correlation (as do the `xtgee` population-averaged models; see [XT] `xtgee`).



#### □ Technical note

The `scobit` model can be difficult to fit because of the functional form. Often it requires many iterations, or the optimizer prints out warning and informative messages during the optimization. For example, without the `nrtol(1e-3)` option, the model using the `union` dataset will not converge. See [R] `maximize` for details about the optimizer.



#### □ Technical note

The main reason for using `scobit` rather than `logit` is that the effects of the regressors on the probability of success are not constrained to be the largest when the probability is 0.5. Rather, the independent variables might show their largest impact when the probability of success is 0.3 or 0.6. This added flexibility results because the `scobit` function, unlike the `logit` function, can be skewed and is not constrained to be mirror symmetric about the 0.5 probability of success.

As Nagler (1994) pointed out, the point of maximum impact is constrained under the `scobit` model to fall within the interval  $(0, 1 - e^{(-1)})$  or approximately (0, 0.63). Achen (2002) notes that if we believe the maximum impact to be outside that range, we can instead estimate the “power logit” model by simply reversing the 0s and 1s of our outcome variable and estimating a `scobit` model on failure, rather than success. We would need to reverse the signs of the coefficients if we wanted to interpret them in terms of impact on success, or we could leave them as they are and interpret them in terms of impact on failure. The important thing to remember is that the `scobit` model, unlike the `logit` model, is not invariant to the choice of which result is assigned to success.



*(Continued on next page)*

## Saved results

`scobit` saves the following in `e()`:

### Scalars

|                           |                                                |
|---------------------------|------------------------------------------------|
| <code>e(N)</code>         | number of observations                         |
| <code>e(k)</code>         | number of parameters                           |
| <code>e(k_eq)</code>      | number of equations                            |
| <code>e(k_aux)</code>     | number of auxiliary parameters                 |
| <code>e(k_dv)</code>      | number of dependent variables                  |
| <code>e(k_autoCns)</code> | number of base, empty, and omitted constraints |
| <code>e(ll)</code>        | log likelihood                                 |
| <code>e(ll_c)</code>      | log likelihood, comparison model               |
| <code>e(N_f)</code>       | number of failures (zero outcomes)             |
| <code>e(N_s)</code>       | number of successes (nonzero outcomes)         |
| <code>e(alpha)</code>     | alpha                                          |
| <code>e(N_clust)</code>   | number of clusters                             |
| <code>e(chi2)</code>      | $\chi^2$                                       |
| <code>e(chi2_c)</code>    | $\chi^2$ for comparison test                   |
| <code>e(p)</code>         | significance                                   |
| <code>e(rank)</code>      | rank of <code>e(V)</code>                      |
| <code>e(ic)</code>        | number of iterations                           |
| <code>e(rc)</code>        | return code                                    |
| <code>e(converged)</code> | 1 if converged, 0 otherwise                    |

### Macros

|                                 |                                                                                 |
|---------------------------------|---------------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>scobit</code>                                                             |
| <code>e(cmdline)</code>         | command as typed                                                                |
| <code>e(depvar)</code>          | name of dependent variable                                                      |
| <code>e(wtype)</code>           | weight type                                                                     |
| <code>e(wexp)</code>            | weight expression                                                               |
| <code>e(title)</code>           | title in estimation output                                                      |
| <code>e(clustvar)</code>        | name of cluster variable                                                        |
| <code>e(offset)</code>          | offset                                                                          |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                         |
| <code>e(chi2_ct)</code>         | Wald or LR; type of model $\chi^2$ test corresponding to <code>e(chi2_c)</code> |
| <code>e(vce)</code>             | <i>vctype</i> specified in <code>vce()</code>                                   |
| <code>e(vctype)</code>          | title used to label Std. Err.                                                   |
| <code>e(diparm#)</code>         | display transformed parameter #                                                 |
| <code>e(opt)</code>             | type of optimization                                                            |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization        |
| <code>e(ml_method)</code>       | type of ml method                                                               |
| <code>e(user)</code>            | name of likelihood-evaluator program                                            |
| <code>e(technique)</code>       | maximization technique                                                          |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular                     |
| <code>e(crittype)</code>        | optimization criterion                                                          |
| <code>e(properties)</code>      | b V                                                                             |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                                  |
| <code>e(footnote)</code>        | program used to implement the footnote display                                  |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>                  |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>                  |



## Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(gradient)</code>     | gradient vector                              |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`scobit` is implemented as an ado-file.

Skewed logit analysis is an alternative to logit that relaxes the assumption that individuals with initial probability of 0.5 are most sensitive to changes in independent variables.

The log-likelihood function for skewed logit is

$$\ln L = \sum_{j \in S} w_j \ln F(\mathbf{x}_j \mathbf{b}) + \sum_{j \notin S} w_j \ln \{1 - F(\mathbf{x}_j \mathbf{b})\}$$

where  $S$  is the set of all observations  $j$  such that  $y_j \neq 0$ ,  $F(z) = 1 - 1/\{1 + \exp(z)\}^\alpha$ , and  $w_j$  denotes the optional weights.  $\ln L$  is maximized as described in [R] [maximize](#).

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

`scobit` also supports estimation with survey data. For details on VCEs with survey data, see [SVY] [variance estimation](#).

## References

- Achen, C. H. 2002. Toward a new political methodology: Microfoundations and ART. *Annual Review of Political Science* 5: 423–450.
- Nagler, J. 1994. Scobit: An alternative estimator to logit and probit. *American Journal of Political Science* 38: 230–255.

## Also see

- [R] [scobit postestimation](#) — Postestimation tools for `scobit`
- [R] [cloglog](#) — Complementary log-log regression
- [R] [glm](#) — Generalized linear models
- [R] [logistic](#) — Logistic regression, reporting odds ratios
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `scobit`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic nooffset]

predict [type] { stub* | newvarreg newvarlnalpha } [if] [in] , scores
```

| statistic         | description                                    |
|-------------------|------------------------------------------------|
| <code>pr</code>   | probability of a positive outcome; the default |
| <code>xb</code>   | $\mathbf{x}_j\mathbf{b}$ , linear prediction   |
| <code>stdp</code> | standard error of the linear prediction        |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

`pr`, the default, calculates the probability of a positive outcome.

`xb` calculates the linear prediction.

`stdp` calculates the standard error of the linear prediction.

`nooffset` is relevant only if you specified `offset(varname)` for `scobit`. It modifies the calculations made by `predict` so that they ignore the offset variable; the linear prediction is treated as  $\mathbf{x}_j\mathbf{b}$  rather than as  $\mathbf{x}_j\mathbf{b} + \text{offset}_j$ .

`scores` calculates equation-level score variables.

The first new variable will contain  $\partial \ln L / \partial (\mathbf{x}_j\boldsymbol{\beta})$ .

The second new variable will contain  $\partial \ln L / \partial \ln \alpha$ .

## Remarks

Once you have fit a model, you can obtain the predicted probabilities by using the `predict` command for both the estimation sample and other samples; see [U] 20 Estimation and postestimation commands and [R] `predict`. Here we will make only a few additional comments.

`predict` without arguments calculates the predicted probability of a positive outcome. With the `xb` option, it calculates the linear combination  $\mathbf{x}_j\mathbf{b}$ , where  $\mathbf{x}_j$  are the independent variables in the  $j$ th observation and  $\mathbf{b}$  is the estimated parameter vector.

With the `stdp` option, `predict` calculates the standard error of the prediction, which is *not* adjusted for replicated covariate patterns in the data.

### ► Example 1

In [example 1](#) of [R] `scobit`, we fit the model `scobit foreign mpg`. To obtain predicted probabilities, we type

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. keep make mpg weight foreign
. scobit foreign mpg
(output omitted)
. predict p
(option pr assumed; Pr(foreign))
. summarize foreign p
```

| Variable | Obs | Mean     | Std. Dev. | Min      | Max     |
|----------|-----|----------|-----------|----------|---------|
| foreign  | 74  | .2972973 | .4601885  | 0        | 1       |
| p        | 74  | .2974049 | .182352   | .0714664 | .871624 |

◀

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [scobit](#) — Skewed logistic regression

[U] [20 Estimation and postestimation commands](#)

## Syntax

*One-sample variance-comparison test*

```
sdtest varname == # [if] [in] [, level(#)]
```

*Two-sample variance-comparison test*

```
sdtest varname1 == varname2 [if] [in] [, level(#)]
```

*Two-group variance-comparison test*

```
sdtest varname [if] [in] , by(groupvar) [level(#)]
```

*Immediate form of one-sample variance-comparison test*

```
sdtesti #obs { #mean | . } #sd #val [, level(#)]
```

*Immediate form of two-sample variance-comparison test*

```
sdtesti #obs,1 { #mean,1 | . } #sd,1 #obs,2 { #mean,2 | . } #sd,2 [, level(#)]
```

*Robust tests for equality of variance*

```
robvar varname [if] [in] , by(groupvar)
```

*by* is allowed with `sdtest` and `robvar`; see [\[D\]](#) `by`.

## Menu

### one-sample

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > One-sample variance-comparison test

### two-sample

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > Two-sample variance-comparison test

### two-group

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > Two-group variance-comparison test

### immediate command: one-sample

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > One-sample variance-comparison calculator

## immediate command: two-sample

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > Two-sample variance-comparison calculator

## robvar

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > Robust equal variance test

# Description

`sdtest` performs tests on the equality of standard deviations (variances). In the first form, `sdtest` tests that the standard deviation of *varname* is *#*. In the second form, `sdtest` tests that *varname*<sub>1</sub> and *varname*<sub>2</sub> have the same standard deviation. In the third form, `sdtest` performs the same test, using the standard deviations of the two groups defined by *groupvar*.

`sdtesti` is the immediate form of `sdtest`; see [U] 19 Immediate commands.

Both the traditional *F* test for the homogeneity of variances and Bartlett's generalization of this test to *K* samples are sensitive to the assumption that the data are drawn from an underlying Gaussian distribution. See, for example, the cautionary results discussed by Markowski and Markowski (1990). Levene (1960) proposed a test statistic for equality of variance that was found to be robust under nonnormality. Then Brown and Forsythe (1974) proposed alternative formulations of Levene's test statistic that use more robust estimators of central tendency in place of the mean. These reformulations were demonstrated to be more robust than Levene's test when dealing with skewed populations.

`robvar` reports Levene's robust test statistic ( $W_0$ ) for the equality of variances between the groups defined by *groupvar* and the two statistics proposed by Brown and Forsythe that replace the mean in Levene's formula with alternative location estimators. The first alternative ( $W_{50}$ ) replaces the mean with the median. The second alternative replaces the mean with the 10% trimmed mean ( $W_{10}$ ).

# Options

`level(#)` specifies the confidence level, as a percentage, for confidence intervals of the means. The default is `level(95)` or as set by `set level`; see [U] 20.7 Specifying the width of confidence intervals.

`by(groupvar)` specifies the *groupvar* that defines the groups to be compared. For `sdtest`, there should be two groups, but for `robvar` there may be more than two groups. Do not confuse the `by()` option with the `by` prefix; both may be specified.

# Remarks

Remarks are presented under the following headings:

*Basic form*  
*Immediate form*  
*Robust test*

## Basic form

`sdtest` performs two different statistical tests: one testing equality of variances and the other testing that the standard deviation is equal to a known constant. Which test it performs is determined by whether you type a variable name or a number to the right of the equal sign.

### ► Example 1: One-sample test of variance

We have a sample of 74 automobiles. For each automobile, we know the mileage rating. We wish to test whether the overall standard deviation is 5 mpg:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
```

```
. sdtest mpg == 5
```

One-sample test of variance

| Variable | Obs | Mean    | Std. Err. | Std. Dev. | [95% Conf. Interval] |          |
|----------|-----|---------|-----------|-----------|----------------------|----------|
| mpg      | 74  | 21.2973 | .6725511  | 5.785503  | 19.9569              | 22.63769 |

```
sd = sd(mpg) c = chi2 = 97.7384
Ho: sd = 5 degrees of freedom = 73
Ha: sd < 5 Ha: sd != 5 Ha: sd > 5
Pr(C < c) = 0.9717 2*Pr(C > c) = 0.0565 Pr(C > c) = 0.0283
```

◀

### ► Example 2: Variance ratio test

We are testing the effectiveness of a new fuel additive. We run an experiment on 12 cars, running each without and with the additive. The data can be found in [R] `ttest`. The results for each car are stored in the variables `mpg1` and `mpg2`:

```
. use http://www.stata-press.com/data/r11/fuel
```

```
. sdtest mpg1==mpg2
```

Variance ratio test

| Variable | Obs | Mean   | Std. Err. | Std. Dev. | [95% Conf. Interval] |          |
|----------|-----|--------|-----------|-----------|----------------------|----------|
| mpg1     | 12  | 21     | .7881701  | 2.730301  | 19.26525             | 22.73475 |
| mpg2     | 12  | 22.75  | .9384465  | 3.250874  | 20.68449             | 24.81551 |
| combined | 24  | 21.875 | .6264476  | 3.068954  | 20.57909             | 23.17091 |

```
ratio = sd(mpg1) / sd(mpg2) f = 0.7054
Ho: ratio = 1 degrees of freedom = 11, 11
Ha: ratio < 1 Ha: ratio != 1 Ha: ratio > 1
Pr(F < f) = 0.2862 2*Pr(F < f) = 0.5725 Pr(F > f) = 0.7138
```

We cannot reject the hypothesis that the standard deviations are the same.

In [R] `ttest`, we draw an important distinction between paired and unpaired data, which, in this example, means whether there are 12 cars in a before-and-after experiment or 24 different cars. For `sdtest`, on the other hand, there is no distinction. If the data had been unpaired and stored as described in [R] `ttest`, we could have typed `sdtest mpg, by(treated)`, and the results would have been the same.

◀

## Immediate form

### ➤ Example 3: sdtesti

Immediate commands are used not with data, but with reported summary statistics. For instance, to test whether a variable on which we have 75 observations and a reported standard deviation of 6.5 comes from a population with underlying standard deviation 6, we would type

```
. sdtesti 75 . 6.5 6
One-sample test of variance
```

|   | Obs | Mean | Std. Err. | Std. Dev. | [95% Conf. Interval] |   |
|---|-----|------|-----------|-----------|----------------------|---|
| x | 75  | .    | .7505553  | 6.5       | .                    | . |

```
 sd = sd(x) c = chi2 = 86.8472
Ho: sd = 6 degrees of freedom = 74
 Ha: sd < 6 Ha: sd != 6 Ha: sd > 6
Pr(C < c) = 0.8542 2*Pr(C > c) = 0.2916 Pr(C > c) = 0.1458
```

The mean plays no role in the calculation, so it may be omitted.

To test whether the variable comes from a population with the same standard deviation as another for which we have a calculated standard deviation of 7.5 over 65 observations, we would type

```
. sdtesti 75 . 6.5 65 . 7.5
Variance ratio test
```

|          | Obs | Mean | Std. Err. | Std. Dev. | [95% Conf. Interval] |   |
|----------|-----|------|-----------|-----------|----------------------|---|
| x        | 75  | .    | .7505553  | 6.5       | .                    | . |
| y        | 65  | .    | .9302605  | 7.5       | .                    | . |
| combined | 140 | .    | .         | .         | .                    | . |

```
 ratio = sd(x) / sd(y) f = 0.7511
Ho: ratio = 1 degrees of freedom = 74, 64
 Ha: ratio < 1 Ha: ratio != 1 Ha: ratio > 1
Pr(F < f) = 0.1172 2*Pr(F < f) = 0.2344 Pr(F > f) = 0.8828
```



## Robust test

### ➤ Example 4: robvar

We wish to test whether the standard deviation of the length of stay for patients hospitalized for a given medical procedure differs by gender. Our data consist of observations on the length of hospital stay for 1778 patients: 884 males and 894 females. Length of stay, `lengthstay`, is highly skewed (skewness coefficient = 4.912591) and thus violates Bartlett's normality assumption. Therefore, we use `robvar` to compare the variances.



```
. use http://www.stata-press.com/data/r11/stay
```

```
. robvar lengthstay, by(sex)
```

| sex             | Summary of Length of stay in days |                    |       |
|-----------------|-----------------------------------|--------------------|-------|
|                 | Mean                              | Std. Dev.          | Freq. |
| male            | 9.0874434                         | 9.7884747          | 884   |
| female          | 8.800671                          | 9.1081478          | 894   |
| Total           | 8.9432508                         | 9.4509466          | 1778  |
| W0 = .55505315  | df(1, 1776)                       | Pr > F = .45635888 |       |
| W50 = .42714734 | df(1, 1776)                       | Pr > F = .51347664 |       |
| W10 = .44577674 | df(1, 1776)                       | Pr > F = .50443411 |       |

For these data, we cannot reject the null hypothesis that the variances are equal. However, Bartlett's test yields a significance probability of 0.0319 because of the pronounced skewness of the data.



## □ Technical note

`robvar` implements both the conventional Levene's test centered at the mean and a median-centered test. In a simulation study, [Conover, Johnson, and Johnson \(1981\)](#) compare the properties of the two tests and recommend using the median test for asymmetric data, although for small sample sizes the test is somewhat conservative. See [Carroll and Schneider \(1985\)](#) for an explanation of why both mean- and median-centered tests have approximately the same level for symmetric distributions, but for asymmetric distributions the median test is closer to the correct level.



*(Continued on next page)*

## Saved results

`sdtest` and `sdtesti` save the following in `r()`:

|                     |                            |                      |                                        |
|---------------------|----------------------------|----------------------|----------------------------------------|
| Scalars             |                            |                      |                                        |
| <code>r(N)</code>   | number of observations     | <code>r(sd_1)</code> | standard deviation for first variable  |
| <code>r(p_l)</code> | lower one-sided $p$ -value | <code>r(sd_2)</code> | standard deviation for second variable |
| <code>r(p_u)</code> | upper one-sided $p$ -value | <code>r(df)</code>   | degrees of freedom                     |
| <code>r(p)</code>   | two-sided $p$ -value       | <code>r(df_1)</code> | numerator degrees of freedom           |
| <code>r(F)</code>   | $F$ statistic              | <code>r(df_2)</code> | denominator degrees of freedom         |
| <code>r(sd)</code>  | standard deviation         | <code>r(chi2)</code> | $\chi^2$                               |

`robvar` saves the following in `r()`:

|                       |                                             |                       |                                                   |
|-----------------------|---------------------------------------------|-----------------------|---------------------------------------------------|
| Scalars               |                                             |                       |                                                   |
| <code>r(N)</code>     | number of observations                      | <code>r(w10)</code>   | Brown and Forsythe's $F$ statistic (trimmed mean) |
| <code>r(w50)</code>   | Brown and Forsythe's $F$ statistic (median) | <code>r(p_w10)</code> | Brown and Forsythe's $p$ -value (trimmed mean)    |
| <code>r(p_w50)</code> | Brown and Forsythe's $p$ -value             | <code>r(df_1)</code>  | numerator degrees of freedom                      |
| <code>r(w0)</code>    | Levene's $F$ statistic                      | <code>r(df_2)</code>  | denominator degrees of freedom                    |
| <code>r(p_w0)</code>  | Levene's $p$ -value                         |                       |                                                   |

## Methods and formulas

`sdtest`, `sdtesti`, and `robvar` are implemented as ado-files.

See Armitage et al. (2002, 149–153) or Bland (2000, 171–172) for an introduction and explanation of the calculation of these tests.

The test for  $\sigma = \sigma_0$  is given by

$$\chi^2 = \frac{(n-1)s^2}{\sigma_0^2}$$

which is distributed as  $\chi^2$  with  $n - 1$  degrees of freedom.

The test for  $\sigma_x^2 = \sigma_y^2$  is given by

$$F = \frac{s_x^2}{s_y^2}$$

which is distributed as  $F$  with  $n_x - 1$  and  $n_y - 1$  degrees of freedom.

`robvar` is also implemented as an ado-file.

Let  $X_{ij}$  be the  $j$ th observation of  $X$  for the  $i$ th group. Let  $Z_{ij} = |X_{ij} - \overline{X}_i|$ , where  $\overline{X}_i$  is the mean of  $X$  in the  $i$ th group. Levene's test statistic is

$$W_0 = \frac{\sum_i n_i (\overline{Z}_i - \overline{Z})^2 / (g - 1)}{\sum_i \sum_j (Z_{ij} - \overline{Z}_i)^2 / \sum_i (n_i - 1)}$$

where  $n_i$  is the number of observations in group  $i$  and  $g$  is the number of groups.  $W_{50}$  is obtained by replacing  $\overline{X}_i$  with the  $i$ th group median of  $X_{ij}$ , whereas  $W_{10}$  is obtained by replacing  $\overline{X}_i$  with the 10% trimmed mean for group  $i$ .

## References

- Armitage, P., G. Berry, and J. N. S. Matthews. 2002. *Statistical Methods in Medical Research*. 4th ed. Oxford: Blackwell.
- Bland, M. 2000. *An Introduction to Medical Statistics*. 3rd ed. Oxford: Oxford University Press.
- Brown, M. B., and A. B. Forsythe. 1974. Robust tests for the equality of variances. *Journal of the American Statistical Association* 69: 364–367.
- Carroll, R. J., and H. Schneider. 1985. A note on Levene’s tests for equality of variances. *Statistics and Probability Letters* 3: 191–194.
- Cleves, M. A. 1995. [sg35: Robust tests for the equality of variances](#). *Stata Technical Bulletin* 25: 13–15. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 91–93. College Station, TX: Stata Press.
- . 2000. [sg35.2: Robust tests for the equality of variances update to Stata 6](#). *Stata Technical Bulletin* 53: 17–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 158–159. College Station, TX: Stata Press.
- Conover, W. J., M. E. Johnson, and M. M. Johnson. 1981. A comparative study of tests for homogeneity of variances, with applications to the outer continental shelf bidding data. *Technometrics* 23: 351–361.
- Levene, H. 1960. Robust tests for equality of variances. In *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*, ed. I. Olkin, S. G. Ghurye, W. Hoeffding, W. G. Madow, and H. B. Mann, 278–292. Menlo Park, CA: Stanford University Press.
- Markowski, C. A., and E. P. Markowski. 1990. Conditions for the effectiveness of a preliminary test of variance. *American Statistician* 44: 322–326.
- Seed, P. T. 2000. [sbe33: Comparing several methods of measuring the same quantity](#). *Stata Technical Bulletin* 55: 2–9. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 73–82. College Station, TX: Stata Press.
- Tobías, A. 1998. [gr28: A graphical procedure to test equality of variances](#). *Stata Technical Bulletin* 42: 4–6. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 68–70. College Station, TX: Stata Press.

## Also see

[\[R\] ttest](#) — Mean-comparison tests

Syntax

```
search word [word ...] [, search_options]

set searchdefault { local | net | all } [, permanently]

findit word [word ...]
```

| search_options    | description                                                                                                    |
|-------------------|----------------------------------------------------------------------------------------------------------------|
| local             | search using Stata's keyword database; the default                                                             |
| net               | search across materials available via Stata's <b>net</b> command                                               |
| all               | search across both the local keyword database and the <b>net</b> material                                      |
| <u>author</u>     | search by author's name                                                                                        |
| <u>entry</u>      | search by entry ID                                                                                             |
| <u>exact</u>      | search across both the local keyword database and the <b>net</b> materials; prevents matching on abbreviations |
| faq               | search the FAQs posted to the Stata web site                                                                   |
| <u>historical</u> | search entries that are of historical interest only                                                            |
| or                | list an entry if <i>any</i> of the words typed after <b>search</b> are associated with the entry               |
| <u>manual</u>     | search the entries in the <i>Stata Documentation</i>                                                           |
| sj                | search the entries in the <i>Stata Journal</i> and the STB                                                     |

Menu

Help > Search...

Description

**search** searches a keyword database and the Internet.

Capitalization of the words following **search** is irrelevant, as is the inclusion or exclusion of special characters such as commas, hyphens, and the like.

**set searchdefault** affects the default behavior of the **search** command. **local** is the default.

**findit** is equivalent to **search word** [*word ...*], **all**. **findit** results are displayed in the Viewer. **findit** is the best way to search for information on a topic across all sources, including the online help, the FAQs at the Stata web site, the *Stata Journal*, and all Stata-related Internet sources including user-written additions. From **findit**, you can click to go to a source or to install additions.

See [\[R\] hsearch](#) for a command that searches help files.

## Options for search

**local**, the default (unless changed by `set searchdefault`), specifies that the search be performed using only Stata's keyword database.

**net** specifies that the search be performed across the materials available via Stata's `net` command. Using `search word [word ...]`, **net** is equivalent to typing `net search word [word ...]` (without options); see [R] [net search](#).

**all** specifies that the search be performed across both the local keyword database and the **net** materials.

**author** specifies that the search be performed on the basis of author's name rather than keywords. A search with the **author** option is performed on the local keyword database only.

**entry** specifies that the search be performed on the basis of entry IDs rather than keywords. A search with the **entry** option is performed on the local keyword database only.

**exact** prevents matching on abbreviations. A search with the **exact** option is performed across both the local keyword database and the **net** materials.

**faq** limits the search to the FAQs posted on the Stata web site: <http://www.stata.com>. A search with the **faq** option is performed on the local keyword database only.

**historical** adds to the search entries that are of historical interest only. By default, such entries are not listed. Past entries are classified as historical if they discuss a feature that later became an official part of Stata. Updates to historical entries will always be found, even if **historical** is not specified. A search with the **historical** option is performed on the local keyword database only.

**or** specifies that an entry be listed if any of the words typed after **search** are associated with the entry. The default is to list the entry only if all the words specified are associated with the entry. A search with the **or** option is performed on the local keyword database only.

**manual** limits the search to entries in the *Stata Documentation*; that is, the search is limited to the *User's Guide* and all the reference manuals. A search with the **manual** option is performed on the local keyword database only.

**sj** limits the search to entries in the *Stata Journal* and its predecessor, the *Stata Technical Bulletin*. A search with the **sj** option is performed on the local keyword database only.

## Option for set searchdefault

**permanently** specifies that, in addition to making the change right now, the `searchdefault` setting be remembered and become the default setting when you invoke Stata.

## Remarks

Remarks are presented under the following headings:

- [Introduction](#)
- [Internet searches](#)
- [Author searches](#)
- [Entry ID searches](#)
- [Return codes](#)

## Introduction

See [U] [4 Stata's help and search facilities](#) for a tutorial introduction to `search`. `search` is one of Stata's most useful commands. To understand the advanced features of `search`, you need to know how it works.

`search` has a database—files—containing the titles, etc., of every entry in the *User's Guide*, the *Base Reference Manual*, the *Data-Management Reference Manual*, the *Graphics Reference Manual*, the *Longitudinal-Data/Panel-Data Reference Manual*, the *Multiple-Imputation Reference Manual*, the *Multivariate Statistics Reference Manual*, the *Programming Reference Manual*, the *Survey Data Reference Manual*, the *Survival Analysis and Epidemiological Tables Reference Manual*, the *Time-Series Reference Manual*, the *Mata Reference Manual*, the articles in the *Stata Journal* and the *Stata Technical Bulletin*, and the FAQs posted on the Stata web site. In these files is a list of words, called keywords, associated with each entry.

When you type `search xyz`, `search` reads the database and compares the list of keywords with `xyz`. If it finds `xyz` in the list or a keyword that allows an abbreviation of `xyz`, it displays the entry.

When you type `search xyz abc`, `search` does the same thing but displays an entry only if it contains both keywords. The order does not matter, so you can `search linear regression` or `search regression linear`.

Obviously, how many entries `search` finds depends on how the search database was constructed. We have included a plethora of keywords under the theory that, for a given request, it is better to list too much rather than risk listing nothing at all. Still, you are in the position of guessing the keywords. Do you look up normality test, normality tests, or tests of normality? Well, normality test would be best, but all would work. In general, use the singular, and strike the unnecessary words. For guidelines for specifying keywords, see [U] [4.6 More on search](#).

`set searchdefault` allows you to specify where `search` searches. `set searchdefault local`, the default, restricts `search` to using only Stata's keyword database. `set searchdefault net` restricts `search` to searching only the Internet. `set searchdefault all` indicates that both the keyword database and the Internet are to be searched.

## Internet searches

`search` with the `net` option searches the Internet for user-written additions to Stata, including, but not limited to, user-written additions published in the *Stata Journal* (SJ) and the *Stata Technical Bulletin* (STB). `search keywords`, `net` performs the same search as the command `net search` (with no options); see [R] [net search](#).

```
. search random effect, net
```

### **Keyword search**

```
Keywords: random effect
```

```
Search: (1) Web resources from Stata and from other users
```

### **Web resources from Stata and other users**

```
(contacting http://www.stata.com)
```

```
43 packages found (Stata Journal and STB listed first)
```

```

st0156 from http://www.stata-journal.com/software/sj9-1
```

```
SJ9-1 st0156. Multivariate random-effects meta-analysis / Multivariate
random-effects meta-analysis / by Ian R. White, MRC Biostatistics Unit, /
Institute of Public Health, UK / Support: ian.white@mrc-bsu.cam.ac.uk /
After installation, type help mvmeta and mvmeta_make
```

```

st0158 from http://www.stata-journal.com/software/sj9-1
 SJ9-1 st0158. A general-purpose method for two-group... / A
 general-purpose method for two-group randomization tests / by Johannes
 Kaiser, Lab. for Experimental Economics, / Univ. of Bonn, Germany /
 Michael G. Lacy, Sociology Dept., Colo. State. Univ., / Fort Collins, CO,
 (output omitted)

paran from http://www.doyenne.com/stata
 paran. Horn's Test of Principal Components/Factors (Parallel Analysis) /
 Program by Alexis Dinno. / Support: adinno at post dot harvard dot edu /
 Version: 1.4.0 (updated October 14, 2008) / Distribution-Date: 14oct2008 /
 / This package contains the program {cmd:paran} which performs Horn's
 permutation from http://www-gene.cimr.cam.ac.uk/clayton/software/stata
 Routines for generalized permutation testing / David Clayton / / Diabetes
 and Inflammation Laboratory Tel: 44 (0)1223 762669 / Cambridge Institute
 for Medical Research Fax: 44 (0)1223 762102 / Wellcome Trust/MRC Building
 david.claytoncimr.cam.ac.uk / Addenbrooke's Hospital, Cambridge, CB2 2XY
 (output omitted)

28 references found in tables of contents

http://www.stata-journal.com/software/sj9-1/
 Row rank and sort a set of variables / Update: Cumulative meta-analysis,
 with graphics / Update: GLS for trend estimation of summarized /
 dose-response data / Update: Fit a linear model with two high-dimensional
 / fixed effects / Estimation and comparison of receiver operating /

http://www.stata-journal.com/software/sj8-4/
 Map chains of events / Report number(s) of distinct observations or values
 / Update: Drop variables (observations) that are all / missing / Update:
 Graphical representation of interactions / Update: Meta-regression in
 Stata (revised) / Update: Concordance correlation coefficient and /
 (output omitted)

http://www.stata.com/users/jhilbe/
 Materials by Joseph Hilbe, Arizona State University / The following
 programs have been supplied by Joseph Hilbe, who retired from / the
 University of Hawaii in 1990 and since 1992 has served as an adjunct /
 professor at Arizona State University. Dr. Hilbe was the founding editor
 (end of search)

```

## Author searches

search ordinarily compares the words following search with the keywords for the entry. If you specify the author option, however, it compares the words with the author's name. In the search database, we have filled in author names for all SJ and STB inserts.

For instance, in [R] **kdensity** in this manual you will discover that Isaías H. Salgado-Ugarte wrote the first version of Stata's **kdensity** command and published it in the STB. Assume that you read his original insert and found the discussion useful. You might now wonder what else he has written in the SJ or STB. To find out, you type

```

. search Salgado-Ugarte, author
(output omitted)

```

Names like Salgado-Ugarte are confusing to many people. search does not require you to specify the entire name; what you type is compared with each "word" of the name and, if any part matches, the entry is listed. The hyphen is a special character, and you can omit it. Thus you can obtain the same list by looking up Salgado, Ugarte, or Salgado Ugarte without the hyphen.

Actually, to find all entries written by Salgado-Ugarte, you need to type

```
. search Salgado-Ugarte, author historical
(output omitted)
```

Prior inserts in the SJ or STB that provide a feature that later was superseded by a built-in feature of Stata are marked as historical in the search database and, by default, are not listed. The `historical` option ensures that all entries are listed.

## Entry ID searches

If you specify the `entry` option, `search` compares what you have typed with the entry ID. The entry ID is not the title—it is the reference listed to the left of the title that tells you where to look. For instance, in

```
[R] regress Linear regression
(help regress)
```

[R] `regress` is the entry ID. This is a reference, of course, to this manual. In

```
FAQ Analysis of multiple failure-time survival data
. M. Cleves
11/99 How do I analyze multiple failure-time data using Stata?
http://www.stata.com/support/faqs/stat/stmfail.html
```

“FAQ” is the entry ID. In

```
SJ-7-1 st0118 . . A survey on survey stat.: What is and can be done in Stata
. F. Kreuter and R. Valliant
Q1/07 SJ7(1):1--21 (no commands)
discusses survey issues in analyzing complex survey
data and describes some of Stata's capabilities for
such analyses
```

“SJ-7-1” is the entry ID.

`search` with the `entry` option searches these entry IDs.

Thus you could generate a table of contents for the *User's Guide* by typing

```
. search [U], entry
(output omitted)
```

You could generate a table of contents for *Stata Journal*, Volume 1, Issue 1, by typing

```
. search sj-1-1, entry
(output omitted)
```

To generate a table of contents for the 26th issue of the STB, you would type

```
. search STB-26, entry historical
(output omitted)
```

The `historical` option here is possibly important. STB-26 was published in July 1995, and perhaps some of its inserts have already been marked historical.

You could obtain a list of all inserts associated with sg53 by typing

```
. search sg53, entry historical
(output omitted)
```

Again we include the `historical` option in case any of the relevant inserts have been marked historical.



## Return codes

In addition to indexing the entries in the *User's Guide* and all the *Reference* manuals, `search` also can be used to search return codes.

To see information on return code 131, type

```
. search rc 131
[R] Return code 131
 not possible with test;
 You requested a test of a hypothesis that is nonlinear in the
 variables. test tests only linear hypotheses. Use testnl.
```

If you want a list of all Stata return codes, type

```
. search error, entry
(output omitted)
```

## Methods and formulas

`findit` is implemented as an ado-file.

## Acknowledgment

`findit` grew from a suggestion by Nicholas J. Cox, Durham University.

## Also see

[R] [hsearch](#) — Search help files

[R] [help](#) — Display online help

[R] [net search](#) — Search the Internet for installable packages

[U] [4 Stata's help and search facilities](#)

# Title

serrbar — Graph standard error bar chart

## Syntax

```
serrbar mvar svar xvar [if] [in] [, options]
```

| options                                 | description                                                                               |
|-----------------------------------------|-------------------------------------------------------------------------------------------|
| Main                                    |                                                                                           |
| <code>scale(#)</code>                   | scale length of graph bars; default is <code>scale(1)</code>                              |
| Error bars                              |                                                                                           |
| <code>rcap_options</code>               | affect rendition of capped spikes                                                         |
| Plotted points                          |                                                                                           |
| <code>mvopts(scatter_options)</code>    | affect rendition of plotted points                                                        |
| Add plots                               |                                                                                           |
| <code>addplot(plot)</code>              | add other plots to generated graph                                                        |
| Y axis, X axis, Titles, Legend, Overall |                                                                                           |
| <code>twoway_options</code>             | any options other than <code>by()</code> documented in <a href="#">[G] twoway_options</a> |

## Menu

Statistics > Other > Quality control > Standard error bar chart

## Description

`serrbar` graphs  $mvar \pm scale() \times svar$  against `xvar`. Usually, but not necessarily, `mvar` and `svar` will contain means and standard errors or standard deviations of some variable so that a standard error bar chart is produced.

## Options

|                                                                                                                                                                                                              |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Main                                                                                                                                                                                                         |
| <code>scale(#)</code> controls the length of the bars. The upper and lower limits of the bars will be $mvar + scale() \times svar$ and $mvar - scale() \times svar$ . The default is <code>scale(1)</code> . |
| Error bars                                                                                                                                                                                                   |
| <code>rcap_options</code> affect the rendition of the plotted error bars (the capped spikes). See <a href="#">[G] graph twoway rcap</a> .                                                                    |

## Plotted points

`mvopts(scatter_options)` affects the rendition of the plotted points (*mvar* versus *xvar*). See [G] [graph twoway scatter](#).

## Add plots

`addplot(plot)` provides a way to add other plots to the generated graph; see [G] [addplot\\_option](#).

## Y axis, X axis, Titles, Legend, Overall

*twoway\_options* are any of the options documented in [G] [twoway\\_options](#), excluding `by()`. These include options for titling the graph (see [G] [title\\_options](#)) and for saving the graph to disk (see [G] [saving\\_option](#)).

## Remarks

### ► Example 1

In quality-control applications, the three most commonly used variables with this command are the process mean, process standard deviation, and time. For instance, we have data on the average weights and standard deviations from an assembly line in San Francisco for the period January 8 to January 16. Our data are

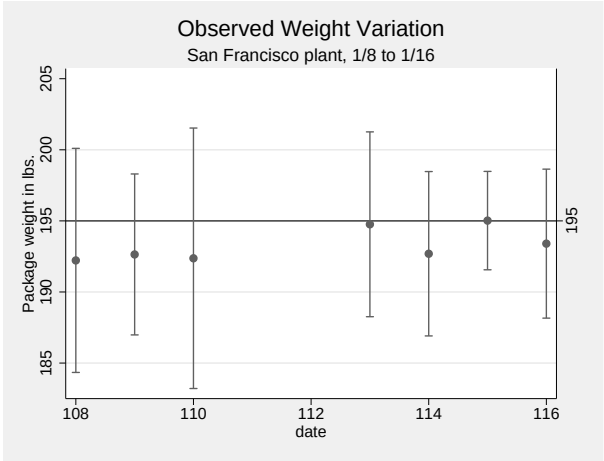
```
. use http://www.stata-press.com/data/r11/assembly
. list, sep(0) divider
```

|    | date | mean   | std  |
|----|------|--------|------|
| 1. | 108  | 192.22 | 3.94 |
| 2. | 109  | 192.64 | 2.83 |
| 3. | 110  | 192.37 | 4.58 |
| 4. | 113  | 194.76 | 3.25 |
| 5. | 114  | 192.69 | 2.89 |
| 6. | 115  | 195.02 | 1.73 |
| 7. | 116  | 193.40 | 2.62 |

We type `serrbar mean std date, scale(2)` but, after seeing the result, decide to make it fancier:

(Continued on next page)

```
. serrbar mean std date, scale(2) title("Observed Weight Variation")
> sub("San Francisco plant, 1/8 to 1/16") ylabel(195) yaxis(1 2)
> ylab(195, axis(2)) ytitle("", axis(2))
```



Methods and formulas

serrbar is implemented as an ado-file.

Acknowledgment

serrbar was written by Nicholas J. Cox of Durham University.

Also see

[R] qc — Quality control charts

# Title

set — Overview of system parameters

## Syntax

```
set [setcommand ...]
```

`set` typed without arguments is equivalent to `query` typed without arguments.

## Description

This entry provides a reference to Stata's `set` commands. For many entries, more thorough information is provided elsewhere; see the Reference field in each entry below for the location of this information.

To reset system parameters to factory defaults, see [\[R\] `set\_defaults`](#).

## Remarks

`set adosize`

Syntax: `set adosize # [ , permanently ]`

Default: 1,000

Description: sets the maximum amount of memory that automatically loaded do-files may consume.  $10 \leq \# \leq 10000$ .

Reference: [\[P\] `sysdir`](#)

`set autotabgraphs` (Windows only)

Syntax: `set autotabgraphs {on|off} [ , permanently ]`

Default: off

Description: determines whether graphs are created as tabs within one window or as separate windows.

`set checksum`

Syntax: `set checksum {on|off} [ , permanently ]`

Default: off

Description: determines whether files should be prevented from being downloaded from the Internet if checksums do not match.

Reference: [\[D\] `checksum`](#)

**set conren** (Unix console only)Syntax 1:    **set conren**Syntax 2:    **set conren clear**Syntax 3:    **set conren** [**sf** | **bf** | **it**]  
                  {**result** | [**txt** | **text**] | **input** | **error** | **link** | **hilite**}  
                  [**char**[**char**...]]Syntax 4:    **set conren** {**ulon** | **uloff**} [**char** [**char**...]]Syntax 5:    **set conren reset** [**char** [**char**...]]

Description: can possibly make the output on your screen appear prettier.

**set conren** displays a list of the currently defined display codes.**set conren clear** clears all codes.**set conren** followed by a font type (**bf**, **sf**, or **it**) and display context (**result**, **error**, **link**, or **hilite**) and then followed by a series of space-separated characters sets the code for the specified font type and display context. If the font type is omitted, the code is set to the same specified code for all three font types.**set conren ulon** and **set conren uloff** set the codes for turning on and off underlining.**set conren reset** sets the code that will turn off all display and underlining codes.Reference:    [\[GSU\] conren](#)**set copycolor** (Mac and Windows only)Syntax:      **set copycolor** {**automatic** | **asis** | **gs1** | **gs2** | **gs3**} [, **permanently**]Default:     **automatic**

Description: determines how colors are handled when graphs are copied to the Clipboard.

Reference:    [\[G\] set printcolor](#)**set dockable** (Windows only)Syntax:      **set dockable** {**on** | **off**} [, **permanently**]Default:     **off**

Description: determines whether to enable the use of dockable window characteristics, including the ability to dock or tab a window into another window.

**set dockingguides** (Windows only)Syntax:      **set dockingguides** {**on** | **off**} [, **permanently**]Default:     **on**

Description: determines whether to enable the use of dockable guides when repositioning a dockable window.

**set doublebuffer** (Windows only)Syntax:      **set doublebuffer** {**on** | **off**} [, **permanently**]Default:     **on**

Description: enables or disables double buffering of the Results, Viewer, and Data Editor windows. Double buffering prevents the windows from flickering when redrawn or resized. Users who encounter performance problems such as the Results window outputting very slowly should disable double buffering.

**set dp**

Syntax:     **set dp** {comma|period} [ , permanently ]  
Default:    period  
Description: determines whether a period or a comma is to be used as the decimal point.  
Reference:   [D] [format](#)

**set emptycells**

Syntax:     **set emptycells** {keep|drop} [ , permanently ]  
Default:    keep  
Description: sets what to do with empty cells in interactions.  
Reference:   [R] [set emptycells](#)

**set eolchar (Mac only)**

Syntax:     **set eolchar** {mac|unix} [ , permanently ]  
Default:    unix  
Description: sets the default end-of-line delimiter for text files created in Stata.

**set fastscroll (Unix and Windows only)**

Syntax:     **set fastscroll** {on|off} [ , permanently ]  
Default:    on  
Description: sets the scrolling method for new output in the Results window. Setting **fastscroll** to on is faster but can be jumpy. Setting **fastscroll** to off is slower but smoother.

**set floatresults (Windows only)**

Syntax:     **set floatresults** {on|off}  
Default:    off  
Description: determines whether to enable floating window behavior for the Results window. The term “float” in this context means the Results window will always float over the main Stata window; the Results window can never be placed behind the main Stata window. There is no **permanently** option because **permanently** is implied.

**set floatwindows (Windows only)**

Syntax:     **set floatwindows** {on|off}  
Default:    off  
Description: determines whether to enable floating window behavior for dialog boxes and dockable window. The term “float” in this context means that a window will always float over the main Stata window; these windows cannot be placed behind the main Stata window. There is no **permanently** option because **permanently** is implied.

**set graphics**

Syntax:     **set graphics** {on|off}  
Default:    on; default is off for console Stata  
Description: determines whether graphs are displayed on your monitor.  
Reference:   [G] [set graphics](#)

**set httpproxy**

Syntax:      `set httpproxy {on|off} [ , init ]`

Default:      `off`

Description: turns on/off the use of a proxy server. There is no `permanently` option because `permanently` is implied.

Reference:    [\[R\] netio](#)

**set httpproxyauth**

Syntax:      `set httpproxyauth {on|off}`

Default:      `off`

Description: determines whether authorization is required for the proxy server. There is no `permanently` option because `permanently` is implied.

Reference:    [\[R\] netio](#)

**set httpproxyhost**

Syntax:      `set httpproxyhost ["name"]`

Description: sets the name of a host to be used as a proxy server. There is no `permanently` option because `permanently` is implied.

Reference:    [\[R\] netio](#)

**set httpproxyport**

Syntax:      `set httpproxyport #`

Default:      8080 if Stata cannot autodetect the proper setting for your computer.

Description: sets the port number for a proxy server. There is no `permanently` option because `permanently` is implied.

Reference:    [\[R\] netio](#)

**set httpproxypw**

Syntax:      `set httpproxypw ["password"]`

Description: sets the appropriate password. There is no `permanently` option because `permanently` is implied.

Reference:    [\[R\] netio](#)

**set httpproxyuser**

Syntax:      `set httpproxyuser ["name"]`

Description: sets the appropriate user ID. There is no `permanently` option because `permanently` is implied.

Reference:    [\[R\] netio](#)

**set level**

Syntax:      `set level # [ , permanently ]`

Default:      95

Description: sets the default significance level for confidence intervals for all commands that report confidence intervals.  $10.00 \leq \# \leq 99.99$ , and  $\#$  can have at most two digits after the decimal point.

Reference:    [\[R\] level](#)



**set linegap**Syntax: set linegap #

Default: 1

Description: sets the space between lines, in pixels, in the Results window. There is no permanently option because permanently is implied.

**set linesize**Syntax: set linesize #

Default: 1 less than the full width of the screen

Description: sets the line width, in characters, for both the screen and the log file.

Reference: [\[R\]](#) [log](#)**set locksplitters (Windows only)**Syntax: set locksplitters {on|off} [, permanently]

Default: off

Description: determines whether splitters should be locked so that docked windows cannot be resized.

**set logtype**Syntax: set logtype {text|smcl} [, permanently]

Default: smcl

Description: sets the default log filetype.

Reference: [\[R\]](#) [log](#)

set matacache, set matafavor, set matalibs, set matalnum, set matamofirst, set mataoptimize, and set matastrict; see [\[M-3\]](#) [mata set](#).

**set matsize**Syntax: set matsize # [, permanently]

Default: 400 for Stata/MP, Stata/SE, and Stata/IC; 40 for Small Stata

Description: sets the maximum number of variables that can be included in any estimation command. This setting cannot be changed in Small Stata.

 $10 \leq \# \leq 11000$  for Stata/MP and Stata/SE;  $10 \leq \# \leq 800$  for Stata/IC.
Reference: [\[R\]](#) [matsize](#)**set maxdb**Syntax: set maxdb # [, permanently]

Default: 50

Description: sets the maximum number of dialog boxes whose contents are remembered from one invocation to the next during a session.  $5 \leq \# \leq 1000$ .Reference: [\[R\]](#) [db](#)**set maxiter**Syntax: set maxiter # [, permanently]

Default: 16000

Description: sets the default maximum number of iterations for estimation commands.  $0 \leq \# \leq 16000$ .Reference: [\[R\]](#) [maximize](#)

**set maxvar**

Syntax:      `set maxvar # [ , permanently ]`  
Default:      5000 for Stata/MP and Stata/SE; 2048 for Stata/IC; 99 for Small Stata  
Description: sets the maximum number of variables. This can be changed only in Stata/MP and Stata/SE.  $2048 \leq \# \leq 32767$ .  
Reference:    [\[D\] memory](#)

**set memory**

Syntax:      `set memory #[b|k|m|g] [ , permanently ]`  
Default:      50m for Stata/MP and Stata/SE; 10m for Stata/IC; fixed data area of 99 variables on roughly 1,200 observations for Small Stata  
Description: sets memory allocated to Stata's data areas. In Small Stata this cannot be changed.  
Reference:    [\[D\] memory](#)

**set more**

Syntax:      `set more {on|off} [ , permanently ]`  
Default:      on  
Description: pauses when `—more—` is displayed, continuing only when the user presses a key.  
Reference:    [\[R\] more](#)

**set notifyuser (Mac only)**

Syntax:      `set notifyuser {on|off} [ , permanently ]`  
Default:      on  
Description: sets the default Notification Manager behavior in Stata.

**set obs**

Syntax:      `set obs #`  
Default:      current number of observations  
Description: changes the number of observations in the current dataset. `#` must be at least as large as the current number of observations. If there are variables in memory, the values of all new observations are set to *missing*.  
Reference:    [\[D\] obs](#)

**set odbcmgr (Unix only)**

Syntax:      `set odbcmgr {iodbc|unixodbc} [ , permanently ]`  
Default:      `iodbc`  
Description: determines whether iODBC or unixODBC is your ODBC driver manager.  
Reference:    [\[D\] odbc](#)

**set output**

Syntax:      `set output {proc|inform|error}`  
Default:      `proc`  
Description: specifies the output to be displayed. `proc` means display all output; `inform` suppresses procedure output but displays informative messages and error messages; `error` suppresses all output except error messages. `set output` is seldom used.  
Reference:    [\[P\] quietly](#)

**set pagesize**

Syntax:     **set pagesize #**  
Default:     2 less than the physical number of lines on the screen  
Description: sets the number of lines between **—more—** messages.  
Reference:   [\[R\] more](#)

**set persistfv** (Windows only)

Syntax:     **set persistfv {on|off}**  
Default:     off  
Description: determines whether floating viewers are to persist between Stata sessions or when saving and loading a windowing preference.

**set persistvtopic** (Windows only)

Syntax:     **set persistvtopic {on|off}**  
Default:     off  
Description: determines whether viewer topics are to persist between Stata sessions or when saving and loading a windowing preference.

**set pinnable** (Windows only)

Syntax:     **set pinnable {on|off} [ , permanently ]**  
Default:     off  
Description: determines whether to enable the use of pinnable window characteristics for certain windows in Stata.

**set playsnd** (Mac only)

Syntax:     **set playsnd {on|off} [ , permanently ]**  
Default:     on  
Description: sets the sound behavior for the Notification Manager behavior in Stata.

**set printcolor**

Syntax:     **set printcolor { automatic | asis | gs1 | gs2 | gs3 } [ , permanently ]**  
Default:     automatic  
Description: determines how colors are handled when graphs are printed.  
Reference:   [\[G\] set printcolor](#)

**set processors**

Syntax:     **set processors #**  
Description: sets the number of processors or cores that Stata/MP will use. The default is the number of processors available on the computer, or the number of processors allowed by Stata/MP's license, whichever is less.

**set reventries**

Syntax:     **set reventries # [ , permanently ]**  
Default:     5000  
Description: sets the number of scrollbar lines available in the Review window.  
               $5 \leq \# \leq 32000$ .

**set revkeyboard** (Mac only)

Syntax:      `set revkeyboard {on|off} [, permanently]`

Default:      `on`

Description: sets the keyboard navigation behavior for the Review window. `on` indicates that you can use the keyboard to navigate and enter items from the Review window into the Command window. `off` indicates that all keyboard input be directed at the Command window; items can be entered from the Review window only by using the mouse.

**set rmsg**

Syntax:      `set rmsg {on|off} [, permanently]`

Default:      `off`

Description: indicates whether a return message telling the execution time is to be displayed at the completion of each command.

Reference:    [[P](#)] [rmsg](#)

**set scheme**

Syntax:      `set scheme schemename [, permanently]`

Default:      `s2color`

Description: determines the overall look for graphs.

Reference:    [[G](#)] [set scheme](#)

**set scrollbufsize**

Syntax:      `set scrollbufsize #`

Default:      `200000`

Description: sets the scrollbar buffer size, in bytes, for the Results window; may be set between 10,000 and 2,000,000.

**set searchdefault**

Syntax:      `set searchdefault {local|net|all} [, permanently]`

Default:      `local`

Description: sets the default behavior of the `search` command. `set searchdefault local` restricts `search` to use only Stata's keyword database. `set searchdefault net` restricts `search` to searching only the Internet. `set searchdefault all` indicates that both the keyword database and the Internet are to be searched.

Reference:    [[R](#)] [search](#)

**set seed**

Syntax:      `set seed {#|code}`

Default:      `123456789`

Description: specifies initial value of the random-number seed used by the `runiform()` function.

Reference:    [[R](#)] [set seed](#)

**set smoothfonts** (Mac only)

Syntax:      `set smoothfonts {on|off}`

Default:      `on`

Description: determines whether to use font smoothing (antialiased text) in the Results, Viewer, and Data Editor windows.

**set timeout1**

Syntax: `set timeout1 #seconds` [`,` `permanently`]

Default: 120

Description: sets the number of seconds Stata will wait for a remote host to respond to an initial contact before giving up. In general, users should not modify this value unless instructed to do so by Stata Technical Services.

Reference: [\[R\] netio](#)

**set timeout2**

Syntax: `set timeout2 #seconds` [`,` `permanently`]

Default: 300

Description: sets the number of seconds Stata will keep trying to get information from a remote host after initial contact before giving up. In general, users should not modify this value unless instructed to do so by Stata Technical Services.

Reference: [\[R\] netio](#)

**set trace**

Syntax: `set trace {on|off}`

Default: off

Description: determines whether to trace the execution of programs for debugging.

Reference: [\[P\] trace](#)

**set tracedepth**

Syntax: `set tracedepth #`

Default: 32000 (equivalent to  $\infty$ )

Description: if `trace` is set on, traces execution of programs and nested programs up to `tracedepth`. For example, if `tracedepth` is 2, the current program and any subroutine called would be traced, but subroutines of subroutines would not be traced.

Reference: [\[P\] trace](#)

**set traceexpand**

Syntax: `set traceexpand {on|off}` [`,` `permanently`]

Default: on

Description: if `trace` is set on, shows lines both before and after macro expansion. If `traceexpand` is set off, only the line before macro expansion is shown.

Reference: [\[P\] trace](#)

**set tracehilite**

Syntax: `set tracehilite "pattern"` [`,` `word`]

Default: ""

Description: highlights *pattern* in the trace output.

Reference: [\[P\] trace](#)

**set traceindent**

Syntax: `set traceindent {on|off}` [`,` `permanently`]

Default: on

Description: if `trace` is set on, indents displayed lines according to their nesting level. The lines of the main program are not indented. Two spaces of indentation are used for each level of nested subroutine.

Reference: [\[P\] trace](#)

**set tracenumber**

Syntax:      **set** tracenumber {on|off} [ , permanently ]

Default:      off

Description: if **trace** is set on, shows the nesting level numerically in front of the line. Lines of the main program are preceded by 01, lines of subroutines called by the main program are preceded by 02, etc.

Reference:    [P] **trace**

**set tracesep**

Syntax:      **set** tracesep {on|off} [ , permanently ]

Default:      on

Description: if **trace** is set on, displays a horizontal separator line that displays the name of the subroutine whenever a subroutine is called or exits.

Reference:    [P] **trace**

**set type**

Syntax:      **set** type {float|double} [ , permanently ]

Default:      float

Description: specifies the default storage type assigned to new variables.

Reference:    [D] **generate**

**set update\_interval** (Mac and Windows only)

Syntax:      **set** update\_interval #

Default:      7

Description: sets the number of days to elapse before performing the next automatic update query.

Reference:    [R] **update**

**set update\_prompt** (Mac and Windows only)

Syntax:      **set** update\_prompt {on|off}

Default:      on

Description: determines wheter a dialog is to be displayed before performing an automatic update query. There is no **permanently** option because **permanently** is implied.

Reference:    [R] **update**

**set update\_query** (Mac and Windows only)

Syntax:      **set** update\_query {on|off}

Default:      on

Description: determines whether update query is to be automatically performed when Stata is launched. There is no **permanently** option because **permanently** is implied.

Reference:    [R] **update**

**set varabbrev**

Syntax:      **set** varabbrev {on|off} [ , permanently ]

Default:      on

Description: indicates whether Stata should allow variable abbreviations.

Reference:    [P] **varabbrev**

`set varkeyboard` (Mac only)

Syntax: `set varkeyboard {on|off} [, permanently]`

Default: `on`

Description: sets the keyboard navigation behavior for the Variables window. `on` indicates that you can use the keyboard to navigate and enter items from the Variables window into the Command window. `off` indicates that all keyboard input be directed at the Command window; items can be entered from the Variables window only by using the mouse.

`set varlabelpos` (Unix only)

Syntax: `set varlabelpos #`

Default: `32`

Description: sets the maximum number of characters to display for variable names in the Variables window.  $8 \leq \# \leq 32$ .

`set virtual`

Syntax: `set virtual {on|off}`

Default: `off`

Description: indicates whether Stata should work to arrange its memory to keep objects close together.

Reference: [\[D\] memory](#)

## Also see

[\[R\] query](#) — Display system parameters

[\[R\] set\\_defaults](#) — Reset system parameters to original Stata defaults

[\[M-3\] mata set](#) — Set and display Mata system parameters

# Title

**set\_defaults** — Reset system parameters to original Stata defaults

## Syntax

```
set_defaults { category | _all } [, permanently]
```

where *category* is one of memory | output | interface | graphics | efficiency | network | update | trace | mata | other

## Description

`set_defaults` resets settings made by `set` to the original default settings that were shipped with Stata.

`set_defaults _all` resets all the categories, whereas `set defaults category` resets only the settings for the specified category.

## Option

`permanently` specifies that, in addition to making the change right now, the settings be remembered and become the default settings when you invoke Stata.

## Remarks

### ► Example 1

To assist us in debugging a new command, we modified some of the `trace` settings. To return them to their original values, we type

```
. set_defaults trace
-> set trace off
-> set tracedepth 32000
-> set traceexpand on
-> set tracesep on
-> set traceindent on
-> set tracenum off
-> set tracehilite ""
(preferences reset)
```

◀

## Methods and formulas

`set_defaults` is implemented as an ado-file.



## Also see

[R] [query](#) — Display system parameters

[R] [set](#) — Overview of system parameters

[M-3] [mata set](#) — Set and display Mata system parameters

# Title

**set emptycells** — Set what to do with empty cells in interactions

## Syntax

```
set emptycells { keep|drop } [, permanently]
```

## Description

`set emptycells` allows you to control how Stata handles interaction terms with empty cells. Stata can keep empty cells or drop them. The default is to keep empty cells.

## Option

`permanently` specifies that, in addition to making the change right now, the setting be remembered and become the default setting when you invoke Stata.

## Remarks

By default, Stata keeps empty cells so they can be reported in the coefficient table. For example, type

```
. use http://www.stata-press.com/data/r11/auto
. regress mpg rep78#foreign, baselevels
```

and you will see a regression of `mpg` on 10 indicator variables because `rep78` takes on 5 values and `foreign` takes on 2 values in the `auto` dataset. Two of those cells will be reported as empty because the data contain no observations of foreign cars with a `rep78` value of 1 or 2.

Many real datasets contain a large number of empty cells, and this could cause the “matsize too small” error, `r(908)`. In that case, type

```
. set emptycells drop
```

to get Stata to drop empty cells from the list of coefficients. If you commonly fit models with empty cells, you can permanently set Stata to drop empty cells by typing the following:

```
. set emptycells drop, permanently
```

## Also see

[R] [set](#) — Overview of system parameters

# Title

**set seed** — Specify initial value of random-number seed

## Syntax

```
set seed {#|code}
```

## Description

`set seed #` specifies the initial value of the random-number seed used by the `runiform()` function; see [\[D\] functions](#). `#` should be a positive integer. You may also specify *code* to indicate a previous internal state.

`set seed code` specifies a string that records the state of the random-number generator `runiform()`. You can obtain the current state of the random-number generator from `c(seed)`.

## Remarks

`set seed {#|code}` reinitializes the seed for the pseudorandom number generator `runiform()` and should be specified as a positive integer or as a *code*, indicating a previous internal state obtained from `c(seed)`. (Concerning `#`s: If a negative number is specified, it will be made positive.) The seed determines the sequence of pseudorandom numbers produced by `runiform()`. When you start Stata, it initializes the seed to 123456789, an easy-to-remember nonrandom number that in no way affects the pseudorandomness of the numbers produced by `runiform()`. Because Stata initializes the seed to a constant, Stata produces the same sequence of random numbers (measured from the start of the session) unless you reinitialize the seed. The seed you select affects only the sequence, not the pseudorandomness. Without loss of pseudorandomness, the seed may be set to small numbers; e.g., `set seed 2`.

See [\[D\] functions](#) for details and formulas.

### □ Technical note

Suppose that you just invoked Stata and typed the following:

```
. display c(seed)
X075bcd151f123bb5159a55e50022865746ad
```

The `c`-class macro `c(seed)` contains the current value of `set seed`. The code `X075bcd151f123bb5159a55e50022865746ad` is equivalent to 123456789. Each time you generate a random number, the internal state changes.

```
. generate u = runiform()
. display c(seed)
X167c0bd9c43f462544a474abacbdd93d03c8
```

You can use the previous contents of `c(seed)` to return `runiform()` to a previous state.

```
. local curseed = c(seed)
. generate x = runiform()
. etc.
. set seed 'curseed'
```



## Also see

[R] [set](#) — Overview of system parameters

[D] [functions](#) — Functions

# Title

signrank — Equality tests on matched data

## Syntax

*Wilcoxon matched-pairs signed-ranks test*

```
signrank varname = exp [if] [in]
```

*Sign test of matched pairs*

```
signtest varname = exp [if] [in]
```

by is allowed with `signrank` and `signtest`; see [D] [by](#).

## Menu

### signrank

Statistics > Nonparametric analysis > Tests of hypotheses > Wilcoxon matched-pairs signed-rank test

### signtest

Statistics > Nonparametric analysis > Tests of hypotheses > Test equality of matched pairs

## Description

`signrank` tests the equality of matched pairs of observations by using the Wilcoxon matched-pairs signed-ranks test ([Wilcoxon 1945](#)). The null hypothesis is that both distributions are the same.

`signtest` also tests the equality of matched pairs of observations ([Arbuthnott 1710](#)), but better explained by [Snedecor and Cochran 1989](#)) by calculating the differences between *varname* and the expression. The null hypothesis is that the median of the differences is zero; no further assumptions are made about the distributions. This, in turn, is equivalent to the hypothesis that the true proportion of positive (negative) signs is one-half.

For equality tests on unmatched data, see [R] [ranksum](#).

## Remarks

### ► Example 1: signrank

We are testing the effectiveness of a new fuel additive. We run an experiment with 12 cars. We first run each car without the fuel treatment and measure the mileage. We then add the fuel treatment and repeat the experiment. The results of the experiment are

| Without<br>treatment | With<br>treatment | Without<br>treatment | With<br>treatment |
|----------------------|-------------------|----------------------|-------------------|
| 20                   | 24                | 18                   | 17                |
| 23                   | 25                | 24                   | 28                |
| 21                   | 21                | 20                   | 24                |
| 25                   | 22                | 24                   | 27                |
| 18                   | 23                | 23                   | 21                |
| 17                   | 18                | 19                   | 23                |

We create two variables called `mpg1` and `mpg2`, representing mileage without and with the treatment, respectively. We can test the null hypothesis that the treatment had no effect by typing

```
. use http://www.stata-press.com/data/r11/fuel
. signrank mpg1=mpg2
Wilcoxon signed-rank test
```

| sign     | obs | sum ranks | expected |
|----------|-----|-----------|----------|
| positive | 3   | 13.5      | 38.5     |
| negative | 8   | 63.5      | 38.5     |
| zero     | 1   | 1         | 1        |
| all      | 12  | 78        | 78       |

```
unadjusted variance 162.50
adjustment for ties -1.62
adjustment for zeros -0.25

adjusted variance 160.62
Ho: mpg1 = mpg2
 z = -1.973
 Prob > |z| = 0.0485
```

The output indicates that we can reject the null hypothesis at any level above 4.85%.

➤ Example 2: `signtest`

`signtest` tests that the median of the differences is zero, making no further assumptions, whereas `signrank` assumed that the distributions are equal as well. Using the data above, we type

```
. signtest mpg1=mpg2
Sign test
```

| sign     | observed | expected |
|----------|----------|----------|
| positive | 3        | 5.5      |
| negative | 8        | 5.5      |
| zero     | 1        | 1        |
| all      | 12       | 12       |

```
One-sided tests:
Ho: median of mpg1 - mpg2 = 0 vs.
Ha: median of mpg1 - mpg2 > 0
 Pr(#positive >= 3) =
 Binomial(n = 11, x >= 3, p = 0.5) = 0.9673
Ho: median of mpg1 - mpg2 = 0 vs.
Ha: median of mpg1 - mpg2 < 0
 Pr(#negative >= 8) =
 Binomial(n = 11, x >= 8, p = 0.5) = 0.1133
```

```
Two-sided test:
Ho: median of mpg1 - mpg2 = 0 vs.
Ha: median of mpg1 - mpg2 != 0
Pr(#positive >= 8 or #negative >= 8) =
 min(1, 2*Binomial(n = 11, x >= 8, p = 0.5)) = 0.2266
```

The summary table indicates that there were three comparisons for which mpg1 exceeded mpg2, eight comparisons for which mpg2 exceeded mpg1, and one comparison for which they were the same.

The output below the summary table is based on the binomial distribution. The significance of the one-sided test, where the alternative hypothesis is that the median of mpg2 – mpg1 is greater than zero, is 0.1133. The significance of the two-sided test, where the alternative hypothesis is simply that the median of the differences is different from zero, is  $0.2266 = 2 \times 0.1133$ .

◀

## Saved results

`signrank` saves the following in `r()`:

Scalars

|                         |                                |                         |                           |
|-------------------------|--------------------------------|-------------------------|---------------------------|
| <code>r(N_neg)</code>   | number of negative comparisons | <code>r(sum_neg)</code> | sum of the negative ranks |
| <code>r(N_pos)</code>   | number of positive comparisons | <code>r(z)</code>       | $z$ statistic             |
| <code>r(N_tie)</code>   | number of tied comparisons     | <code>r(Var_a)</code>   | adjusted variance         |
| <code>r(sum_pos)</code> | sum of the positive ranks      |                         |                           |

`signtest` saves the following in `r()`:

Scalars

|                       |                                |                       |                                              |
|-----------------------|--------------------------------|-----------------------|----------------------------------------------|
| <code>r(N_neg)</code> | number of negative comparisons | <code>r(p_2)</code>   | two-sided probability                        |
| <code>r(N_pos)</code> | number of positive comparisons | <code>r(p_neg)</code> | one-sided probability of negative comparison |
| <code>r(N_tie)</code> | number of tied comparisons     | <code>r(p_pos)</code> | one-sided probability of positive comparison |

## Methods and formulas

`signrank` and `signtest` are implemented as ado-files.

For a practical introduction to these techniques with an emphasis on examples rather than theory, see [Bland \(2000\)](#) or [Sprent and Smeeton \(2007\)](#). For a summary of these tests, see [Snedecor and Cochran \(1989\)](#).

Methods and formulas are presented under the following headings:

*signrank*  
*signtest*

## signrank

Both the sign test and Wilcoxon signed-rank tests test the null hypothesis that the distribution of a random variable  $D = \text{varname} - \text{exp}$  has median zero. The sign test makes no additional assumptions, but the Wilcoxon signed-rank test makes the additional assumption that the distribution of  $D$  is symmetric. If  $D = X_1 - X_2$ , where  $X_1$  and  $X_2$  have the same distribution, then it follows that the distribution of  $D$  is symmetric about zero. Thus the Wilcoxon signed-rank test is often described as a test of the hypothesis that two distributions are the same, i.e.,  $X_1 \sim X_2$ .

Let  $d_j$  denote the difference for any matched pair of observations,

$$d_j = x_{1j} - x_{2j} = \text{varname} - \text{exp}$$

for  $j = 1, 2, \dots, n$ .

Rank the absolute values of the differences,  $|d_j|$ , and assign any tied values the average rank. Consider the signs of  $d_j$ , and let

$$r_j = \text{sign}(d_j) \text{rank}(|d_j|)$$

be the signed ranks. The test statistic is

$$T_{\text{obs}} = \sum_{j=1}^n r_j = (\text{sum of ranks for } + \text{ signs}) - (\text{sum of ranks for } - \text{ signs})$$

The null hypothesis is that the distribution of  $d_j$  is symmetric about 0. Hence the likelihood is unchanged if we flip signs on the  $d_j$ , and thus the randomization datasets are the  $2^n$  possible sign changes for the  $d_j$ . Thus the randomization distribution of our test statistic  $T$  can be computed by considering all the  $2^n$  possible values of

$$T = \sum_{j=1}^n S_j r_j$$

where the  $r_j$  are the observed signed ranks (considered fixed) and  $S_j$  is either  $+1$  or  $-1$ .

With this distribution, the mean and variance of  $T$  are given by

$$E(T) = 0 \quad \text{and} \quad \text{Var}_{\text{adj}}(T) = \sum_{j=1}^n r_j^2$$

The test statistic for the Wilcoxon signed-rank test is often expressed (equivalently) as the sum of the positive signed-ranks,  $T_+$ , where

$$E(T_+) = \frac{n(n+1)}{4} \quad \text{and} \quad \text{Var}_{\text{adj}}(T_+) = \frac{1}{4} \sum_{j=1}^n r_j^2$$

Zeros and ties do not affect the theory above, and the exact variance is still given by the above formula for  $\text{Var}_{\text{adj}}(T_+)$ . When  $d_j = 0$  is observed,  $d_j$  will always be zero in each of the randomization datasets (using  $\text{sign}(0) = 0$ ). When there are ties, you can assign averaged ranks for each group of ties and then treat them the same as the other ranks.

The “unadjusted variance” reported by **signrank** is the variance that the randomization distribution would have had if there had been no ties or zeros:

$$\text{Var}_{\text{unadj}}(T_+) = \frac{1}{4} \sum_{j=1}^n j^2 = \frac{n(n+1)(2n+1)}{24}$$

The adjustment for zeros is the change in the variance when the ranks for the zeros are signed to make  $r_j = 0$ ,

$$\Delta \text{Var}_{\text{zero adj}}(T_+) = -\frac{1}{4} \sum_{j=1}^{n_0} j^2 = -\frac{n_0(n_0+1)(2n_0+1)}{24}$$

where  $n_0$  is the number of zeros. The adjustment for ties is the change in the variance when the ranks (for nonzero observations) are replaced by averaged ranks:

$$\Delta \text{Var}_{\text{ties adj}}(T_+) = \text{Var}_{\text{adj}}(T_+) - \text{Var}_{\text{unadj}}(T_+) - \Delta \text{Var}_{\text{zero adj}}(T_+)$$

A normal approximation is used to calculate

$$z = \frac{T_+ - E(T_+)}{\sqrt{\text{Var}_{\text{adj}}(T_+)}}$$



## signtest

The test statistic for the sign test is the number  $n_+$  of differences

$$d_j = x_{1j} - x_{2j} = \text{varname} - \text{exp}$$

greater than zero. Assuming that the probability of a difference being equal to zero is exactly zero, then, under the null hypothesis,  $n_+ \sim \text{binomial}(n, p = 1/2)$ , where  $n$  is the total number of observations.

But what if some differences are zero? This question has a ready answer if you view the test from the perspective of Fisher’s Principle of Randomization (Fisher 1935). Fisher’s idea (stated in a modern way) was to look at a family of transformations of the observed data such that the a priori likelihood (under the null hypothesis) of the transformed data is the same as the likelihood of the observed data. The distribution of the test statistic is then produced by calculating its value for each of the transformed “randomization” datasets, assuming that each dataset is equally likely.

For the sign test, the “data” are simply the set of signs of the differences. Under the null hypothesis of the sign test, the probability that  $d_j$  is less than zero is equal to the probability that  $d_j$  is greater than zero. Thus you can transform the observed signs by flipping any number of them, and the set of signs will have the same likelihood. The  $2^n$  possible sign changes form the family of randomization datasets. If you have no zeros, this procedure again leads to  $n_+ \sim \text{binomial}(n, p = 1/2)$ .

If you do have zeros, changing their signs leaves them as zeros. So, if you observe  $n_0$  zeros, each of the  $2^n$  sign-change datasets will also have  $n_0$  zeros. Hence, the values of  $n_+$  calculated over the sign-change datasets range from 0 to  $n - n_0$ , and the “randomization” distribution of  $n_+$  is  $\text{binomial}(n - n_0, p = 1/2)$ .

The work of Arbuthnott (1710) and later eighteenth-century contributions is discussed by Hald (2003, chap. 17).

Frank Wilcoxon (1892–1965) was born in Ireland to American parents. After working in various occupations (including merchant seaman, oil-well pump attendant, and tree surgeon), he settled in chemistry, gaining degrees from Rutgers and Cornell and employment from various companies. Working mainly on the development of fungicides and insecticides, Wilcoxon became interested in statistics in 1925 and made several key contributions to nonparametric methods. After retiring from industry, he taught statistics at Florida State until his death.

## References

- Arbuthnott, J. 1710. An argument for divine providence, taken from the constant regularity observed in the births of both sexes. *Philosophical Transaction of the Royal Society of London* 27: 186–190.
- Bland, M. 2000. *An Introduction to Medical Statistics*. 3rd ed. Oxford: Oxford University Press.
- Bradley, R. A. 2001. Frank Wilcoxon. In *Statisticians of the Centuries*, ed. C. C. Heyde and E. Seneta, 420–424. New York: Springer.
- Fisher, R. A. 1935. *The Design of Experiments*. Edinburgh: Oliver & Boyd.
- Hald, A. 2003. *A History of Probability and Statistics and Their Applications before 1750*. New York: Wiley.
- Kaiser, J. 2007. An exact and a Monte Carlo proposal to the Fisher–Pitman permutation tests for paired replicates and for independent samples. *Stata Journal* 7: 402–412.
- Newson, R. 2006. Confidence intervals for rank statistics: Somers’ D and extensions. *Stata Journal* 6: 309–334.
- Snedecor, G. W., and W. G. Cochran. 1989. *Statistical Methods*. 8th ed. Ames, IA: Iowa State University Press.

- Sprent, P., and N. C. Smeeton. 2007. *Applied Nonparametric Statistical Methods*. 4th ed. Boca Raton, FL: Chapman & Hall/CRC.
- Sribney, W. M. 1995. crc40: Correcting for ties and zeros in sign and rank tests. *Stata Technical Bulletin* 26: 2–4. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 5–8. College Station, TX: Stata Press.
- Wilcoxon, F. 1945. Individual comparisons by ranking methods. *Biometrics* 1: 80–83.

## Also see

- [R] **ranksum** — Equality tests on unmatched data
- [R] **ttest** — Mean-comparison tests

## Syntax

`simulate` [*exp\_list*] , `reps`(#) [*options*] : *command*

| <i>options</i>                               | description                            |
|----------------------------------------------|----------------------------------------|
| <code>nodots</code>                          | suppress the replication dots          |
| <code>noisily</code>                         | display any output from <i>command</i> |
| <code>trace</code>                           | trace the <i>command</i>               |
| <code>saving</code> ( <i>filename</i> , ...) | save results to <i>filename</i>        |
| <code>nolegend</code>                        | suppress table legend                  |
| <code>verbose</code>                         | display the full table legend          |
| <code>seed</code> (#)                        | set random-number seed to #            |

All weight types supported by *command* are allowed; see [U] 11.1.6 **weight**.

|                          |                                                                                |
|--------------------------|--------------------------------------------------------------------------------|
| <i>exp_list</i> contains | ( <i>name</i> : <i>elist</i> )<br><i>elist</i><br><i>eexp</i>                  |
| <i>elist</i> contains    | <i>newvar</i> = ( <i>exp</i> )<br>( <i>exp</i> )                               |
| <i>eexp</i> is           | <i>specname</i><br>[ <i>eqno</i> ] <i>specname</i>                             |
| <i>specname</i> is       | <code>_b</code><br><code>_b[]</code><br><code>_se</code><br><code>_se[]</code> |
| <i>eqno</i> is           | <code>##</code><br><i>name</i>                                                 |

*exp* is a standard Stata expression; see [U] 13 **Functions and expressions**.

Distinguish between `[]`, which are to be typed, and `[]`, which indicate optional arguments.

## Description

`simulate` eases the programming task of performing Monte Carlo–type simulations. Typing  
`. simulate exp_list, reps(#): command`

runs *command* for # replications and collects the results in *exp\_list*.

*command* defines the command that performs one simulation. Most Stata commands and user-written programs can be used with `simulate`, as long as they follow standard Stata syntax; see [U] 11 **Language syntax**. The `by` prefix may not be part of *command*.

*exp\_list* specifies the expression to be calculated from the execution of *command*. If no expressions are given, *exp\_list* assumes a default, depending upon whether *command* changes results in `e()` or `r()`. If *command* changes results in `e()`, the default is `_b`. If *command* changes results in `r()` (but not `e()`), the default is all the scalars posted to `r()`. It is an error not to specify an expression in *exp\_list* otherwise.

## Options

`reps(#)` is required—it specifies the number of replications to be performed.

`nodots` suppresses display of the replication dots. By default, one dot character is displayed for each successful replication. A red ‘x’ is displayed if *command* returns an error or if one of the values in *exp\_list* is missing.

`noisily` requests that any output from *command* be displayed. This option implies the `nodots` option.

`trace` causes a trace of the execution of *command* to be displayed. This option implies the `noisily` option.

`saving(filename[ , suboptions])` creates a Stata data file (`.dta` file) consisting of, for each statistic in *exp\_list*, a variable containing the simulated values.

`double` specifies that the results for each replication be stored as `doubles`, meaning 8-byte reals.

By default, they are stored as `floats`, meaning 4-byte reals.

`every(#)` specifies that results be written to disk every *#*th replication. `every()` should be specified only in conjunction with `saving()` when *command* takes a long time for each replication. This will allow recovery of partial results should some other software crash your computer. See [\[P\] postfile](#).

`replace` specifies that *filename* be overwritten if it exists.

`nolegend` suppresses display of the table legend. The table legend identifies the rows of the table with the expressions they represent.

`verbose` requests that the full table legend be displayed. By default, coefficients and standard errors are not displayed.

`seed(#)` sets the random-number seed. Specifying this option is equivalent to typing the following command before calling `simulate`:

```
. set seed #
```

## Remarks

For an introduction to Monte Carlo methods, see [Cameron and Trivedi \(2009, 113–145\)](#).

### ► Example 1

We have a dataset containing means and variances of 100-observation samples from a lognormal distribution (as a first step in evaluating, say, the coverage of a 95%, *t*-based confidence interval). Then we perform the experiment 1,000 times.

The following command definition will generate 100 independent observations from a lognormal distribution and compute the summary statistics for this sample.

```
program lnsim, rclass
 version 11
 drop _all
 set obs 100
 gen z = exp(rnormal())
 summarize z
 return scalar mean = r(mean)
 return scalar Var = r(Var)
end
```

We can save 1,000 simulated means and variances from lnsim by typing

```
. set seed 1234
. simulate mean=r(mean) var=r(Var), reps(1000) nodots: lnsim
 command: lnsim
 mean: r(mean)
 var: r(Var)
```

```
. describe *
```

| variable name | storage type | display format | value label | variable label |
|---------------|--------------|----------------|-------------|----------------|
| mean          | float        | %9.0g          |             | r(mean)        |
| var           | float        | %9.0g          |             | r(Var)         |

```
. summarize
```

| Variable | Obs  | Mean     | Std. Dev. | Min      | Max      |
|----------|------|----------|-----------|----------|----------|
| mean     | 1000 | 1.658096 | .2256156  | 1.113111 | 3.920677 |
| var      | 1000 | 5.062774 | 12.58704  | 1.167172 | 384.8679 |

□ Technical note

Before executing our lnsim simulator, we can verify that it works by executing it interactively.

```
. set seed 1234
. lnsim
obs was 0, now 100
```

| Variable | Obs | Mean   | Std. Dev. | Min      | Max      |
|----------|-----|--------|-----------|----------|----------|
| z        | 100 | 1.7379 | 3.116258  | .1193447 | 26.30184 |

```
. return list
```

```
scalars:
 r(Var) = 9.711062636719475
 r(mean) = 1.737900345399976
```

➤ Example 2

Consider a more complicated problem. Let’s experiment with fitting  $y_j = a + bx_j + u_j$  when the true model has  $a = 1$ ,  $b = 2$ ,  $u_j = z_j + cx_j$ , and when  $z_j$  is  $N(0, 1)$ . We will save the parameter estimates and standard errors and experiment with varying  $c$ .  $x_j$  will be fixed across experiments but will originally be generated as  $N(0, 1)$ . We begin by interactively making the true data:

```
. drop _all
. set obs 100
obs was 0, now 100
```

```
. set seed 54321
. gen x = rnormal()
. gen true_y = 1+2*x
. save truth
file truth.dta saved
```

Our program is

```
program hetero1
 version 11
 args c
 use truth, clear
 gen y = true_y + (rnormal() + 'c'*x)
 regress y x
end
```

Note the use of `'c'` in our statement for generating `y`. `c` is a local macro generated from `args c` and thus refers to the first argument supplied to `hetero1`. If we want  $c = 3$  for our experiment, we type

```
. simulate _b _se, reps(10000): hetero1 3
(output omitted)
```

Our program `hetero1` could, however, be more efficient because it rereads the file `truth` once every replication. It would be better if we could read the data just once. In fact, if we read in the data right before running `simulate`, we really should not have to reread for each subsequent replication. A faster version reads

```
program hetero2
 version 11
 args c
 capture drop y
 gen y = true_y + (rnormal() + 'c'*x)
 regress y x
end
```

Requiring that the current dataset has the variables `true_y` and `x` may become inconvenient. Another improvement would be to require that the user supply variable names, such as in

```
program hetero3
 version 11
 args truey x c
 capture drop y
 gen y = 'truey' + (rnormal() + 'c'*'x')
 regress y x
end
```

Thus we can type

```
. simulate _b _se, reps(10000): hetero3 true_y x 3
(output omitted)
```

◀

## ► Example 3

Now let's consider the problem of simulating the ratio of two medians. Suppose that each sample of size  $n_i$  comes from a normal population with a mean  $\mu_i$  and standard deviation  $\sigma_i$ , where  $i = 1, 2$ . We write the program below and save it as a text file called `myratio.ado` (see [U] 17 [Ado-files](#)). Our program is an `rclass` command that requires six arguments as input, identified by the local macros `n1`, `mu1`, `sigma1`, `n2`, `mu2`, and `sigma2`, which correspond to  $n_1$ ,  $\mu_1$ ,  $\sigma_1$ ,  $n_2$ ,  $\mu_2$ , and  $\sigma_2$ , respectively. With these arguments, `myratio` will generate the data for the two samples, use `summarize` to compute the two medians and save the ratio of the medians in `r(ratio)`.

```

program myratio, rclass
 version 11
 args n1 mu1 sigma1 n2 mu2 sigma2
 // generate the data
 drop _all
 local N = 'n1'+`n2'
 set obs `N'
 tempvar y
 generate `y' = rnormal()
 replace `y' = cond(_n<=`n1', `mu1'+`y'*`sigma1', `mu2'+`y'*`sigma2')
 // calculate the medians
 tempname m1
 summarize `y' if _n<=`n1', detail
 scalar `m1' = r(p50)
 summarize `y' if _n>`n1', detail
 // save the results
 return scalar ratio = `m1' / r(p50)
end

```

The result of running our simulation is

```

. set seed 19192
. simulate ratio=r(ratio), reps(1000) nodots: myratio 5 3 1 10 3 2
 command: myratio 5 3 1 10 3 2
 ratio: r(ratio)
. summarize

```

| Variable | Obs  | Mean     | Std. Dev. | Min      | Max      |
|----------|------|----------|-----------|----------|----------|
| ratio    | 1000 | 1.087339 | .4222457  | .4116514 | 4.876788 |

◀

## □ Technical note

Stata lets us do simulations of simulations and simulations of bootstraps. Stata's `bootstrap` command (see [R] [bootstrap](#)) works much like `simulate`, except that it feeds the user-written program a bootstrap sample. Say that we want to evaluate the bootstrap estimator of the standard error of the median when applied to lognormally distributed data. We want to perform a simulation, resulting in a dataset of medians and bootstrap estimated standard errors.

As background, `summarize` calculates summary statistics, leaving the mean in `r(mean)` and the variance in `r(Var)`. `summarize` with the `detail` option also calculates summary statistics, but more of them, and leaves the median in `r(p50)`.

Thus our plan is to perform simulations by randomly drawing a dataset: we calculate the median of our random sample, we use `bootstrap` to obtain a dataset of medians calculated from bootstrap samples of our random sample, the standard deviation of those medians is our estimate of the standard error, and the summary statistics are saved in the results of `summarize`.

Our simulator is

(Continued on next page)

```
program define bsse, rclass
 version 11
 drop _all
 set obs 100
 gen x = rnormal()
 tempfile bsfile
 bootstrap midp=r(p50), rep(100) saving('bsfile'): summarize x, detail
 use 'bsfile', clear
 summarize midp
 return scalar mean = r(mean)
 return scalar sd = r(sd)
end
```

We can obtain final results, running our simulation 1,000 times, by typing

```
. set seed 48901
. simulate med=r(mean) bs_se=r(sd), reps(1000): bsse
 command: bsse
 med: r(mean)
 bs se: r(sd)
```

Simulations (1000)

———— 1 ———— 2 ———— 3 ———— 4 ———— 5  
 ..... 50  
 ..... 100  
 ..... 150  
 ..... 200  
 (output omitted)  
 ..... 850  
 ..... 900  
 ..... 950  
 ..... 1000

```
. summarize
```

| Variable | Obs  | Mean      | Std. Dev. | Min       | Max      |
|----------|------|-----------|-----------|-----------|----------|
| med      | 1000 | -.0068042 | .1215801  | -.3862474 | .4906133 |
| bs_se    | 1000 | .1268423  | .0289583  | .049932   | .2226238 |

This is a case where the simulation dots (drawn by default, unless the `nodots` option is specified) will give us an idea of how long this simulation will take to finish as it runs. □



## Methods and formulas

`simulate` is implemented as an ado-file.

## References

- Cameron, A. C., and P. K. Trivedi. 2009. *Microeconometrics Using Stata*. College Station, TX: Stata Press.
- Gould, W. W. 1994. [ssi6.1: Simplified Monte Carlo simulations](#). *Stata Technical Bulletin* 20: 22–24. Reprinted in *Stata Technical Bulletin Reprints*, vol. 4, pp. 207–210. College Station, TX: Stata Press.
- Hamilton, L. C. 2009. *Statistics with Stata (Updated for Version 10)*. Belmont, CA: Brooks/Cole.
- Weesie, J. 1998. [ip25: Parameterized Monte Carlo simulations: Enhancement to the simulation command](#). *Stata Technical Bulletin* 43: 13–15. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 75–77. College Station, TX: Stata Press.



## Also see

[R] [bootstrap](#) — Bootstrap sampling and estimation

[R] [jackknife](#) — Jackknife estimation

[R] [permute](#) — Monte Carlo permutation tests

## Description

The *Stata Journal* (SJ) is a quarterly journal containing articles about statistics, data analysis, teaching methods, and effective use of Stata's language. The SJ publishes reviewed papers together with shorter notes and comments, regular columns, tips, book reviews, and other material of interest to researchers applying statistics in a variety of disciplines. You can read all about the *Stata Journal* at <http://www.stata-journal.com>.

The *Stata Journal* is a printed and electronic journal with corresponding software. If you want the journal, you must subscribe, but the software is available for no charge from our web site at <http://www.stata-journal.com>. PDF copies of SJ articles that are older than three years are available for download for no charge at <http://www.stata-journal.com/archives.html>. More recent articles may be individually purchased.

The predecessor to the *Stata Journal* was the *Stata Technical Bulletin* (STB). The STB was also a printed and electronic journal with corresponding software. PDF copies of all STB journals are available for download for no charge at <http://www.stata-press.com/journals/stbj.html>. The STB software is available for no charge from our web site at <http://www.stata.com>.

Below are instructions for installing the *Stata Journal* and the *Stata Technical Bulletin* software from our web site.

## Remarks

Remarks are presented under the following headings:

*Installing the Stata Journal software*

*Obtaining from the Internet by pointing and clicking*

*Obtaining from the Internet via command mode*

*Installing the STB software*

*Obtaining from the Internet by pointing and clicking*

*Obtaining from the Internet via command mode*

## Installing the Stata Journal software

Each issue of the *Stata Journal* is labeled Volume #, Number #. Volume 1 refers to the first year of publication, Volume 2 to the second, and so on. Issues are numbered 1, 2, 3, and 4 within each year. The first issue of the *Journal* was published in the fourth quarter of 2001, and that issue is numbered Volume 1, Number 1. For installation purposes, we refer to this issue as sj1-1.

The articles, columns, notes, and comments that make up the *Stata Journal* are assigned a letter-and-number code, called an insert tag, such as st0001, an0034, or ds0011. The letters represent a category: st is the statistics category, an is the announcements category, etc. The numbers are assigned sequentially, so st0001 is the first article in the statistics category.

Sometimes inserts are subsequently updated, either to fix bugs or to add new features. A number such as st0001\_1 indicates that this article, column, note, or comment is an update to the original st0001 article. Updates are complete; that is, installing st0001\_1 provides all the features of the original article and more.

The *Stata Journal* software may be obtained by pointing and clicking or by using command mode.

The sections below detail how to install an insert. In all cases, pretend that you wish to install insert `st0001_1` from `sj2-2`.

## Obtaining from the Internet by pointing and clicking

1. Select **Help > SJ and User-written Programs**.
2. Click on *Stata Journal*.
3. Click on *sj2-2*.
4. Click on *st0001\_1*.
5. Click on (*click here to install*).

## Obtaining from the Internet via command mode

Type the following:

```
. net from http://www.stata-journal.com/software
. net cd sj2-2
. net describe st0001_1
. net install st0001_1
```

The above could be shortened to

```
. net from http://www.stata-journal.com/software/sj2-2
. net describe st0001_1
. net install st0001_1
```

Alternatively, you could type

```
. net sj 2-2
. net describe st0001_1
. net install st0001_1
```

but going about it the long way is more entertaining, at least the first time.

## Installing the STB software

Each issue of the STB is numbered. STB-1 refers to the first issue (published May 1991), STB-2 refers to the second (published July 1991), and so on.

An issue of the STB consists of inserts—articles—and these are assigned letter-and-number combinations, such as `sg84`, `dm80`, `sbe26.1`, etc. The letters represent a category; for example, `sg` is the general statistics category and `dm` the data-management category. The numbers are assigned sequentially, so `sbe39` is the 39th insert in the biostatistics and epidemiology series.

Insert `sbe39`, it turns out, provides a method of accounting for publication bias in meta-analysis; it adds a new command called `metatrim` to Stata. If you installed `sbe39`, you would have that command and its online help. Insert `sbe39` was published in STB-57 (September 2000). Obtaining `metatrim` simply requires going to STB-57 and getting `sbe39`.

Sometimes inserts were subsequently updated, either to fix bugs or to add new features. `sbe39` was updated: the first update is `sbe39.1` and the second is `sbe39.2`. You could install insert `sbe39.2`, and it would not matter whether you had previously installed `sbe39.1`. Updates are complete: installing `sbe39.2` provides all the features of the original insert and more.

For computer naming purposes, insert sbe39.2 is referred to as `sbe39_2`. When referred to in normal text, however, the insert is still called sbe39.2 because that looks nicer.

Inserts are easily available from the Internet. Inserts may be obtained by pointing and clicking or by using command mode.

The sections below detail how to install an insert. In all cases, pretend that you wish to install insert sbe39.2 from STB-61.

## Obtaining from the Internet by pointing and clicking

1. Select **Help > SJ and User-written Programs**.
2. Click on *STB*.
3. Click on *stb61*.
4. Click on *sbe39\_2*.
5. Click on (*click here to install*).

## Obtaining from the Internet via command mode

Type the following:

```
. net from http://www.stata.com
. net cd stb
. net cd stb61
. net describe sbe39_2
. net install sbe39_2
```

The above could be shortened to

```
. net from http://www.stata.com/stb/stb61
. net describe sbe39_2
. net install sbe39_2
```

but going about it the long way is more entertaining, at least the first time.

## Also see

[R] [search](#) — Search Stata documentation

[R] [net](#) — Install and manage user-written additions from the Internet

[R] [net search](#) — Search the Internet for installable packages

[R] [update](#) — Update Stata

[U] [3.5 The Stata Journal](#)

[U] [28 Using the Internet to keep up to date](#)

[GSM] [19 Updating and extending Stata—Internet functionality](#)

[GSU] [19 Updating and extending Stata—Internet functionality](#)

[GSW] [19 Updating and extending Stata—Internet functionality](#)

Title

sktest — Skewness and kurtosis test for normality

Syntax

```
sktest varlist [if] [in] [weight] [, noadjust]
```

aweights and fweights are allowed; see [U] 11.1.6 weight.

Menu

Statistics > Summaries, tables, and tests > Distributional plots and tests > Skewness and kurtosis normality test

Description

For each variable in *varlist*, **sktest** presents a test for normality based on skewness and another based on kurtosis and then combines the two tests into an overall test statistic. **sktest** requires a minimum of 8 observations to make its calculations.

Option

Main

**noadjust** suppresses the empirical adjustment made by Royston (1991c) to the overall  $\chi^2$  and its significance level and presents the unaltered test as described by D’Agostino, Belanger, and D’Agostino Jr. (1990).

Remarks

Also see [R] **swilk** for the Shapiro–Wilk and Shapiro–Francia tests for normality. Those tests are, in general, preferred for nonaggregated data (Gould and Rogers 1991; Gould 1992; Royston 1991c). Moreover, a normal quantile plot should be used with any test for normality; see [R] **diagnostic plots** for more information.

➤ Example 1

Using our automobile dataset, we will test whether the variables **mpg** and **trunk** are normally distributed:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. sktest mpg trunk
```

| Skewness/Kurtosis tests for Normality |     |              |              |             |                 |
|---------------------------------------|-----|--------------|--------------|-------------|-----------------|
| Variable                              | Obs | Pr(Skewness) | Pr(Kurtosis) | adj chi2(2) | joint Prob>chi2 |
| mpg                                   | 74  | 0.0015       | 0.0804       | 10.95       | 0.0042          |
| trunk                                 | 74  | 0.9115       | 0.0445       | 4.19        | 0.1228          |

We can reject the hypothesis that `mpg` is normally distributed, but we cannot reject the hypothesis that `trunk` is normally distributed, at least at the 12% level. The kurtosis for `trunk` is 2.19, as can be verified by issuing the command

```
. summarize trunk, detail
(output omitted)
```

and the  $p$ -value of 0.0445 shown in the table above indicates that it is significantly different from the kurtosis of a normal distribution at the 5% significance level. However, on the basis of skewness alone, we cannot reject the hypothesis that `trunk` is normally distributed.



❑ Technical note

`sktest` implements the test as described by [D’Agostino, Belanger, and D’Agostino Jr. \(1990\)](#) but with the adjustment made by [Royston \(1991c\)](#). In the above example, if we had specified the `noadjust` option, the  $\chi^2$  values would have been 13.13 for `mpg` and 4.05 for `trunk`. With the adjustment, the  $\chi^2$  value might show as ‘.’. This result should be interpreted as an absurdly large number; the data are most certainly not normal.



## Saved results

`sktest` saves the following in `r()`:

|                        |                                              |
|------------------------|----------------------------------------------|
| Scalars                |                                              |
| <code>r(chi2)</code>   | $\chi^2$                                     |
| <code>r(P_skew)</code> | Pr(skewness)                                 |
| <code>r(P_kurt)</code> | Pr(kurtosis)                                 |
| <code>r(P_chi2)</code> | Prob > chi2                                  |
| Matrices               |                                              |
| <code>r(N)</code>      | matrix of observations                       |
| <code>r(Utest)</code>  | matrix of test results, one row per variable |

## Methods and formulas

`sktest` is implemented as an ado-file.

`sktest` implements the test described by [D’Agostino, Belanger, and D’Agostino Jr. \(1990\)](#) with the empirical correction developed by [Royston \(1991c\)](#).

Let  $g_1$  denote the coefficient of skewness and  $b_2$  denote the coefficient of kurtosis as calculated by `summarize`, and let  $n$  denote the sample size. If weights are specified, then  $g_1$ ,  $b_2$ , and  $n$  denote the weighted coefficients of skewness and kurtosis and weighted sample size, respectively. See [\[R\] summarize](#) for the formulas for skewness and kurtosis.

To perform the test of skewness, we compute

$$Y = g_1 \left\{ \frac{(n+1)(n+3)}{6(n-2)} \right\}^{1/2}$$

$$\beta_2(g_1) = \frac{3(n^2 + 27n - 70)(n+1)(n+3)}{(n-2)(n+5)(n+7)(n+9)}$$

$$W^2 = -1 + [2 \{\beta_2(g_1) - 1\}]^{1/2}$$

$$\alpha = \{2/(W^2 - 1)\}^{1/2}$$

and

Then the distribution of the test statistic

$$Z_1 = \frac{1}{\sqrt{\ln W}} \ln \left[ Y/\alpha + \{(Y/\alpha)^2 + 1\}^{1/2} \right]$$

is approximately standard normal under the null hypothesis that the data are distributed normally.

To perform the test of kurtosis, we compute

$$E(b_2) = \frac{3(n-1)}{n+1}$$

$$\text{var}(b_2) = \frac{24n(n-2)(n-3)}{(n+1)^2(n+3)(n+5)}$$

$$X = \{b_2 - E(b_2)\} / \sqrt{\text{var}(b_2)}$$

$$\sqrt{\beta_1(b_2)} = \frac{6(n^2 - 5n + 2)}{(n+7)(n+9)} \left\{ \frac{6(n+3)(n+5)}{n(n-2)(n-3)} \right\}^{1/2}$$

$$A = 6 + \frac{8}{\sqrt{\beta_1(b_2)}} \left[ \frac{2}{\sqrt{\beta_1(b_2)}} + \left\{ 1 + \frac{4}{\beta_1(b_2)} \right\}^{1/2} \right]$$

and

Then the distribution of the test statistic

$$Z_2 = \frac{1}{\sqrt{2/(9A)}} \left[ \left( 1 - \frac{2}{9A} \right) - \left\{ \frac{1 - 2/A}{1 + X\sqrt{2/(A-4)}} \right\}^{1/3} \right]$$

is approximately standard normal under the null hypothesis that the data are distributed normally.

D'Agostino, Balanger, and D'Agostino Jr.'s omnibus test of normality uses the statistic

$$K^2 = Z_1^2 + Z_2^2$$

which has approximately a  $\chi^2$  distribution with 2 degrees of freedom under the null of normality.

[Royston \(1991c\)](#) proposed the following adjustment to the test of normality, which `sktest` uses by default. Let  $\Phi(x)$  denote the cumulative standard normal distribution function for  $x$ , and let  $\Phi^{-1}(p)$  denote the inverse cumulative standard normal function [i.e.,  $x = \Phi^{-1}\{\Phi(x)\}$ ]. Define the following terms:

$$\begin{aligned}
Z_c &= -\Phi^{-1} \left\{ \exp \left( -\frac{1}{2} K^2 \right) \right\} \\
Z_t &= 0.55n^{0.2} - 0.21 \\
a_1 &= (-5 + 3.46 \ln n) \exp(-1.37 \ln n) \\
b_1 &= 1 + (0.854 - 0.148 \ln n) \exp(-0.55 \ln n) \\
a_2 &= a_1 - \{2.13/(1 - 2.37 \ln n)\} Z_t \\
b_2 &= 2.13/(1 - 2.37 \ln n) + b_1
\end{aligned}$$

and

If  $Z_c < -1$  set  $Z = Z_c$ ; else if  $Z_c < Z_t$  set  $Z = a_1 + b_1 Z_c$ ; else set  $Z = a_2 + b_2 Z_c$ . Define  $P = 1 - \Phi(Z)$ . Then  $K^2 = -2 \ln P$  is approximately distributed  $\chi^2$  with 2 degrees of freedom.

The relative merits of the skewness and kurtosis test versus the Shapiro–Wilk and Shapiro–Francia tests have been a subject of debate. The interested reader is directed to the articles in the *Stata Technical Bulletin*. Our recommendation is to use the Shapiro–Francia test whenever possible, that is, whenever dealing with nonaggregated or ungrouped data (Gould and Rogers 1991; Gould 1992); see [R] [swilk](#). If normality is rejected, use `sktest` to determine the source of the problem.

As both D’Agostino, Belanger, and D’Agostino Jr. (1990) and Royston (1991d) mention, researchers should also examine the normal quantile plot to determine normality rather than blindly relying on a few test statistics. See the `qnorm` command documented in [R] [diagnostic plots](#) for more information on normal quantile plots.

`sktest` is similar in spirit to the Jarque–Bera (1987) test of normality. The Jarque–Bera test statistic is also calculated from the sample skewness and kurtosis, though it is based on asymptotic standard errors with no corrections for sample size. In effect, `sktest` offers two adjustments for sample size, that of Royston (1991c) and that of D’Agostino, Belanger, and D’Agostino Jr. (1990).

## Acknowledgments

`sktest` has benefited greatly by the comments and work of Patrick Royston of the MRC Clinical Trials Unit, London; at this point, the program should be viewed as due as much to Royston as to us, except, of course, for any errors. We are also indebted to Nicholas J. Cox, Durham University, for helpful comments.

## References

- D’Agostino, R. B., A. J. Belanger, and R. B. D’Agostino Jr. 1990. A suggestion for using powerful and informative tests of normality. *American Statistician* 44: 316–321.
- . 1991. [sg3.3: Comment on tests of normality](#). *Stata Technical Bulletin* 3: 20. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 105–106. College Station, TX: Stata Press.
- Gould, W. W. 1991. [sg3: Skewness and kurtosis tests of normality](#). *Stata Technical Bulletin* 1: 20–21. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 99–101. College Station, TX: Stata Press.
- . 1992. [sg11.1: Quantile regression with bootstrapped standard errors](#). *Stata Technical Bulletin* 9: 19–21. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 137–139. College Station, TX: Stata Press.
- Gould, W. W., and W. H. Rogers. 1991. [sg3.4: Summary of tests of normality](#). *Stata Technical Bulletin* 3: 20–23. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 106–110. College Station, TX: Stata Press.
- Jarque, C. M., and A. K. Bera. 1987. A test for normality of observations and regression residuals. *International Statistical Review* 2: 163–172.
- Royston, P. 1991a. [sg3.1: Tests for departure from normality](#). *Stata Technical Bulletin* 2: 16–17. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 101–104. College Station, TX: Stata Press.
- . 1991b. [sg3.2: Shapiro–Wilk and Shapiro–Francia tests](#). *Stata Technical Bulletin* 3: 19. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, p. 105. College Station, TX: Stata Press.



- . 1991c. [sg3.5: Comment on sg3.4 and an improved D’Agostino test](#). *Stata Technical Bulletin* 3: 23–24. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 110–112. College Station, TX: Stata Press.
- . 1991d. [sg3.6: A response to sg3.3: Comment on tests of normality](#). *Stata Technical Bulletin* 4: 8–9. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 112–114. College Station, TX: Stata Press.

## Also see

- [\[R\] diagnostic plots](#) — Distributional diagnostic plots
- [\[R\] ladder](#) — Ladder of powers
- [\[R\] lv](#) — Letter-value displays
- [\[R\] swilk](#) — Shapiro–Wilk and Shapiro–Francia tests for normality

Syntax

```
slogit depvar [indepvars] [if] [in] [weight] [, options]
```

| <i>options</i>                          | description                                                                                                                                                                 |
|-----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                   |                                                                                                                                                                             |
| <code>dimension(#)</code>               | dimension of the model; default is <code>dimension(1)</code>                                                                                                                |
| <code>baseoutcome(# lbl)</code>         | set the base outcome to level <code>#</code> or <code>lbl</code> ; default is the last outcome                                                                              |
| <code>constraints(numlist)</code>       | apply specified linear constraints                                                                                                                                          |
| <code>collinear</code>                  | keep collinear variables                                                                                                                                                    |
| <code>nocorner</code>                   | do not generate the corner constraints                                                                                                                                      |
| SE/Robust                               |                                                                                                                                                                             |
| <code>vce(<i>vcetype</i>)</code>        | <i>vcetype</i> may be <code>oim</code> , <code>robust</code> , <code>cluster <i>clustvar</i></code> , <code>opg</code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                               |                                                                                                                                                                             |
| <code>level(#)</code>                   | set confidence level; default is <code>level(95)</code>                                                                                                                     |
| <code>nocnsreport</code>                | do not display constraints                                                                                                                                                  |
| <code>display_options</code>            | control spacing and display of omitted variables and base and empty cells                                                                                                   |
| Maximization                            |                                                                                                                                                                             |
| <code>maximize_options</code>           | control the maximization process; seldom used                                                                                                                               |
| <code>initialize(<i>inotype</i>)</code> | method of initializing scale parameters; <i>inotype</i> can be <code>constant</code> , <code>random</code> , or <code>svd</code> ; see <a href="#">Options</a> for details  |
| <code>nonormalize</code>                | do not normalize the numeric variables                                                                                                                                      |
| <sup>†</sup> <code>coeflegend</code>    | display coefficients' legend instead of coefficient table                                                                                                                   |

<sup>†</sup>`coeflegend` does not appear in the dialog box.

*indepvars* may contain factor variables; see [\[U\] 11.4.3 Factor variables](#).

`bootstrap`, `by`, `jackknife`, `rolling`, `statsby`, and `svy` are allowed; see [\[U\] 11.1.10 Prefix commands](#).

Weights are not allowed with the `bootstrap` prefix.

`vce()` and weights are not allowed with the `svy` prefix.

`fweights`, `iweights`, and `pweights` are allowed; see [\[U\] 11.1.6 weight](#).

See [\[U\] 20 Estimation and postestimation commands](#) for more capabilities of estimation commands.

Menu

Statistics > Categorical outcomes > Stereotype logistic regression

## Description

`slogit` fits maximum-likelihood stereotype logistic regression models as developed by [Anderson \(1984\)](#). Like multinomial logistic and ordered logistic models, stereotype logistic models are for use with categorical dependent variables. In a multinomial logistic model, the categories cannot be ranked, whereas in an ordered logistic model the categories follow a natural ranking scheme. You can view stereotype logistic models as a compromise between those two models. You can use them when you are unsure of the relevance of the ordering, as is often the case when subjects are asked to assess or judge something. You can also use them in place of multinomial logistic models when you suspect that some of the alternatives are similar. Unlike ordered logistic models, stereotype logistic models do not impose the proportional-odds assumption.

## Options

### Model

`dimension(#)` specifies the dimension of the model, which is the number of equations required to describe the relationship between the dependent variable and the independent variables. The maximum dimension is  $\min(m - 1, p)$ , where  $m$  is the number of categories of the dependent variable and  $p$  is the number of independent variables in the model. The stereotype model with maximum dimension is a reparameterization of the multinomial logistic model.

`baseoutcome(# | lbl)` specifies the outcome level whose scale parameters and intercept are constrained to be zero. The base outcome may be specified as a number of a label. By default, `slogit` assumes that the outcome levels are ordered and uses the largest level of the dependent variable as the base outcome.

`constraints(numlist)`, `collinear`; see [\[R\] estimation options](#).

By default, the linear equality constraints suggested by [Anderson \(1984\)](#), termed the corner constraints, are generated for you. You can add constraints to these as needed, or you can turn off the corner constraints by specifying `nocorner`. These constraints are in addition to the constraints placed on the  $\phi$  parameters corresponding to `baseoutcome(#)`.

`nocorner` specifies that `slogit` not generate the corner constraints. If you specify `nocorner`, you must specify at least `dimension() × dimension()` constraints for the model to be identified.

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [\[R\] vce\\_option](#).

If specifying `vce(bootstrap)` or `vce(jackknife)`, you must also specify `baseoutcome()`.

### Reporting

`level(#)`; see [\[R\] estimation options](#).

`nocnsreport`; see [\[R\] estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [\[R\] estimation options](#).

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [R] [maximize](#). These options are seldom used.

Setting the optimization type to `technique(bhhh)` resets the default `vcetype` to `vce(opg)`.

`initialize(constant | random | svd)` specifies how initial estimates are computed. The default, `initialize(constant)`, is to set the scale parameters to the constant  $\min(1/2, 1/d)$ , where  $d$  is the dimension specified in `dimension()`.

`initialize(random)` requests that uniformly distributed random numbers between 0 and 1 be used as initial values for the scale parameters. If you specify this option, you should also use `set seed` to ensure that you can replicate your results (see [D] [generate](#)).

`initialize(svd)` requests that a singular value decomposition (SVD) be performed on the matrix of regression estimates from `mlogit` to reduce its rank to the dimension specified in `dimension()`. `slogit` uses the reduced-rank components of the SVD as initial estimates for the scale and regression coefficients. For details, see [Methods and formulas](#).

`nonnormalize` specifies that the numeric variables not be normalized. Normalization of the numeric variables improves numerical stability but consumes more memory in generating temporary double-precision variables. Variables that are of type `byte` are not normalized, and if initial estimates are specified using the `from()` option, normalization of variables is not performed. See [Methods and formulas](#) for more information.

The following option is available with `slogit` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

Remarks are presented under the following headings:

[Introduction](#)

[One-dimensional model](#)

[Higher-dimension models](#)

## Introduction

Stereotype logistic models are often used when subjects are requested to assess or judge something. For example, consider a survey in which consumers may be asked to rate the quality of a product on a scale from 1 to 5, with 1 indicating poor quality and 5 indicating excellent quality. If the categories are monotonically related to an underlying latent variable, the ordered logistic model is appropriate. However, suppose that consumers assess quality not just along one dimension, but rather weigh two or three latent factors. Stereotype logistic regression allows you to specify multiple equations to capture the effects of those latent variables, which you then parameterize in terms of observable characteristics. Unlike with multinomial logit, the number of equations you specify could be less than  $m - 1$ , where  $m$  is the number of categories of the dependent variable.

Stereotype logistic models are also used when categories may be indistinguishable. Suppose that a consumer must choose among A, B, C, or D. Multinomial logistic modeling assumes that the four choices are distinct in the sense that a consumer choosing one of the goods can distinguish its characteristics from the others. If goods A and B are in fact similar, consumers may be randomly picking between the two. One alternative is to combine the two categories and fit a three-category multinomial logistic model. A more flexible alternative is to use a stereotype logistic model.

In the multinomial logistic model, you estimate  $m - 1$  parameter vectors  $\tilde{\beta}_k$ ,  $k = 1, \dots, m - 1$ , where  $m$  is the number of categories of the dependent variable. The stereotype logistic model is a restriction on the multinomial model in the sense that there are  $d$  parameter vectors, where  $d$  is between one and  $\min(m - 1, p)$ , and  $p$  is the number of regressors. The relationship between the stereotype model's coefficients  $\beta_j$ ,  $j = 1, \dots, d$ , and the multinomial model's coefficients is  $\tilde{\beta}_k = -\sum_{j=1}^d \phi_{jk} \beta_j$ . The  $\phi$ s are scale parameters to be estimated along with the  $\beta_j$ s.

Given a row vector of covariates  $\mathbf{x}$ , let  $\eta_k = \theta_k - \sum_{j=1}^d \phi_{jk} \mathbf{x} \beta_j$ . The probability of observing outcome  $k$  is

$$\Pr(Y_i = k) = \begin{cases} \frac{\exp(\eta_k)}{1 + \sum_{l=1}^{m-1} \exp(\eta_l)} & k < m \\ \frac{1}{1 + \sum_{l=1}^{m-1} \exp(\eta_l)} & k = m \end{cases}$$

This model includes a set of  $\theta$  parameters so that each equation has an unrestricted constant term. If  $d = m - 1$ , the stereotype model is just a reparameterization of the multinomial logistic model. To identify the  $\phi$ s and the  $\beta$ s, you must place at least  $d^2$  restrictions on the parameters. By default, `slogit` uses the “corner constraints”  $\phi_{jj} = 1$  and  $\phi_{jk} = 0$  for  $j \neq k$ ,  $k \leq d$ , and  $j \leq d$ .

## One-dimensional model

### ► Example 1

We have 2 years of repair rating data on the make, price, mileage rating, and gear ratio of 104 foreign and 44 domestic automobiles (with 13 missing values on repair rating). We wish to fit a stereotype logistic model to discriminate between the levels of repair rating using mileage, price, gear ratio, and origin of the manufacturer. Here is an overview of our data:

```
. use http://www.stata-press.com/data/r11/auto2yr
(Automobile Models)
. tabulate repair
```

| repair    | Freq. | Percent | Cum.   |
|-----------|-------|---------|--------|
| Poor      | 5     | 3.70    | 3.70   |
| Fair      | 19    | 14.07   | 17.78  |
| Average   | 57    | 42.22   | 60.00  |
| Good      | 38    | 28.15   | 88.15  |
| Excellent | 16    | 11.85   | 100.00 |
| Total     | 135   | 100.00  |        |

The variable `repair` can take five values, 1, ..., 5, which represent the subjective rating of the car model's repair record as *Poor*, *Fair*, *Average*, *Good*, and *Excellent*.

We wish to fit the one-dimensional stereotype logistic model

$$\eta_k = \theta_k - \phi_k (\beta_1 \text{foreign} + \beta_2 \text{mpg} + \beta_3 \text{price} + \beta_4 \text{gratio})$$

for  $k < 5$  and  $\eta_5 = 0$ . To fit this model, we type

```
. slogit repair foreign mpg price gratio
Iteration 0: log likelihood = -173.78178 (not concave)
Iteration 1: log likelihood = -164.77316
Iteration 2: log likelihood = -161.7069
Iteration 3: log likelihood = -159.76138
Iteration 4: log likelihood = -159.34327
Iteration 5: log likelihood = -159.25914
Iteration 6: log likelihood = -159.25691
Iteration 7: log likelihood = -159.25691

Stereotype logistic regression Number of obs = 135
 Wald chi2(4) = 9.33
Log likelihood = -159.25691 Prob > chi2 = 0.0535

(1) [phi1_1]_cons = 1
```

| repair  | Coef.     | Std. Err.      | z     | P> z  | [95% Conf. Interval] |           |
|---------|-----------|----------------|-------|-------|----------------------|-----------|
| foreign | 5.947382  | 2.094126       | 2.84  | 0.005 | 1.84297              | 10.05179  |
| mpg     | .1911968  | .08554         | 2.24  | 0.025 | .0235414             | .3588521  |
| price   | -.0000576 | .0001357       | -0.42 | 0.671 | -.0003236            | .0002083  |
| gratio  | -4.307571 | 1.884713       | -2.29 | 0.022 | -8.00154             | -.6136017 |
| /phi1_1 | 1         | .              | .     | .     | .                    | .         |
| /phi1_2 | 1.262268  | .3530565       | 3.58  | 0.000 | .5702904             | 1.954247  |
| /phi1_3 | 1.17593   | .3169397       | 3.71  | 0.000 | .5547394             | 1.79712   |
| /phi1_4 | .8657195  | .2411228       | 3.59  | 0.000 | .3931275             | 1.338311  |
| /phi1_5 | 0         | (base outcome) |       |       |                      |           |
| /theta1 | -6.864749 | 4.21252        | -1.63 | 0.103 | -15.12114            | 1.391639  |
| /theta2 | -7.613977 | 4.861803       | -1.57 | 0.117 | -17.14294            | 1.914981  |
| /theta3 | -5.80655  | 4.987508       | -1.16 | 0.244 | -15.58189            | 3.968786  |
| /theta4 | -3.85724  | 3.824132       | -1.01 | 0.313 | -11.3524             | 3.637922  |
| /theta5 | 0         | (base outcome) |       |       |                      |           |

(repair=Excellent is the base outcome)

The coefficient associated with the first scale parameter,  $\phi_{11}$ , is 1, and its standard error and other statistics are missing. This is the corner constraint applied to the one-dimensional model; in the header, this constraint is listed as `[phi1_1]_cons = 1`. Also, the  $\phi$  and  $\theta$  parameters that are associated with the base outcome are identified. Keep in mind, though, that there are no coefficient estimates for `[phi1_5]_cons` or `[theta5]_cons` in the `ereturn` matrix `e(b)`. The Wald statistic is for a test of the joint significance of the regression coefficients on `foreign`, `mpg`, `price`, and `gratio`.

The one-dimensional stereotype model restricts the multinomial logistic regression coefficients  $\tilde{\beta}_k$ ,  $k = 1, \dots, m - 1$  to be parallel; that is,  $\tilde{\beta}_k = -\phi_k\beta$ . As [Lunt \(2001\)](#) discusses, in the one-dimensional stereotype model, one linear combination  $\mathbf{x}_i\beta$  best discriminates the outcomes of the dependent variable, and the scale parameters  $\phi_k$  measure the distance between the outcome levels and the linear predictor. If  $\phi_1 \geq \phi_2 \geq \dots \phi_{m-1} \geq \phi_m \equiv 0$ , the model suggests that the subjective assessment of the dependent variable is indeed ordered. Here the maximum likelihood estimates of the  $\phi$ s are not monotonic, as would be assumed in an ordered logit model.

We test that  $\phi_1 = \phi_2$  by typing

```
. test [phi1_2]_cons = [phi1_1]_cons
(1) - [phi1_1]_cons + [phi1_2]_cons = 0
 chi2(1) = 0.55
 Prob > chi2 = 0.4576
```

Because the two parameters are not statistically different, we decide to add a constraint to force  $\phi_1 = \phi_2$ :

```
. constraint 1 [phi1_2]_cons = [phi1_1]_cons
. slogit repair foreign mpg price gratio, constraint(1) nolog
Stereotype logistic regression Number of obs = 135
 Wald chi2(4) = 21.28
Log likelihood = -159.65769 Prob > chi2 = 0.0003
(1) [phi1_1]_cons = 1
(2) - [phi1_1]_cons + [phi1_2]_cons = 0
```

| repair  | Coef.     | Std. Err.      | z     | P> z  | [95% Conf. Interval] |           |
|---------|-----------|----------------|-------|-------|----------------------|-----------|
| foreign | 7.166515  | 1.690177       | 4.24  | 0.000 | 3.853829             | 10.4792   |
| mpg     | .2340043  | .0807042       | 2.90  | 0.004 | .0758271             | .3921816  |
| price   | -.000041  | .0001618       | -0.25 | 0.800 | -.0003581            | .000276   |
| gratio  | -5.218107 | 1.798717       | -2.90 | 0.004 | -8.743528            | -1.692686 |
| /phi1_1 | 1         | .              | .     | .     | .                    | .         |
| /phi1_2 | 1         | .              | .     | .     | .                    | .         |
| /phi1_3 | .9751096  | .1286563       | 7.58  | 0.000 | .7229478             | 1.227271  |
| /phi1_4 | .7209343  | .1220353       | 5.91  | 0.000 | .4817494             | .9601191  |
| /phi1_5 | 0         | (base outcome) |       |       |                      |           |
| /theta1 | -8.293452 | 4.645182       | -1.79 | 0.074 | -17.39784            | .8109368  |
| /theta2 | -6.958451 | 4.629292       | -1.50 | 0.133 | -16.0317             | 2.114795  |
| /theta3 | -5.620232 | 4.953981       | -1.13 | 0.257 | -15.32986            | 4.089392  |
| /theta4 | -3.745624 | 3.809189       | -0.98 | 0.325 | -11.2115             | 3.720249  |
| /theta5 | 0         | (base outcome) |       |       |                      |           |

(repair=Excellent is the base outcome)

The  $\phi$  estimates are now monotonically decreasing and the standard errors of the  $\phi$ s are small relative to the size of the estimates, so we conclude that, with the exception of outcomes *Poor* and *Fair*, the groups are distinguishable for the one-dimensional model and that the quality assessment can be ordered.

◀

## Higher-dimension models

The stereotype logistic model is not limited to ordered categorical dependent variables; you can use it on nominal data to reduce the dimension of the regressions. Recall that a multinomial model fit to a categorical dependent variable with  $m$  levels will have  $m - 1$  sets of regression coefficients. However, a model with fewer dimensions may fit the data equally well, suggesting that some of the categories are indistinguishable.

### ► Example 2

As discussed in [R] [mlogit](#), we have data on the type of health insurance available to 616 psychologically depressed subjects in the United States (Tarlov et al. 1989; Wells et al. 1989). Patients may have either an indemnity (fee-for-service) plan or a prepaid plan, such as an HMO, or may be uninsured. Demographic variables include age, gender, race, and site.

First, we fit the saturated, two-dimensional model that is equivalent to a multinomial logistic model. We choose the base outcome to be 1 (indemnity insurance) because that is the default for [mlogit](#).

```
. use http://www.stata-press.com/data/r11/sysdsn1
(Health insurance data)

. slogit insure age male nonwhite i.site, dim(2) base(1)
Iteration 0: log likelihood = -534.36165
Iteration 1: log likelihood = -534.36165

Stereotype logistic regression
Log likelihood = -534.36165
Number of obs = 615
Wald chi2(10) = 38.17
Prob > chi2 = 0.0000

(1) [phi1_2]_cons = 1
(2) [phi1_3]_cons = 0
(3) [phi2_2]_cons = 0
(4) [phi2_3]_cons = 1
```

| insure   | Coef.     | Std. Err.      | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|----------------|-------|-------|----------------------|-----------|
| dim1     |           |                |       |       |                      |           |
| age      | .011745   | .0061946       | 1.90  | 0.058 | -.0003962            | .0238862  |
| male     | -.5616934 | .2027465       | -2.77 | 0.006 | -.9590693            | -.1643175 |
| nonwhite | -.9747768 | .2363213       | -4.12 | 0.000 | -1.437958            | -.5115955 |
| site     |           |                |       |       |                      |           |
| 2        | -.1130359 | .2101903       | -0.54 | 0.591 | -.5250013            | .2989296  |
| 3        | .5879879  | .2279351       | 2.58  | 0.010 | .1412433             | 1.034733  |
| dim2     |           |                |       |       |                      |           |
| age      | .0077961  | .0114418       | 0.68  | 0.496 | -.0146294            | .0302217  |
| male     | -.4518496 | .3674867       | -1.23 | 0.219 | -1.17211             | .268411   |
| nonwhite | -.2170589 | .4256361       | -0.51 | 0.610 | -1.05129             | .6171725  |
| site     |           |                |       |       |                      |           |
| 2        | 1.211563  | .4705127       | 2.57  | 0.010 | .2893747             | 2.133751  |
| 3        | .2078123  | .3662926       | 0.57  | 0.570 | -.510108             | .9257327  |
| /phi1_1  | 0         | (base outcome) |       |       |                      |           |
| /phi1_2  | 1         | .              | .     | .     | .                    | .         |
| /phi1_3  | (omitted) |                |       |       |                      |           |
| /phi2_1  | 0         | (base outcome) |       |       |                      |           |
| /phi2_2  | (omitted) |                |       |       |                      |           |
| /phi2_3  | 1         | .              | .     | .     | .                    | .         |
| /theta1  | 0         | (base outcome) |       |       |                      |           |
| /theta2  | .2697127  | .3284422       | 0.82  | 0.412 | -.3740222            | .9134476  |
| /theta3  | -1.286943 | .5923219       | -2.17 | 0.030 | -2.447872            | -.1260134 |

(insure=Indemnity is the base outcome)



For comparison, we also fit the model by using `mlogit`:

```
. mlogit insure age male nonwhite i.site, nolog
```

Multinomial logistic regression

Number of obs = 615

LR chi2(10) = 42.99

Prob > chi2 = 0.0000

Pseudo R2 = 0.0387

Log likelihood = -534.36165

| insure    | Coef.          | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-----------|----------------|-----------|-------|-------|----------------------|-----------|
| Indemnity | (base outcome) |           |       |       |                      |           |
| Prepaid   |                |           |       |       |                      |           |
| age       | -.011745       | .0061946  | -1.90 | 0.058 | -.0238862            | .0003962  |
| male      | .5616934       | .2027465  | 2.77  | 0.006 | .1643175             | .9590693  |
| nonwhite  | .9747768       | .2363213  | 4.12  | 0.000 | .5115955             | 1.437958  |
| site      |                |           |       |       |                      |           |
| 2         | .1130359       | .2101903  | 0.54  | 0.591 | -.2989296            | .5250013  |
| 3         | -.5879879      | .2279351  | -2.58 | 0.010 | -1.034733            | -.1412433 |
| _cons     | .2697127       | .3284422  | 0.82  | 0.412 | -.3740222            | .9134476  |
| Uninsure  |                |           |       |       |                      |           |
| age       | -.0077961      | .0114418  | -0.68 | 0.496 | -.0302217            | .0146294  |
| male      | .4518496       | .3674867  | 1.23  | 0.219 | -.268411             | 1.17211   |
| nonwhite  | .2170589       | .4256361  | 0.51  | 0.610 | -.6171725            | 1.05129   |
| site      |                |           |       |       |                      |           |
| 2         | -1.211563      | .4705127  | -2.57 | 0.010 | -2.133751            | -.2893747 |
| 3         | -.2078123      | .3662926  | -0.57 | 0.570 | -.9257327            | .510108   |
| _cons     | -1.286943      | .5923219  | -2.17 | 0.030 | -2.447872            | -.1260134 |

Apart from having opposite signs, the coefficients from the stereotype logistic model are identical to those from the multinomial logit model. Recall the [definition](#) of  $\eta_k$  given in the *Remarks*, particularly the minus sign in front of the summation. One other difference in the output is that the constant estimates labeled `/theta` in the `slogit` output are the constants labeled `_cons` in the `mlogit` output.

(Continued on next page)

Next we examine the one-dimensional model.

```
. slogit insure age male nonwhite i.site, dim(1) base(1) nolog
Stereotype logistic regression Number of obs = 615
 Wald chi2(5) = 28.20
Log likelihood = -539.75205 Prob > chi2 = 0.0000
(1) [phi1_2]_cons = 1
```

| insure   | Coef.     | Std. Err.      | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|----------------|-------|-------|----------------------|-----------|
| age      | .0108366  | .0061918       | 1.75  | 0.080 | -.0012992            | .0229723  |
| male     | -.5032537 | .2078171       | -2.42 | 0.015 | -.9105678            | -.0959396 |
| nonwhite | -.9480351 | .2340604       | -4.05 | 0.000 | -1.406785            | -.489285  |
| site     |           |                |       |       |                      |           |
| 2        | -.2444316 | .2246366       | -1.09 | 0.277 | -.6847113            | .1958481  |
| 3        | .556665   | .2243799       | 2.48  | 0.013 | .1168886             | .9964415  |
| /phi1_1  | 0         | (base outcome) |       |       |                      |           |
| /phi1_2  | 1         | .              | .     | .     | .                    | .         |
| /phi1_3  | .0383539  | .4079705       | 0.09  | 0.925 | -.7612535            | .8379613  |
| /theta1  | 0         | (base outcome) |       |       |                      |           |
| /theta2  | .187542   | .3303847       | 0.57  | 0.570 | -.4600001            | .835084   |
| /theta3  | -1.860134 | .2158898       | -8.62 | 0.000 | -2.28327             | -1.436997 |

(insure=Indemnity is the base outcome)

We have reduced a two-dimensional multinomial model to one dimension, reducing the number of estimated parameters by four and decreasing the model likelihood by  $\approx 5.4$ .

slogit does not report a model likelihood-ratio test. The test of  $d = 1$  (a one-dimensional model) versus  $d = 0$  (the null model) does not have an asymptotic  $\chi^2$  distribution because the unconstrained  $\phi$  parameters (/phi1\_3 in the previous example) cannot be identified if  $\beta = 0$ . More generally, this problem precludes testing any hierarchical model of dimension  $d$  versus  $d - 1$ . Of course, the likelihood-ratio test of a full-dimension model versus  $d = 0$  is valid because the full model is just multinomial logistic, and all the  $\phi$  parameters are fixed at 0 or 1.

◀

□ Technical note

The stereotype model is a special case of the reduced-rank vector generalized linear model discussed by Yee and Hastie (2003). If we define  $\eta_{ik} = \theta_k - \sum_{j=1}^d \phi_{jk} \mathbf{x}_i \beta_j$ , for  $k = 1, \dots, m - 1$ , we can write the expression in matrix notation as

$$\boldsymbol{\eta}_i = \boldsymbol{\theta} + \boldsymbol{\Phi} (\mathbf{x}_i \mathbf{B})'$$

where  $\boldsymbol{\Phi}$  is a  $(m - 1) \times d$  matrix containing the  $\phi_{jk}$  parameters and  $\mathbf{B}$  is a  $p \times d$  matrix with columns containing the  $\beta_j$  parameters,  $j = 1, \dots, d$ . The factorization  $\boldsymbol{\Phi} \mathbf{B}'$  is not unique because  $\boldsymbol{\Phi} \mathbf{B}' = \boldsymbol{\Phi} \mathbf{M} \mathbf{M}^{-1} \mathbf{B}'$  for any nonsingular  $d \times d$  matrix  $\mathbf{M}$ . To avoid this identifiability problem, we choose  $\mathbf{M} = \boldsymbol{\Phi}_1^{-1}$ , where

$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\Phi}_1 \\ \boldsymbol{\Phi}_2 \end{pmatrix}$$

and  $\Phi_1$  is  $d \times d$  of rank  $d$  so that

$$\Phi \mathbf{M} = \begin{pmatrix} \mathbf{I}_d \\ \Phi_2 \Phi_1^{-1} \end{pmatrix}$$

and  $\mathbf{I}_d$  is a  $d \times d$  identity matrix. Thus the corner constraints used by `slogit` are  $\phi_{jj} \equiv 1$  and  $\phi_{jk} \equiv 0$  for  $j \neq k$  and  $k, j \leq d$ .

□

## Saved results

`slogit` saves the following in `e()`:

### Scalars

|                            |                                                |
|----------------------------|------------------------------------------------|
| <code>e(N)</code>          | number of observations                         |
| <code>e(k)</code>          | number of parameters                           |
| <code>e(k_indvars)</code>  | number of independent variables                |
| <code>e(k_out)</code>      | number of outcomes                             |
| <code>e(k_eq)</code>       | number of equations in <code>e(b)</code>       |
| <code>e(k_eq_model)</code> | number of equations in model Wald test         |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints |
| <code>e(df_m)</code>       | Wald test degrees of freedom                   |
| <code>e(df_0)</code>       | null model degrees of freedom                  |
| <code>e(k_dim)</code>      | model dimension                                |
| <code>e(i_base)</code>     | base outcome index                             |
| <code>e(ll)</code>         | log likelihood                                 |
| <code>e(ll_0)</code>       | null model log likelihood                      |
| <code>e(N_clust)</code>    | number of clusters                             |
| <code>e(chi2)</code>       | $\chi^2$                                       |
| <code>e(p)</code>          | significance                                   |
| <code>e(ic)</code>         | number of iterations                           |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                      |
| <code>e(rc)</code>         | return code                                    |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                    |

*(Continued on next page)*

## Macros

|                                 |                                                                            |
|---------------------------------|----------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>slogit</code>                                                        |
| <code>e(cmdline)</code>         | command as typed                                                           |
| <code>e(depvar)</code>          | name of dependent variable                                                 |
| <code>e(indvars)</code>         | independent variables                                                      |
| <code>e(k_eq_skip)</code>       | identifies which equations should not be reported in the coefficient table |
| <code>e(wtype)</code>           | weight type                                                                |
| <code>e(wexp)</code>            | weight expression                                                          |
| <code>e(title)</code>           | title in estimation output                                                 |
| <code>e(clustvar)</code>        | name of cluster variable                                                   |
| <code>e(out#)</code>            | outcome labels, $\# = 1, \dots, e(k\_out)$                                 |
| <code>e(chi2type)</code>        | Wald; type of model $\chi^2$ test                                          |
| <code>e(labels)</code>          | outcome labels or numeric levels                                           |
| <code>e(vce)</code>             | <i>vce</i> type specified in <code>vce()</code>                            |
| <code>e(vctype)</code>          | title used to label Std. Err.                                              |
| <code>e(diparm#)</code>         | display transformed parameter $\#$                                         |
| <code>e(opt)</code>             | type of optimization                                                       |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization   |
| <code>e(ml_method)</code>       | type of ml method                                                          |
| <code>e(user)</code>            | name of likelihood-evaluator program                                       |
| <code>e(technique)</code>       | maximization technique                                                     |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular                |
| <code>e(crittype)</code>        | optimization criterion                                                     |
| <code>e(properties)</code>      | b V                                                                        |
| <code>e(predict)</code>         | program used to implement predict                                          |
| <code>e(footnote)</code>        | program used to implement the footnote display                             |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>             |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>             |

## Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(outcomes)</code>     | outcome values                               |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(gradient)</code>     | gradient vector                              |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`slogit` is implemented as an ado-file.

`slogit` obtains the maximum likelihood estimates for the stereotype logistic model by using `ml`; see [R] [ml](#). Each set of regression estimates, one set of  $\beta_j$ s for each dimension, constitutes one `ml` model equation. The  $d \times (m - 1)$   $\phi$ s and the  $(m - 1)$   $\theta$ s are `ml` ancillary parameters.

Without loss of generality, let the base outcome level be the  $m$ th level of the dependent variable. Define the row vector  $\phi_k = (\phi_{1k}, \dots, \phi_{dk})$  for  $k = 1, \dots, m - 1$ , and define the  $p \times d$  matrix  $\mathbf{B} = (\beta_1, \dots, \beta_d)$ . For observation  $i$ , the log odds of outcome level  $k$  relative to level  $m$ ,  $k = 1, \dots, m - 1$  is the index

$$\begin{aligned}\ln \left\{ \frac{\Pr(Y_i = k)}{\Pr(Y_i = m)} \right\} &= \eta_{ik} = \theta_k - \phi_k (\mathbf{x}_i \mathbf{B})' \\ &= \theta_k - \phi_k \boldsymbol{\nu}_i'\end{aligned}$$

The row vector  $\boldsymbol{\nu}_i$  can be interpreted as a latent variable reducing the  $p$ -dimensional vector of covariates to a more interpretable  $d < p$  dimension.

The probability of the  $i$ th observation having outcome level  $k$  is then

$$\Pr(Y_i = k) = p_{ik} = \begin{cases} \frac{e^{\eta_{ik}}}{1 + \sum_{j=1}^{m-1} e^{\eta_{ij}}}, & \text{if } k < m \\ \frac{1}{1 + \sum_{j=1}^{m-1} e^{\eta_{ij}}}, & \text{if } k = m \end{cases}$$

from which the log-likelihood function is computed as

$$L = \sum_{i=1}^n w_i \sum_{k=1}^m I_k(y_i) \ln(p_{ik}) \quad (1)$$

Here  $w_i$  is the weight for observation  $i$  and

$$I_k(y_i) = \begin{cases} 1, & \text{if observation } y_i \text{ has outcome } k \\ 0, & \text{otherwise} \end{cases}$$

Numeric variables are normalized for numerical stability during optimization where a new double-precision variable  $\tilde{x}_j$  is created from variable  $x_j$ ,  $j = 1, \dots, p$ , such that  $\tilde{x}_j = (x_j - \bar{x}_j)/s_j$ . This feature is turned off if you specify `nonnormalize`, or if you use the `from()` option for initial estimates. Normalization is not performed on byte variables, including the indicator variables generated by [R] `xi`. The linear equality constraints for regression parameters, if specified, must be scaled also. Assume that a constraint is applied to the regression parameter associated with variable  $j$  and dimension  $i$ ,  $\beta_{ji}$ , and the corresponding element of the constraint matrix (see [P] `makecns`) is divided by  $s_j$ .

After convergence, the parameter estimates for variable  $j$  and dimension  $i$ — $\tilde{\beta}_{ji}$ , say—are transformed back to their original scale,  $\beta_{ji} = \tilde{\beta}_{ji}/s_j$ . For the intercepts, you compute

$$\theta_k = \tilde{\theta}_k + \sum_{i=1}^d \phi_{ik} \sum_{j=1}^p \frac{\tilde{\beta}_{ji} \tilde{x}_j}{s_j}$$

Initial values are computed using estimates obtained using `mlogit` to fit a multinomial logistic model. Let the  $p \times (m-1)$  matrix  $\tilde{\mathbf{B}}$  contain the multinomial logistic regression parameters less the  $m-1$  intercepts. Each  $\phi$  is initialized with constant values  $\min(1/2, 1/d)$ , the `initialize(constant)` option (the default), or, with uniform random numbers, the `initialize(random)` option. Constraints are then applied to the starting values so that the structure of the  $(m-1) \times d$  matrix  $\Phi$  is

$$\Phi = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{m-1} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_d \\ \underline{\Phi} \end{bmatrix}$$

where  $\mathbf{I}_d$  is a  $d \times d$  identity matrix. Assume that only the corner constraints are used, but any constraints you place on the scale parameters are also applied to the initial scale estimates, so the structure of  $\Phi$  will change accordingly. The  $\phi$  parameters are invariant to the scale of the covariates, so initial estimates in  $[0, 1]$  are reasonable. The constraints guarantee that the rank of  $\Phi$  is at least  $d$ , so the initial estimates for the stereotype regression parameters are obtained from  $\mathbf{B} = \tilde{\mathbf{B}}\Phi(\Phi'\Phi)^{-1}$ .

One other approach for initial estimates is provided: `initialize(svd)`. It starts with the `mlogit` estimates and computes  $\tilde{\mathbf{B}}' = \mathbf{U}\mathbf{D}\mathbf{V}'$ , where  $\mathbf{U}_{m-1 \times p}$  and  $\mathbf{V}_{p \times p}$  are orthonormal matrices and  $\mathbf{D}_{p \times p}$  is a diagonal matrix containing the singular values of  $\tilde{\mathbf{B}}$ . The estimates for  $\Phi$  and  $\mathbf{B}$  are the first  $d$  columns of  $\mathbf{U}$  and  $\mathbf{V}\mathbf{D}$ , respectively (Yee and Hastie 2003).

The score for regression coefficients is

$$\mathbf{u}_i(\beta_j) = \frac{\partial L_{ik}}{\partial \beta_j} = \mathbf{x}_i \left( \sum_{l=1}^{m-1} \phi_{jl} p_{il} - \phi_{jk} \right)$$

the score for the scale parameters is

$$u_i(\phi_{jl}) = \frac{\partial L_{ik}}{\partial \phi_{jl}} = \begin{cases} \mathbf{x}_i \beta_j (p_{ik} - 1), & \text{if } l = k \\ \mathbf{x}_i \beta_j p_{il}, & \text{if } l \neq k \end{cases}$$

for  $l = 1, \dots, m-1$ ; and the score for the intercepts is

$$u_i(\theta_l) = \frac{\partial L_{ik}}{\partial \theta_l} = \begin{cases} 1 - p_{ik}, & \text{if } l = k \\ -p_{il}, & \text{if } l \neq k \end{cases}$$

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

`slogit` also supports estimation with survey data. For details on VCEs with survey data, see [\[SVY\] variance estimation](#).

## References

- Anderson, J. A. 1984. Regression and ordered categorical variables (with discussion). *Journal of the Royal Statistical Society, Series B* 46: 1–30.
- Lunt, M. 2001. [sg163: Stereotype ordinal regression](#). *Stata Technical Bulletin* 61: 12–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 298–307. College Station, TX: Stata Press.
- Tarlov, A. R., J. E. Ware Jr., S. Greenfield, E. C. Nelson, E. Perrin, and M. Zubkoff. 1989. The medical outcomes study. An application of methods for monitoring the results of medical care. *Journal of the American Medical Association* 262: 925–930.
- Wells, K. B., R. D. Hays, M. A. Burnam, W. H. Rogers, S. Greenfield, and J. E. Ware Jr. 1989. Detection of depressive disorder for patients receiving prepaid or fee-for-service care. Results from the Medical Outcomes Survey. *Journal of the American Medical Association* 262: 3298–3302.
- Yee, T. W., and T. J. Hastie. 2003. Reduced-rank vector generalized linear models. *Statistical Modelling* 3: 15–41.

## Also see

- [R] **slogit postestimation** — Postestimation tools for slogit
- [R] **roc** — Receiver operating characteristic (ROC) analysis
- [R] **logistic** — Logistic regression, reporting odds ratios
- [R] **mlogit** — Multinomial (polytomous) logistic regression
- [R] **ologit** — Ordered logistic regression
- [R] **oprobit** — Ordered probit regression
- [U] **20 Estimation and postestimation commands**

Description

The following postestimation commands are available for `slogit`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predicted probabilities, estimated index and its approximate standard error                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Syntax for predict

```
predict [type] { stub*|newvar|newvarlist } [if] [in] [, statistic outcome(outcome)]

predict [type] { stub*|newvarlist } [if] [in], scores
```

| statistic         | description                                    |
|-------------------|------------------------------------------------|
| Main              |                                                |
| <code>pr</code>   | probability of a positive outcome; the default |
| <code>xb</code>   | linear prediction                              |
| <code>stdp</code> | standard error of the linear prediction        |

If you do not specify `outcome()`, `pr` (with one new variable specified), `xb`, and `stdp` assume `outcome(#1)`. You specify one or *k* new variables with `pr`, where *k* is the number of outcomes. You specify one new variable with `xb` and `stdp`. These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.



## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

**pr**, the default, calculates the probability of each of the categories of the dependent variable or the probability of the level specified in `outcome(outcome)`. If you specify the `outcome(outcome)` option, you need to specify only one new variable; otherwise, you must specify a new variable for each category of the dependent variable.

**xb** calculates the index,  $\theta_k - \sum_{j=1}^d \phi_{jk} \mathbf{x}_i \beta_j$ , for outcome level  $k \neq \mathbf{e}(\mathbf{i\_base})$  and dimension  $d = \mathbf{e}(\mathbf{k\_dim})$ . It returns a vector of zeros if  $k = \mathbf{e}(\mathbf{i\_base})$ . A synonym for **xb** is **index**. This option requires the `outcome(outcome)` option.

**stdp** calculates the standard error of the index. A synonym for **stdp** is **seindex**. This option requires the `outcome(outcome)` option.

`outcome(outcome)` specifies the outcome for which the statistic is to be calculated. `equation()` is a synonym for `outcome()`: it does not matter which you use. `outcome()` or `equation()` can be specified using

#1, #2, ..., where #1 means the first category of the dependent variable, #2 means the second category, etc.;

the values of the dependent variable; or

the value labels of the dependent variable if they exist.

**scores** calculates the equation-level score variables. For models with  $d$  dimensions and  $m$  levels,  $d + (d + 1)(m - 1)$  new variables are created. Assume  $j = 1, \dots, d$  and  $k = 1, \dots, m$  in the following.

The first  $d$  new variables will contain  $\partial \ln L / \partial (\mathbf{x} \beta_j)$ .

The next  $d(m - 1)$  new variables will contain  $\partial \ln L / \partial \phi_{jk}$ .

The last  $m - 1$  new variables will contain  $\partial \ln L / \partial \theta_k$ .

## Remarks

Once you have fit a stereotype logistic model, you can obtain the predicted probabilities by using the **predict** command for both the estimation sample and other samples; see [\[U\] 20 Estimation and postestimation commands](#) and [\[R\] predict](#).

**predict** without arguments (or with the **pr** option) calculates the predicted probability of each outcome of the dependent variable. You must therefore give a new variable name for each of the outcomes. To compute the estimated probability of one outcome, you use the `outcome(outcome)` option where *outcome* is the level encoding the outcome. If the dependent variable's levels are labeled, the outcomes can also be identified by the label values (see [\[D\] label](#)).

The **xb** option in conjunction with `outcome(outcome)` specifies that the index be computed for the outcome encoded by level *outcome*. Its approximate standard error is computed if the **stdp** option is specified. Only one of the **pr**, **xb**, or **stdp** options can be specified with a call to **predict**.

► Example 1

In [example 2](#) of [\[R\] slogit](#), we fit the one-dimensional stereotype model, where the *depvar* is *insure* with levels  $k = 1$  for outcome *Indemnity*,  $k = 2$  for *Prepaid*, and  $k = 3$  for *Uninsure*. The base outcome for the model is *Indemnity*, so for  $k \neq 1$  the vector of indices for the  $k$ th level is

$$\eta_k = \theta_k - \phi_k (\beta_1 \text{age} + \beta_2 \text{male} + \beta_3 \text{nonwhite} + \beta_4 2.\text{site} + \beta_5 3.\text{site})$$

We estimate the group probabilities by calling `predict` after `slogit`.

```
. use http://www.stata-press.com/data/r11/sysdsn1
(Health insurance data)
. slogit insure age male nonwhite i.site, dim(1) base(1) nolog
(output omitted)
. predict pIndemnity pPrepaid pUninsure, p
. list pIndemnity pPrepaid pUninsure insure in 1/10
```

|     | pIndem~y | pPrepaid | pUnins~e | insure    |
|-----|----------|----------|----------|-----------|
| 1.  | .5419344 | .3754875 | .0825782 | Indemnity |
| 2.  | .4359638 | .496328  | .0677081 | Prepaid   |
| 3.  | .5111583 | .4105107 | .0783309 | Indemnity |
| 4.  | .3941132 | .5442234 | .0616633 | Prepaid   |
| 5.  | .4655651 | .4625064 | .0719285 | .         |
| 6.  | .4401779 | .4915102 | .0683118 | Prepaid   |
| 7.  | .4632122 | .4651931 | .0715948 | Prepaid   |
| 8.  | .3772302 | .5635696 | .0592002 | .         |
| 9.  | .4867758 | .4383018 | .0749225 | Uninsure  |
| 10. | .5823668 | .3295802 | .0880531 | Prepaid   |

Observations 5 and 8 are not used to fit the model because *insure* is missing at these points, but `predict` estimates the probabilities for these observations since none of the independent variables is missing. You can use `if e(sample)` in the call to `predict` to use only those observations that are used to fit the model.

◄

Methods and formulas

All postestimation commands listed above are implemented as ado-files.

predict

Let level  $b$  be the base outcome that is used to fit the stereotype logistic regression model of dimension  $d$ . The index for observation  $i$  and level  $k \neq b$  is  $\eta_{ik} = \theta_k - \sum_{j=1}^d \phi_{jk} \mathbf{x}_i \beta_j$ . This is the log odds of outcome encoded as level  $k$  relative to that of  $b$  so that we define  $\eta_{ib} \equiv 0$ . The outcome probabilities for this model are defined as  $\Pr(Y_i = k) = e^{\eta_{ik}} / \sum_{j=1}^m e^{\eta_{ij}}$ . Unlike in `mlogit`, `ologit`, and `oprobit`, the index is no longer a linear function of the parameters. The standard error of index  $\eta_{ik}$  is thus computed using the delta method (see also [\[R\] predictnl](#)).

The equation-level score for regression coefficients is

$$\frac{\partial \ln L_{ik}}{\partial \mathbf{x}_i \beta_j} = \left( \sum_{l=1}^{m-1} \phi_{jl} p_{il} - \phi_{jk} \right)$$

the equation-level score for the scale parameters is

$$\frac{\partial \ln L_{ik}}{\partial \phi_{jl}} = \begin{cases} \mathbf{x}_i \beta_j (p_{ik} - 1), & \text{if } l = k \\ \mathbf{x}_i \beta_j p_{il}, & \text{if } l \neq k \end{cases}$$

for  $l = 1, \dots, m - 1$ ; and the equation-level score for the intercepts is

$$\frac{\partial \ln L_{ik}}{\partial \theta_l} = \begin{cases} 1 - p_{ik}, & \text{if } l = k \\ -p_{il}, & \text{if } l \neq k \end{cases}$$

## Also see

[R] [slogit](#) — Stereotype logistic regression

[U] [20 Estimation and postestimation commands](#)

Syntax

`smooth smoother[,twice] varname [if] [in], generate(newvar)`  
where *smoother* is specified as  $Sm[Sm[\dots]]$  and  $Sm$  is one of

$\{ 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 \} [R]$   
 $3[R]S[S|R][S|R]\dots$   
E  
H

Letters may be specified in lowercase if preferred. Examples of *smoother*[,twice] include

|       |             |       |             |               |
|-------|-------------|-------|-------------|---------------|
| 3RSSH | 3RSSH,twice | 4253H | 4253H,twice | 43RSR2H,twice |
| 3rssh | 3rssh,twice | 4253h | 4253h,twice | 43rsr2h,twice |

Menu

Statistics > Nonparametric analysis > Robust nonlinear smoother

Description

`smooth` applies the specified resistant, nonlinear smoother to *varname* and stores the smoothed series in *newvar*.

Option

`generate(newvar)` is required; it specifies the name of the new variable that will contain the smoothed values.

Remarks

Smoothing is an exploratory data-analysis technique for making the general shape of a series apparent. In this approach (Tukey 1977), the observed data series is assumed to be the sum of an underlying process that evolves smoothly (the smooth) and of an unsystematic noise component (the rough); that is,

$$\text{data} = \text{smooth} + \text{rough}$$

Smoothed values  $z_t$  are obtained by taking medians (or some other location estimate) of each point in the original data  $y_t$  and a few of the points around it. The number of points used is called the span of the smoother. Thus a span-3 smoother produces  $z_t$  by taking the median of  $y_{t-1}$ ,  $y_t$ , and  $y_{t+1}$ . `smooth` provides running median smoothers of spans 1 to 9—indicated by the digit that specifies their span. Median smoothers are resistant to isolated outliers, so they provide robustness to spikes in the data. Because the median is also a nonlinear operator, such smoothers are known as robust (or resistant) nonlinear smoothers.

`smooth` also provides the Hanning linear, nonrobust smoother, indicated by the letter H. Hanning is a span-3 smoother with binomial weights. Repeated applications of H—HH, HHH, etc.— provide binomial smoothers of span 5, 7, etc. See Cox (1997, 2004) for a graphical application of this fact.

Because one smoother usually cannot adequately separate the smooth from the rough, compound smoothers—multiple smoothers applied in sequence—are used. The smoother 35H, for instance, then smooths the data with a span-3 median smoother, smooths the result with a span-5 median smoother, and finally smooths that result with the Hanning smoother. `smooth` allows you to specify any number of smoothers in any sequence.

Three refinements can be combined with the running median and Hanning smoothers. First, the endpoints of a smooth can be given special treatment. This is specified by the E operator. Second, smoothing by 3, the span-3 running median, tends to produce flat-topped hills and valleys. The splitting operator, S, “splits” these repeated values, applies the endpoint operator to them, and then “rejoins” the series. Finally, it is sometimes useful to repeat an odd-span median smoother or the splitting operator until the smooth no longer changes. Following a digit or an S with an R specifies this type of repetition.

Even the best smoother may fail to separate the smooth from the rough adequately. To guard against losing any systematic components of the data series, after smoothing, the smoother can be reapplied to the resulting rough, and any recovered signal can be added back to the original smooth. The `twice` operator specifies this procedure. More generally, an arbitrary smoother can be applied to the rough (using a second `smooth` command), and the recovered signal can be added back to the smooth. This more general procedure is called reroughing (Tukey 1977).

The details of each of the smoothers and operators are explained in [Methods and formulas](#) below.

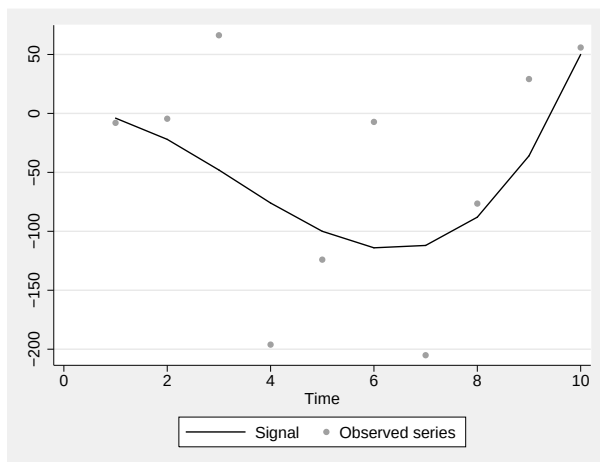
## ► Example 1

`smooth` is designed to recover the general features of a series that has been contaminated with noise. To demonstrate this, we construct a series, add noise to it, and then `smooth` the noisy version to recover an estimate of the original data. First, we construct and display the data:

```
. drop _all
. set obs 10
. set seed 123456789
. generate time = _n
. label variable time "Time"
. generate x = _n^3 - 10*_n^2 + 5*_n
. label variable x "Signal"
. generate z = x + 50*rnormal()
. label variable z "Observed series"
```

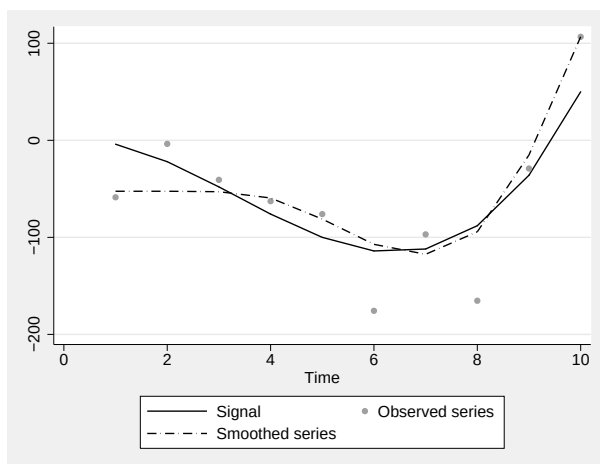
(Continued on next page)

```
. scatter x z time, c(l .) m(i o) ytitle("")
```



Now we smooth the noisy series, *z*, assumed to be the only data we would observe:

```
. smooth 4253eh,twice z, gen(sz)
. label variable sz "Smoothed series"
. scatter x z sz time, c(l . l) m(i o i) ytitle("") || scatter sz time,
> c(l . l) m(i o i) ytitle("") clpattern(dash_dot)
```



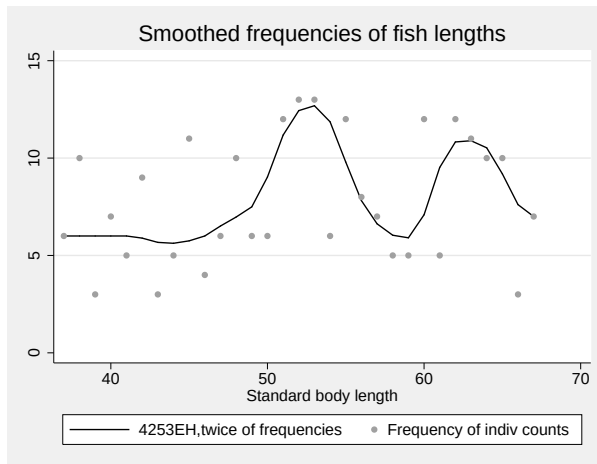
4

## Example 2

Salgado-Ugarte and Curts-García (1993) provide data on the frequencies of observed fish lengths. In this example, the series to be smoothed—the frequencies—is ordered by fish length rather than by time.

```
. use http://www.stata-press.com/data/r11/fishdata, clear
. smooth 4253eh,twice freq, gen(sfreq)
. label var sfreq "4253EH,twice of frequencies"
```

```
. scatter sfreq freq length, c(l .) m(i o)
> title("Smoothed frequencies of fish lengths") ytitle("") xlabel(#4)
```



## □ Technical note

`smooth` allows missing values at the beginning and end of the series, but missing values in the middle are not allowed. Leading and trailing missing values are ignored. If you wish to ignore missing values in the middle of the series, you must drop the missing observations before using `smooth`. Doing so, of course, would violate `smooth`'s assumption that observations are equally spaced—each observation represents a year, a quarter, or a month (or a 1-year birth-rate category). In practice, `smooth` produces good results as long as the spaces between adjacent observations do not vary too much.

Smoothing is usually applied to time series, but any variable with a natural order can be smoothed. For example, a smoother might be applied to the birth rate recorded by the age of the mothers (birth rate for 17-year-olds, birth rate for 18-year-olds, and so on).



## Methods and formulas

`smooth` is implemented as an ado-file.

Methods and formulas are presented under the following headings:

- Running median smoothers of odd span*
- Running median smoothers of even span*
- Repeat operator*
- Endpoint rule*
- Splitting operator*
- Hanning smoother*
- Twicing*

## Running median smoothers of odd span

The smoother 3 defines

$$z_t = \text{median}(y_{t-1}, y_t, y_{t+1})$$

The smoother 5 defines

$$z_t = \text{median}(y_{t-2}, y_{t-1}, y_t, y_{t+1}, y_{t+2})$$

and so on. The smoother 1 defines  $z_t = \text{median}(y_t)$ , so it does nothing.

Endpoints are handled by using smoothers of shorter, odd span. Thus for 3,

$$\begin{aligned} z_1 &= y_1 \\ z_2 &= \text{median}(y_1, y_2, y_3) \\ &\vdots \\ z_{N-1} &= \text{median}(y_{N-2}, y_{N-1}, y_N) \\ Z_N &= y_N \end{aligned}$$

For 5,

$$\begin{aligned} z_1 &= y_1 \\ z_2 &= \text{median}(y_1, y_2, y_3) \\ z_3 &= \text{median}(y_1, y_2, y_3, y_4, y_5) \\ z_4 &= \text{median}(y_2, y_3, y_4, y_5, y_6) \\ &\vdots \\ z_{N-2} &= \text{median}(y_{N-4}, y_{N-3}, y_{N-2}, y_{N-1}, y_N) \\ z_{N-1} &= \text{median}(y_{N-2}, y_{N-1}, y_N) \\ Z_N &= y_N \end{aligned}$$

and so on.

## Running median smoothers of even span

Define the  $\text{median}()$  function as returning the linearly interpolated value when given an even number of arguments. Thus the smoother 2 defines

$$z_{t+0.5} = (y_t + y_{t+1})/2$$

The smoother 4 defines  $z_{t+0.5}$  as the linearly interpolated median of  $(y_{t-1}, y_t, y_{t+1}, y_{t+2})$ , and so on. Endpoints are always handled using smoothers of shorter, even span. Thus for 4,



$$\begin{aligned}
z_{0.5} &= y_1 \\
z_{1.5} &= \text{median}(y_1, y_2) = (y_1 + y_2)/2 \\
z_{2.5} &= \text{median}(y_1, y_2, y_3, y_4) \\
&\vdots \\
z_{N-2.5} &= \text{median}(y_{N-4}, y_{N-3}, y_{N-2}, y_N) \\
z_{N-1.5} &= \text{median}(y_{N-2}, y_{N-1}) \\
z_{N-0.5} &= \text{median}(y_{N-1}, y_N) \\
z_{N+0.5} &= y_N
\end{aligned}$$

As defined above, an even-span smoother increases the length of the series by 1 observation. However, the series can be recentered on the original observation numbers, and the “extra” observation can be eliminated by smoothing the series again with another even-span smoother. For instance, the smooth of 4 illustrated above could be followed by a smooth of 2 to obtain

$$\begin{aligned}
z_1^* &= (z_{0.5} + z_{1.5})/2 \\
z_2^* &= (z_{1.5} + z_{2.5})/2 \\
z_3^* &= (z_{2.5} + z_{3.5})/2 \\
&\vdots \\
z_{N-2}^* &= (z_{N-2.5} + z_{N-1.5})/2 \\
z_{N-1}^* &= (z_{N-1.5} + z_{N-0.5})/2 \\
z_N^* &= (z_{N-0.5} + z_{N+0.5})/2
\end{aligned}$$

`smooth` keeps track of the number of even smoothers applied to the data and expands and shrinks the length of the series accordingly. To ensure that the final smooth has the same number of observations as *varname*, `smooth` requires you to specify an even number of even-span smoothers. However, the pairs of even-span smoothers need not be contiguous; for instance, 4253 and 4523 are both allowed.

## Repeat operator

R indicates that a smoother is to be repeated until convergence, that is, until repeated applications of the smoother produce the same series. Thus 3 applies the smoother of running medians of span 3. 33 applies the smoother twice. 3R produces the result of repeating 3 an infinite number of times. R should be used only with odd-span smoothers because even-span smoothers are not guaranteed to converge.

The smoother 453R2 applies a span-4 smoother, followed by a span-5 smoother, followed by repeated applications of a span-3 smoother, followed by a span-2 smoother.

## Endpoint rule

The endpoint rule E modifies the values  $z_1$  and  $z_N$  according to the following formulas:

$$z_1 = \text{median}(3z_2 - 2z_3, z_1, z_2)$$

$$z_N = \text{median}(3z_{N-2} - 2z_{N-1}, z_N, z_{N-1})$$

When the endpoint rule is not applied, endpoints are typically “copied in”; that is,  $z_1 = y_1$  and  $z_N = y_N$ .

## Splitting operator

The smoothers 3 and 3R can produce flat-topped hills and valleys. The split operator attempts to eliminate such hills and valleys by splitting the sequence, applying the endpoint rule E, rejoining the series, and then resmoothing by 3R.

The S operator may be applied only after 3, 3R, or S.

We recommend that the S operator be repeated once (SS) or until no further changes take place (SR).

## Hanning smoother

H is the Hanning linear smoother:

$$z_t = (y_{t-1} + 2y_t + y_{t+1})/4$$

Endpoints are copied in:  $z_1 = y_1$  and  $z_N = y_N$ . H should be applied only after all nonlinear smoothers.

## Twicing

A smoother divides the data into a smooth and a rough:

$$\text{data} = \text{smooth} + \text{rough}$$

If the smoothing is successful, the rough should exhibit no pattern. Twicing refers to applying the smoother to the observed, calculating the rough, and then applying the smoother to the rough. The resulting “smoothed rough” is then added back to the smooth from the first step.

## Acknowledgments

`smooth` was originally written by William Gould (1992)—at which time it was named `nlsm`—and was inspired by Salgado-Ugarte and Curts-García (1992). Salgado-Ugarte and Curts-García (1993) subsequently reported anomalies in `nlsm`’s treatment of even-span median smoothers. `smooth` corrects these problems and incorporates other improvements but otherwise is essentially the same as originally published.

## References

- Cox, N. J. 1997. [gr22: Binomial smoothing plot](#). *Stata Technical Bulletin* 35: 7–9. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 36–38. College Station, TX: Stata Press.
- . 2004. [gr22\\_1: Software update: Binomial smoothing plot](#). *Stata Journal* 4: 490.
- . 2005. [Speaking Stata: Smoothing in various directions](#). *Stata Journal* 5: 574–593.
- Gould, W. W. 1992. [sg11.1: Quantile regression with bootstrapped standard errors](#). *Stata Technical Bulletin* 9: 19–21. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 137–139. College Station, TX: Stata Press.
- Royston, P., and N. J. Cox. 2005. [A multivariable scatterplot smoother](#). *Stata Journal* 5: 405–412.
- Salgado-Ugarte, I. H., and J. Curts-García. 1992. [sed7: Resistant smoothing using Stata](#). *Stata Technical Bulletin* 7: 8–11. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 99–103. College Station, TX: Stata Press.
- . 1993. [sed7.2: Twice reroughing procedure for resistant nonlinear smoothing](#). *Stata Technical Bulletin* 11: 14–16. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 108–111. College Station, TX: Stata Press.
- Sasieni, P. 1998. [gr27: An adaptive variable span running line smoother](#). *Stata Technical Bulletin* 41: 4–7. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 63–68. College Station, TX: Stata Press.
- Tukey, J. W. 1977. *Exploratory Data Analysis*. Reading, MA: Addison–Wesley.
- Velleman, P. F. 1977. Robust nonlinear data smoothers: Definitions and recommendations. *Proceedings of the National Academy of Sciences* 74: 434–436.
- . 1980. Definition and comparison of robust nonlinear data smoothing algorithms. *Journal of the American Statistical Association* 75: 609–615.
- Velleman, P. F., and D. C. Hoaglin. 1981. *Applications, Basics, and Computing of Exploratory Data Analysis*. Boston: Duxbury.

## Also see

[R] [lowess](#) — Lowess smoothing

[TS] [tssmooth](#) — Smooth and forecast univariate time-series data

Syntax

*Spearman’s rank correlation coefficients*

```
spearman [varlist] [if] [in] [, spearman_options]
```

*Kendall’s rank correlation coefficients*

```
ktau [varlist] [if] [in] [, ktau_options]
```

| spearman_options              | description                                                                           |
|-------------------------------|---------------------------------------------------------------------------------------|
| Main                          |                                                                                       |
| stats( <i>spearman_list</i> ) | list of statistics; select up to three statistics; default is <code>stats(rho)</code> |
| <u>print</u> (#)              | significance level for displaying coefficients                                        |
| <u>star</u> (#)               | significance level for displaying with a star                                         |
| <u>bonferroni</u>             | use Bonferroni-adjusted significance level                                            |
| <u>sidak</u>                  | use Šidák-adjusted significance level                                                 |
| <u>pw</u>                     | calculate all pairwise correlation coefficients by using all available data           |
| <u>matrix</u>                 | display output in matrix form                                                         |

| ktau_options              | description                                                                          |
|---------------------------|--------------------------------------------------------------------------------------|
| Main                      |                                                                                      |
| stats( <i>ktau_list</i> ) | list of statistics; select up to six statistics; default is <code>stats(taua)</code> |
| <u>print</u> (#)          | significance level for displaying coefficients                                       |
| <u>star</u> (#)           | significance level for displaying with a star                                        |
| <u>bonferroni</u>         | use Bonferroni-adjusted significance level                                           |
| <u>sidak</u>              | use Šidák-adjusted significance level                                                |
| <u>pw</u>                 | calculate all pairwise correlation coefficients by using all available data          |
| <u>matrix</u>             | display output in matrix form                                                        |

by is allowed with `spearman` and `ktau`; see [\[D\]](#) `by`.

where the elements of *spearman\_list* may be

|     |                         |
|-----|-------------------------|
| rho | correlation coefficient |
| obs | number of observations  |
| p   | significance level      |

and the elements of *ktau\_list* may be

|       |                                  |
|-------|----------------------------------|
| taua  | correlation coefficient $\tau_a$ |
| taub  | correlation coefficient $\tau_b$ |
| score | score                            |
| se    | standard error of score          |
| obs   | number of observations           |
| p     | significance level               |

## Menu

### spearman

Statistics > Nonparametric analysis > Tests of hypotheses > Spearman's rank correlation

### ktau

Statistics > Nonparametric analysis > Tests of hypotheses > Kendall's rank correlation

## Description

**spearman** displays Spearman's rank correlation coefficients for all pairs of variables in *varlist* or, if *varlist* is not specified, for all the variables in the dataset.

**ktau** displays Kendall's rank correlation coefficients between the variables in *varlist* or, if *varlist* is not specified, for all the variables in the dataset. **ktau** is intended for use on small- and moderate-sized datasets; it requires considerable computation time for larger datasets.

## Options for spearman

### Main

**stats(spearman\_list)** specifies the statistics to be displayed in the matrix of output. **stats(rho)** is the default. Up to three statistics may be specified. **stats(rho obs p)** would display the correlation coefficient, number of observations, and significance level. If *varlist* contains only two variables, all statistics are shown in tabular form, and **stats()**, **print()**, and **star()** have no effect unless the **matrix** option is specified.

**print(#)** specifies the significance level of correlation coefficients to be printed. Correlation coefficients with larger significance levels are left blank in the matrix. Typing **spearman, print(.10)** would list only those correlation coefficients that are significant at the 10% level or lower.

**star(#)** specifies the significance level of correlation coefficients to be marked with a star. Typing **spearman, star(.05)** would “star” all correlation coefficients significant at the 5% level or lower.

**bonferroni** makes the Bonferroni adjustment to calculated significance levels. This adjustment affects printed significance levels and the **print()** and **star()** options. Thus **spearman, print(.05) bonferroni** prints coefficients with Bonferroni-adjusted significance levels of 0.05 or less.

**sidak** makes the Šidák adjustment to calculated significance levels. This adjustment affects printed significance levels and the **print()** and **star()** options. Thus **spearman, print(.05) sidak** prints coefficients with Šidák-adjusted significance levels of 0.05 or less.

**pw** specifies that correlations be calculated using pairwise deletion of observations with missing values. By default, **spearman** uses casewise deletion, where observations are ignored if any of the variables in *varlist* are missing.

**matrix** forces **spearman** to display the statistics as a matrix, even if *varlist* contains only two variables. **matrix** is implied if more than two variables are specified.

## Options for ktau

Main

`stats(ktau_list)` specifies the statistics to be displayed in the matrix of output. `stats(aua)` is the default. Up to six statistics may be specified; `stats(aua taub score se obs p)` would display the correlation coefficients  $\tau_a$ ,  $\tau_b$ , score, standard error of score, number of observations, and significance level. If *varlist* contains only two variables, all statistics are shown in tabular form and `stats()`, `print()`, and `star()` have no effect unless the `matrix` option is specified.

`print(#)` specifies the significance level of correlation coefficients to be printed. Correlation coefficients with larger significance levels are left blank in the matrix. Typing `ktau, print(.10)` would list only those correlation coefficients that are significant at the 10% level or lower.

`star(#)` specifies the significance level of correlation coefficients to be marked with a star. Typing `ktau, star(.05)` would “star” all correlation coefficients significant at the 5% level or lower.

`bonferroni` makes the Bonferroni adjustment to calculated significance levels. This adjustment affects printed significance levels and the `print()` and `star()` options. Thus `ktau, print(.05) bonferroni` prints coefficients with Bonferroni-adjusted significance levels of 0.05 or less.

`sidak` makes the Šidák adjustment to calculated significance levels. This adjustment affects printed significance levels and the `print()` and `star()` options. Thus `ktau, print(.05) sidak` prints coefficients with Šidák-adjusted significance levels of 0.05 or less.

`pw` specifies that correlations be calculated using pairwise deletion of observations with missing values. By default, `ktau` uses casewise deletion, where observations are ignored if any of the variables in *varlist* are missing.

`matrix` forces `ktau` to display the statistics as a matrix, even if *varlist* contains only two variables. `matrix` is implied if more than two variables are specified.

## Remarks

### ► Example 1

We wish to calculate the correlation coefficients among marriage rate (`mrgrate`), divorce rate (`divorce_rate`), and median age (`medage`) in state data. We can calculate the standard Pearson correlation coefficients and significance by typing

```
.use http://www.stata-press.com/data/r11/states2
(State data)
```

```
. pwcorr mrgrate divorce_rate medage, sig
```

|              | mrgrate          | divorc~e          | medage |
|--------------|------------------|-------------------|--------|
| mrgrate      | 1.0000           |                   |        |
| divorce_rate | 0.7895<br>0.0000 | 1.0000            |        |
| medage       | 0.0011<br>0.9941 | -0.1526<br>0.2900 | 1.0000 |

We can calculate Spearman's rank correlation coefficients by typing

```
. spearman mrgrate divorce_rate medage, stats(rho p)
(obs=50)
```

| Key        |  |  |  |
|------------|--|--|--|
| rho        |  |  |  |
| Sig. level |  |  |  |

|              | mrgrate | divorc~e | medage |
|--------------|---------|----------|--------|
| mrgrate      | 1.0000  |          |        |
| divorce_rate | 0.6933  | 1.0000   |        |
|              | 0.0000  |          |        |
| medage       | -0.4869 | -0.2455  | 1.0000 |
|              | 0.0003  | 0.0857   |        |

The large difference in the results is caused by one observation. Nevada's marriage rate is almost 10 times higher than the state with the next-highest marriage rate. An important feature of the Spearman rank correlation coefficient is its reduced sensitivity to extreme values compared with the Pearson correlation coefficient.

We can calculate Kendall's rank correlations by typing

```
. ktau mrgrate divorce_rate medage, stats(taua taub p)
(obs=50)
```

| Key        |  |  |  |
|------------|--|--|--|
| tau_a      |  |  |  |
| tau_b      |  |  |  |
| Sig. level |  |  |  |

|              | mrgrate | divorc~e | medage |
|--------------|---------|----------|--------|
| mrgrate      | 0.9829  |          |        |
|              | 1.0000  |          |        |
| divorce_rate | 0.5110  | 0.9804   |        |
|              | 0.5206  | 1.0000   |        |
|              | 0.0000  |          |        |
| medage       | -0.3486 | -0.1698  | 0.9845 |
|              | -0.3544 | -0.1728  | 1.0000 |
|              | 0.0004  | 0.0828   |        |

There are tied values for variables `mrgrate`, `divorce_rate`, and `medage`, so tied ranks are used. As a result,  $\tau_a < 1$  on the diagonal (see [Methods and formulas](#) for the definition of  $\tau_a$ ).

□ Technical note

According to [Conover \(1999, 323\)](#), “Spearman’s  $\rho$  tends to be larger than Kendall’s  $\tau$  in absolute value. However, as a test of significance, there is no strong reason to prefer one over the other because both will produce nearly identical results in most cases.”



▷ Example 2

We illustrate `spearman` and `ktau` with the `auto` data, which contains some missing values.

```
.use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. spearman mpg rep78

Number of obs = 69
Spearman's rho = 0.3098
Test of Ho: mpg and rep78 are independent
Prob > |t| = 0.0096
```

Because we specified two variables, `spearman` displayed the sample size, correlation, and  $p$ -value in tabular form. To obtain just the correlation coefficient displayed in matrix form, we type

```
. spearman mpg rep78, stats(rho) matrix
(obs=69)
```

|       | mpg    | rep78  |
|-------|--------|--------|
| mpg   | 1.0000 |        |
| rep78 | 0.3098 | 1.0000 |

The `pw` option instructs `spearman` and `ktau` to use all nonmissing observations between a pair of variables when calculating their correlation coefficient. In the output below, some correlations are based on 74 observations, whereas others are based on 69 because 5 observations contain a missing value for `rep78`.

```
. spearman mpg price rep78, pw stats(rho obs p) star(0.01)
```

| Key                  |
|----------------------|
| <i>rho</i>           |
| <i>Number of obs</i> |
| <i>Sig. level</i>    |

|       | mpg                      | price                  | rep78        |
|-------|--------------------------|------------------------|--------------|
| mpg   | 1.0000<br>74             |                        |              |
| price | -0.5419*<br>74<br>0.0000 | 1.0000<br>74           |              |
| rep78 | 0.3098*<br>69<br>0.0096  | 0.1028<br>69<br>0.4008 | 1.0000<br>69 |



Finally, the `bonferroni` and `sidak` options provide adjusted significance levels:

```
. ktau mpg price rep78, stats(taua taub score se p) bonferroni
(obs=69)
```

| Key                |
|--------------------|
| <i>tau_a</i>       |
| <i>tau_b</i>       |
| <i>score</i>       |
| <i>se of score</i> |
| <i>Sig. level</i>  |

|       | mpg                                                   | price                                              | rep78                                     |
|-------|-------------------------------------------------------|----------------------------------------------------|-------------------------------------------|
| mpg   | 0.9471<br>1.0000<br>2222.0000<br>191.8600             |                                                    |                                           |
| price | -0.3973<br>-0.4082<br>-932.0000<br>192.4561<br>0.0000 | 1.0000<br>1.0000<br>2346.0000<br>193.0682          |                                           |
| rep78 | 0.2076<br>0.2525<br>487.0000<br>181.7024<br>0.0224    | 0.0648<br>0.0767<br>152.0000<br>182.2233<br>1.0000 | 0.7136<br>1.0000<br>1674.0000<br>172.2161 |



Charles Edward Spearman (1863–1945) was a British psychologist who made contributions to correlation, factor analysis, test reliability, and psychometrics. After several years’ military service, he obtained a PhD in experimental psychology at Leipzig and became a professor at University College London, where he sustained a long program of work on the interpretation of intelligence tests. Ironically, the rank correlation version bearing his name is not the formula he advocated.

Maurice George Kendall (1907–1983) was a British statistician who contributed to rank correlation, time series, multivariate analysis, and other topics, and wrote many statistical texts. Most notably, perhaps, his advanced survey of the theory of statistics went through several editions, later ones with Alan Stuart; the baton has since passed to others. Kendall was employed in turn as a government and business statistician, as a professor at the London School of Economics, as a consultant, and as director of the World Fertility Survey. He was knighted in 1974.

Saved results

spearman saves the following in `r()`:

|                      |                                             |                   |                                           |
|----------------------|---------------------------------------------|-------------------|-------------------------------------------|
| Scalars              |                                             |                   |                                           |
| <code>r(N)</code>    | number of observations (last variable pair) | <code>r(p)</code> | two-sided $p$ -value (last variable pair) |
| <code>r(rho)</code>  | $\rho$ (last variable pair)                 |                   |                                           |
| Matrices             |                                             |                   |                                           |
| <code>r(Nobs)</code> | number of observations                      | <code>r(P)</code> | two-sided $p$ -value                      |
| <code>r(Rho)</code>  | $\rho$                                      |                   |                                           |

ktau saves the following in `r()`:

|                       |                                             |                          |                                              |
|-----------------------|---------------------------------------------|--------------------------|----------------------------------------------|
| Scalars               |                                             |                          |                                              |
| <code>r(N)</code>     | number of observations (last variable pair) | <code>r(score)</code>    | Kendall’s score (last variable pair)         |
| <code>r(tau_a)</code> | $\tau_a$ (last variable pair)               | <code>r(se_score)</code> | se of score (last variable pair)             |
| <code>r(tau_b)</code> | $\tau_b$ (last variable pair)               | <code>r(p)</code>        | two-sided $p$ -value<br>(last variable pair) |
| Matrices              |                                             |                          |                                              |
| <code>r(Nobs)</code>  | number of observations                      | <code>r(Score)</code>    | Kendall’s score                              |
| <code>r(Tau_a)</code> | $\tau_a$                                    | <code>r(Se_Score)</code> | standard error of score                      |
| <code>r(Tau_b)</code> | $\tau_b$                                    | <code>r(P)</code>        | two-sided $p$ -value                         |

Methods and formulas

spearman and ktau are implemented as ado-files.

Spearman’s (1904) rank correlation is calculated as Pearson’s correlation computed on the ranks and average ranks (Conover 1999, 314–315). Ranks are as calculated by egen; see [D] egen. The significance is calculated using the approximation

$$p = 2 \times \texttt{ttail}(n - 2, |\hat{\rho}| \sqrt{n - 2} / \sqrt{1 - \hat{\rho}^2})$$

For any two pairs of ranks  $(x_i, y_i)$  and  $(x_j, y_j)$  of one variable pair ( $\texttt{varname}_1, \texttt{varname}_2$ ),  $1 \leq i, j \leq n$ , where  $n$  is the number of observations, define them as concordant if

$$(x_i - x_j)(y_i - y_j) > 0$$

and discordant if this product is less than zero.

Kendall’s (1938; also see Kendall and Gibbons [1990] or Bland [2000], 222–225) score  $S$  is defined as  $C - D$ , where  $C$  ( $D$ ) is the number of concordant (discordant) pairs. Let  $N = n(n - 1)/2$  be the total number of pairs, so  $\tau_a$  is given by

$$\tau_a = S/N$$

and  $\tau_b$  is given by

$$\tau_b = \frac{S}{\sqrt{N - U} \sqrt{N - V}}$$

where

$$U = \sum_{i=1}^{N_1} u_i(u_i - 1)/2$$

$$V = \sum_{j=1}^{N_2} v_j(v_j - 1)/2$$

and where  $N_1$  is the number of sets of tied  $x$  values,  $u_i$  is the number of tied  $x$  values in the  $i$ th set,  $N_2$  is the number of sets of tied  $y$  values, and  $v_j$  is the number of tied  $y$  values in the  $j$ th set. Under the null hypothesis of independence between *varname*<sub>1</sub> and *varname*<sub>2</sub>, the variance of  $S$  is exactly (Kendall and Gibbons 1990, 66)

$$\begin{aligned} \text{Var}(S) = & \frac{1}{18} \left\{ n(n-1)(2n+5) - \sum_{i=1}^{N_1} u_i(u_i-1)(2u_i+5) - \sum_{j=1}^{N_2} v_j(v_j-1)(2v_j+5) \right\} \\ & + \frac{1}{9n(n-1)(n-2)} \left\{ \sum_{i=1}^{N_1} u_i(u_i-1)(u_i-2) \right\} \left\{ \sum_{j=1}^{N_2} v_j(v_j-1)(v_j-2) \right\} \\ & + \frac{1}{2n(n-1)} \left\{ \sum_{i=1}^{N_1} u_i(u_i-1) \right\} \left\{ \sum_{j=1}^{N_2} v_j(v_j-1) \right\} \end{aligned}$$

Using a normal approximation with a continuity correction,

$$z = \frac{|S| - 1}{\sqrt{\text{Var}(S)}}$$

For the hypothesis of independence, the statistics  $S$ ,  $\tau_a$ , and  $\tau_b$  produce equivalent tests and give the same significance.

For Kendall's  $\tau$ , the normal approximation is surprisingly accurate for sample sizes as small as 8, at least for calculating  $p$ -values under the null hypothesis for continuous variables. (See Kendall and Gibbons [1990, chap. 4], who also present some tables for calculating exact  $p$ -values for  $n < 10$ .) For Spearman's  $\rho$ , the normal approximation requires larger samples to be valid.

Let  $v$  be the number of variables specified so that  $k = v(v-1)/2$  correlation coefficients are to be estimated. If `bonferroni` is specified, the adjusted significance level is  $p' = \min(1, kp)$ . If `sidak` is specified,  $p' = \min\{1, 1 - (1-p)^n\}$ . See [Methods and formulas](#) in [R] `oneway` for a more complete description of the logic behind these adjustments.

Early work on rank correlation is surveyed by Kruskal (1958).

## Acknowledgment

The original version of `ktau` was written by Sean Beckett, a past editor of the *Stata Technical Bulletin*.

## References

- Barnard, G. A. 1997. Kendall, Maurice George. In *Leading Personalities in Statistical Sciences: From the Seventeenth Century to the Present*, ed. N. L. Johnson and S. Kotz, 130–132. New York: Wiley.
- Bland, M. 2000. *An Introduction to Medical Statistics*. 3rd ed. Oxford: Oxford University Press.
- Conover, W. J. 1999. *Practical Nonparametric Statistics*. 3rd ed. New York: Wiley.
- David, H. A., and W. A. Fuller. 2007. Sir Maurice Kendall (1907–1983): A centenary appreciation. *American Statistician* 61: 41–46.
- Jeffreys, H. 1961. *Theory of Probability*. 3rd ed. Oxford: Oxford University Press.
- Kendall, M. G. 1938. A new measure of rank correlation. *Biometrika* 30: 81–93.
- Kendall, M. G., and J. D. Gibbons. 1990. *Rank Correlation Methods*. 5th ed. New York: Oxford University Press.
- Kruskal, W. H. 1958. Ordinal measures of association. *Journal of the American Statistical Association* 53: 814–861.
- Lovie, P., and A. D. Lovie. 1996. Charles Edward Spearman, F.R.S. (1863–1945). *Notes and Records of the Royal Society of London* 50: 75–88.
- Newson, R. 2000a. [snp15: somersd—Confidence intervals for nonparametric statistics and their differences](#). *Stata Technical Bulletin* 55: 47–55. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 312–322. College Station, TX: Stata Press.
- . 2000b. [snp15.1: Update to somersd](#). *Stata Technical Bulletin* 57: 35. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 322–323. College Station, TX: Stata Press.
- . 2000c. [snp15.2: Update to somersd](#). *Stata Technical Bulletin* 58: 30. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, p. 323. College Station, TX: Stata Press.
- . 2001. [snp15.3: Update to somersd](#). *Stata Technical Bulletin* 61: 22. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, p. 324. College Station, TX: Stata Press.
- . 2003. [snp15\\_4: Software update for somersd](#). *Stata Journal* 3: 325.
- . 2005. [snp15\\_5: Software update for somersd](#). *Stata Journal* 5: 470.
- . 2006. [Confidence intervals for rank statistics: Percentile slopes, differences, and ratios](#). *Stata Journal* 6: 497–520.
- Seed, P. T. 2001. [sg159: Confidence intervals for correlations](#). *Stata Technical Bulletin* 59: 27–28. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 267–269. College Station, TX: Stata Press.
- Spearman, C. 1904. The proof and measurement of association between two things. *American Journal of Psychology* 15: 72–101.
- Wolfe, F. 1997. [sg64: pwcorr: Enhanced correlation display](#). *Stata Technical Bulletin* 35: 22–25. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 163–167. College Station, TX: Stata Press.
- . 1999. [sg64.1: Update to pwcorr](#). *Stata Technical Bulletin* 49: 17. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, p. 159. College Station, TX: Stata Press.

## Also see

[R] [correlate](#) — Correlations (covariances) of variables or coefficients

[R] [nptrend](#) — Test for trend across ordered groups

Syntax

| spikeplot <i>varname</i> [ <i>if</i> ] [ <i>in</i> ] [ <i>weight</i> ] [, <i>options</i> ] |                                                                            |
|--------------------------------------------------------------------------------------------|----------------------------------------------------------------------------|
| <i>options</i>                                                                             | description                                                                |
| Main                                                                                       |                                                                            |
| <u>round</u> (#)                                                                           | round <i>varname</i> to nearest multiple of # (bin width)                  |
| <u>fraction</u>                                                                            | make vertical scale the proportion of total values; default is frequencies |
| <u>root</u>                                                                                | make vertical scale show square roots of frequencies                       |
| Plot                                                                                       |                                                                            |
| <i>spike_options</i>                                                                       | affect rendition of plotted spikes                                         |
| Add plots                                                                                  |                                                                            |
| addplot( <i>plot</i> )                                                                     | add other plots to generated graph                                         |
| Y axis, X axis, Titles, Legend, Overall, By                                                |                                                                            |
| <i>twoway_options</i>                                                                      | any options documented in <a href="#">[G] twoway_options</a>               |
| fweights, awweights, and iweights are allowed; see <a href="#">[U] 11.1.6 weight</a> .     |                                                                            |

Menu

Graphics > Distributional graphs > Spike plot and rootogram

Description

spikeplot produces a frequency plot for a variable in which the frequencies are depicted as vertical lines from zero. The frequency may be a count, a fraction, or the square root of the count (Tukey’s rootogram, circa [1965](#)). The vertical lines may also originate from a baseline other than zero at the user’s option.

Options

|                                                                                                                             |
|-----------------------------------------------------------------------------------------------------------------------------|
| Main                                                                                                                        |
| round(#) rounds the values of <i>varname</i> to the nearest multiple of #. This action effectively specifies the bin width. |
| fraction specifies that the vertical scale be the proportion of total values (percentage) rather than the count.            |
| root specifies that the vertical scale show square roots. This option may not be specified if <i>fraction</i> is specified. |
| Plot                                                                                                                        |
| <i>spike_options</i> affect the rendition of the plotted spikes; see <a href="#">[G] graph twoway spike</a> .               |

Add plots

`addplot(plot)` provides a way to add other plots to the generated graph. See [G] [addplot\\_option](#).

Y axis, X axis, Titles, Legend, Overall, By

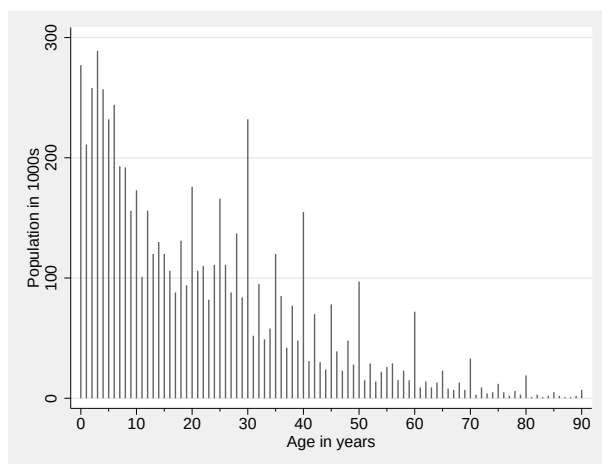
`twoway_options` are any of the options documented in [G] [twoway\\_options](#). These include options for titling the graph (see [G] [title\\_options](#)), options for saving the graph to disk (see [G] [saving\\_option](#)), and the `by()` option (see [G] [by\\_option](#)).

## Remarks

### ► Example 1

Cox and Brady (1997a) present an illustrative example using the age structure of the population of Ghana from the 1960 census (rounded to the nearest 1,000). The dataset has ages from 0 (less than 1 year) to 90. To view the distribution of ages, we would like to use each integer from 0 to 90 as the bins for the dataset.

```
. use http://www.stata-press.com/data/r11/ghanaage
. spikeplot age [fw=pop], ytitle("Population in 1000s") xlab(0(10)90)
> xmtick(5(10)85)
```



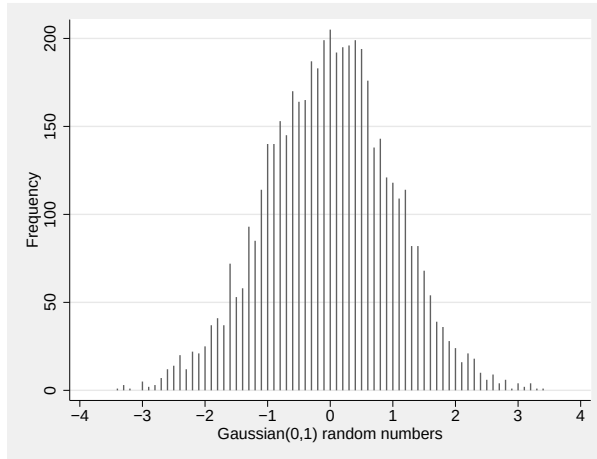
The resulting graph shows a “heaping” of ages at the multiples of 5. Also, ages ending in even numbers are more frequent than ages ending in odd numbers (except for 5). This preference for reporting ages is well known in demography and other social sciences.

Note also that we used the `ytitle()` option to override the default title of “Frequency” and that we used the `xlab()` and `xmtick()` options with *numlists* to further customize the resulting graph. See [U] [11.1.8 numlist](#) for details on specifying *numlists*.

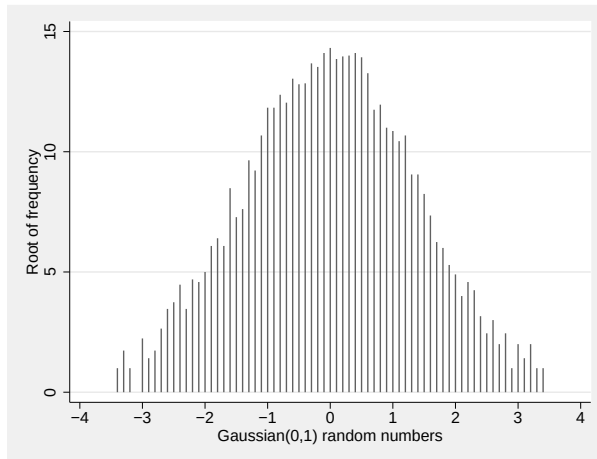
## ► Example 2

The rootogram is a plot of the square-root transformation of the frequency counts. The square root of a normal distribution is a multiple of another normal distribution.

```
. clear
. set seed 1234567
. set obs 5000
obs was 0, now 5000
. generate normal = rnormal()
. label variable normal "Gaussian(0,1) random numbers"
. spikeplot normal, round(.10) xlab(-4(1)4)
```



```
. spikeplot normal, round(.10) xlab(-4(1)4) root
```



Interpreting a histogram in terms of normality is thus similar to interpreting the rootogram for normality.

This example also shows how the `round()` option is used to bin the values for a spike plot of a continuous variable.

### ► Example 3

`spikeplot` can also be used to produce time-series plots. *varname* should be the time variable, and weights should be specified as the values for those times. To get a plot of daily rainfalls, we type

```
. spikeplot day [w=rain] if rain, ytitle("Daily rainfall in mm")
```

The `base()` option of `graph twoway spike` may be used to set a different baseline, such as when we want to show variations relative to an average or to some other measure of level. ↵

## Methods and formulas

`spikeplot` is implemented as an ado-file.

## Acknowledgments

The original version of `spikeplot` was written by Nicholas J. Cox of Durham University and Anthony R. Brady of the Imperial College School of Medicine ([1997a](#), [1997b](#)).

## References

- Cox, N. J., and A. R. Brady. 1997a. [gr25: Spike plots for histograms, rootograms, and time-series plots](#). *Stata Technical Bulletin* 36: 8–11. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 50–54. College Station, TX: Stata Press.
- . 1997b. [gr25.1: Spike plots for histograms, rootograms, and time-series plots: Update](#). *Stata Technical Bulletin* 40: 12. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, p. 58. College Station, TX: Stata Press.
- Tukey, J. W. 1965. The future of processes of data analysis. In *The Collected Works of John W. Tukey, Volume IV: Philosophy and Principles of Data Analysis: 1965–1986*, ed. L. V. Jones, 123–126. Monterey, CA: Wadsworth & Brooks/Cole.

## Also see

[\[R\] histogram](#) — Histograms for continuous and categorical variables



# Title

**ssc** — Install and uninstall packages from SSC

## Syntax

*Summary of packages most recently added or updated at SSC*

```
ssc new [, saving(filename[, replace]) type]
```

*Summary of most popular packages at SSC*

```
ssc hot [, n(#) author(name)]
```

*Describe a specified package at SSC*

```
ssc describe { pkgname | letter } [, saving(filename[, replace])]
```

*Install a specified package from SSC*

```
ssc install pkgname [, all replace]
```

*Uninstall from your computer a previously installed package from SSC*

```
ssc uninstall pkgname
```

*Type a specific file stored at SSC*

```
ssc type filename [, asis]
```

*Copy a specific file from SSC to your computer*

```
ssc copy filename [, plus personal replace public binary]
```

where *letter* in `ssc describe` is a–z or \_.

## Description

`ssc` works with packages (and files) from the Statistical Software Components (SSC) archive, which is often called the Boston College Archive and is provided by <http://repec.org>.

The SSC has become the premier Stata download site for user-written software on the web. `ssc` provides a convenient interface to the resources available there. For example, on Statalist (see <http://www.stata.com/statalist/>), users will often write

The program can be found by typing `ssc install newprogramname`.

Typing that would load everything associated with `newprogramname`, including the help files.

If you are searching for what is available, type `ssc new` and `ssc hot`, and see [\[R\] search](#). `search` searches the SSC and other places, too. `search` provides a GUI interface from which programs can be installed, including the programs at the SSC archive.

You can uninstall particular packages by using `ssc uninstall`. For the packages that you keep, see [\[R\] adoupdate](#) for an automated way of keeping those packages up to date.

## Command overview

**ssc new** summarizes the packages made available or updated recently. Output is presented in the Stata Viewer, and from there you may click to find out more about individual packages or to install them.

**ssc hot** lists the most popular packages—popular based on a moving average of the number of downloads in the past three months. By default, 10 packages are listed.

**ssc describe** *pkgname* describes, but does not install, the specified package. Use **search** to find packages; see [R] [search](#). If you know the package name but do not know the exact spelling, type **ssc describe** followed by one letter, a–z or \_ (underscore), to list all the packages starting with that letter.

**ssc install** *pkgname* installs the specified package. You do not have to describe a package before installing it. (You may also install a package by using **net install**; see [R] [net](#).)

**ssc uninstall** *pkgname* removes the previously installed package from your computer. It does not matter how the package was installed. (**ssc uninstall** is a synonym for **ado uninstall**, so either may be used to installed any package.)

**ssc type** *filename* types a specific file stored at SSC. **ssc cat** is a synonym for **ssc type**, which may appeal to those familiar with Unix.

**ssc copy** *filename* copies a specific file stored at SSC to your computer. By default, the file is copied to the current directory, but you can use options to change this. **ssc copy** is a rarely used alternative to **ssc install ...**, **all**. **ssc cp** is a synonym for **ssc copy**.

## Options for use with **ssc new**

**saving**(*filename*[, **replace**]) specifies that the “what’s new” summary be saved in *filename*. If *filename* is specified without a suffix, *filename.smcl* is assumed. If **saving()** is not specified, **saving(ssc\_results.smcl)** is assumed.

**type** specifies that the “what’s new” results be displayed in the Results window rather than in the Viewer.

## Options for use with **ssc hot**

**n**(#) specifies the number of packages to list; **n**(10) is the default. Specify **n**(.) to list all packages in order of popularity.

**author**(*name*) lists the 10 most popular packages by the specified author. If **n**(#) is also specified, the top # packages are listed.

## Option for use with **ssc describe**

**saving**(*filename*[, **replace**]) specifies that, in addition to the descriptions being displayed on your screen, it be saved in the specified file.

If *filename* is specified without an extension, *.smcl* will be assumed, and the file will be saved as a SMCL file.

If *filename* is specified with an extension, no default extension is added. If the extension is *.log*, the file will be stored as an ASCII text file.

If **replace** is specified, *filename* is replaced if it already exists.

## Options for use with `ssc install`

`all` specifies that any ancillary files associated with the package be downloaded to your current directory, in addition to the program and help files being installed. Ancillary files are files that do not end in `.ado` or `.sthlp` and typically contain datasets or examples of the use of the new command.

You can find out which files are associated with the package by typing `ssc describe pkgname` before or after installing. If you install without using the `all` option and then want the ancillary files, you can `ssc install` again.

`replace` specifies that any files being downloaded that already exist on your computer be replaced by the downloaded files. If `replace` is not specified and any files already exist, none of the files from the package is downloaded or installed.

It is better not to specify the `replace` option and wait to see if there is a problem. If there is a problem, it is usually better to uninstall the old package by using `ssc uninstall` or `ado uninstall` (which are, in fact, the same command).

## Option for use with `ssc type`

`asis` affects how files with the suffixes `.smcl` and `.sthlp` are displayed. The default is to interpret SMCL directives the file might contain. `asis` specifies that the file be displayed in raw, uninterpreted form.

## Options for use with `ssc copy`

`plus` specifies that the file be copied to the PLUS directory, the directory where user-written additions are installed. Typing `sysdir` will display the identity of the PLUS directory on your computer.

`personal` specifies that the file be copied to your PERSONAL directory as reported by `sysdir`.

If neither `plus` nor `personal` is specified, the default is to copy the file to the current directory.

`replace` specifies that, if the file already exists on your computer, the new file replace it.

`public` specifies that the new file be made readable by everyone; otherwise, the file will be created according to the default permission you have set with your operating system.

`binary` specifies that the file being copied is a binary file and that it is to be copied as is. The default is to assume that the file is a text file and change the end-of-line characters to those appropriate for your computer/operating system.

## Remarks

Users can add new features to Stata, and some users choose to make new features that they have written available to others via the web. The files that comprise a new feature are called a package, and a package usually consists of one or more `ado`-files and help files. The `net` command makes it reasonably easy to install and uninstall packages regardless of where they are on the web. One site, the SSC, has become particularly popular as a repository for additions to Stata. Command `ssc` is an easier to use version of `net` designed especially for the SSC.

Many packages are available at the SSC. Packages have names, such as `oaxaca`, `estout`, or `egenmore`. At SSC, capitalization is not significant, so `Oaxaca`, `ESTOUT`, and `EGENmore` are ways of writing the same package names.

When you type

```
. ssc install oaxaca
```

the files associated with the package are downloaded and installed on your computer. Package names usually correspond to the names of the command being added to Stata, so one would expect that installing the package `oaxaca` will add new command `oaxaca` to Stata on your computer, and expect that typing `help oaxaca` will provide the documentation. That is the situation here, but that is not always so. Before or after installing a package, type `ssc describe pkgname` to obtain the details.

➤ Example 1

`ssc new` summarizes the packages most recently made available or updated. Output is presented in the Viewer, from which you may click on a package name to find out more or install it. For example,

```
. ssc new
(contacting http://repec.org)
(output omitted)

METAN
module for fixed and random effects meta-analysis
Authors: Ross Harris Mike Bradburn Jon Deeks Roger Harbord Doug Altman
> Thomas Steichen Jonathan Sterne
Req: Stata version 9 (version 7 for metan7)
Revised: 2009-05-12
(output omitted)

PSMATCH2
module to perform full Mahalanobis and propensity score matching, common
> support graphing, and covariate imbalance testing
Authors: Edwin Leuven Barbara Sianesi
Req: Stata version 9 of 5jul2005 or later
Revised: 2009-05-02
(output omitted)
```

---

End of recent additions and updates

`ssc hot` provides a list of the most popular packages at SSC.

```
. ssc hot
```

**Top 10 packages at SSC**

| Apr2009 |        |          |                                                      |
|---------|--------|----------|------------------------------------------------------|
| Rank    | # hits | Package  | Author(s)                                            |
| 1       | 4403.5 | mat2txt  | Ben Jann, Michael Blasnik                            |
| 2       | 3237.9 | outreg2  | Roy Wada                                             |
| 3       | 2439.5 | estout   | Ben Jann                                             |
| 4       | 1298.7 | psmatch2 | Barbara Sianesi, Edwin Leuven                        |
| 5       | 1137.5 | ivreg2   | Christopher F Baum, Mark E Schaffer, Steven Stillman |
| 6       | 1102.1 | xtabond2 | David Roodman                                        |
| 7       | 1098.0 | outreg   | John Luke Gallup                                     |
| 8       | 662.0  | ranktest | Frank Kleibergen, Mark E Schaffer                    |
| 9       | 646.9  | gllamm   | Sophia Rabe-Hesketh                                  |
| 10      | 593.8  | xtivreg2 | Mark E Schaffer                                      |

(Click on package name for description)

Use the `n(#)` option to change the number of packages listed:

```
. ssc hot, n(20)
```

#### Top 20 packages at SSC

| Apr2009 |        |           |                                                                                                     |
|---------|--------|-----------|-----------------------------------------------------------------------------------------------------|
| Rank    | # hits | Package   | Author(s)                                                                                           |
| 1       | 4403.5 | mat2txt   | Ben Jann, Michael Blasnik                                                                           |
| 2       | 3237.9 | outreg2   | Roy Wada                                                                                            |
| 3       | 2439.5 | estout    | Ben Jann                                                                                            |
| 4       | 1298.7 | psmatch2  | Barbara Sianesi, Edwin Leuven                                                                       |
| 5       | 1137.5 | ivreg2    | Christopher F Baum, Mark E Schaffer, Steven Stillman                                                |
| 6       | 1102.1 | xtabond2  | David Roodman                                                                                       |
| 7       | 1098.0 | outreg    | John Luke Gallup                                                                                    |
| 8       | 662.0  | ranktest  | Frank Kleibergen, Mark E Schaffer                                                                   |
| 9       | 646.9  | gllamm    | Sophia Rabe-Hesketh                                                                                 |
| 10      | 593.8  | xtivreg2  | Mark E Schaffer                                                                                     |
| 11      | 555.0  | tabout    | Ian Watson                                                                                          |
| 12      | 447.5  | whitetst  | Nicholas J. Cox, Christopher F Baum                                                                 |
| 13      | 417.0  | hprescott | Christopher F Baum                                                                                  |
| 14      | 384.5  | xml_tab   | Michael Lokshin, Zurab Sajaia                                                                       |
| 15      | 380.9  | ice       | Patrick Royston                                                                                     |
| 16      | 371.0  | winsor    | Nicholas J. Cox                                                                                     |
| 17      | 366.5  | xtfmb     | Daniel Hoechle                                                                                      |
| 18      | 337.0  | overid    | Steven Stillman, Christopher F Baum, Mark E Schaffer, Vince Wiggins                                 |
| 19      | 333.9  | metan     | Jonathan Sterne, Ross Harris, Roger Harbord, Doug Altman, Thomas Steichen, Mike Bradburn, Jon Deeks |
| 20      | 324.0  | byvar     | Patrick Royston                                                                                     |

(Click on package name for description)

The author(*name*) option allows you to list the most popular packages by a specific person:

```
. ssc hot, author(baum)
```

#### Top 10 packages at SSC by author Baum

| Apr2009 |        |           |                                                                     |
|---------|--------|-----------|---------------------------------------------------------------------|
| Rank    | # hits | Package   | Author(s)                                                           |
| 5       | 1137.5 | ivreg2    | Christopher F Baum, Mark E Schaffer, Steven Stillman                |
| 12      | 447.5  | whitetst  | Nicholas J. Cox, Christopher F Baum                                 |
| 13      | 417.0  | hprescott | Christopher F Baum                                                  |
| 18      | 337.0  | overid    | Steven Stillman, Christopher F Baum, Mark E Schaffer, Vince Wiggins |
| 23      | 275.5  | ivendog   | Mark E Schaffer, Christopher F Baum, Steven Stillman                |
| 35      | 230.5  | levinlin  | Christopher F Baum, Fabian Bornhorst                                |
| 38      | 224.2  | outtable  | Christopher F Baum, Joao Pedro Azevedo                              |
| 41      | 209.0  | ipshin    | Christopher F Baum, Fabian Bornhorst                                |
| 42      | 207.0  | tsmktim   | Christopher F Baum, Vince Wiggins                                   |
| 55      | 168.0  | xttest2   | Christopher F Baum                                                  |

(Click on package name for description)

`ssc describe pkgname` describes, but does not install, the specified package. You must already know the name of the package. See [\[R\] search](#) for assistance in searching for packages. Sometimes you know the package name, but you do not know the exact spelling. Then you can type `ssc describe` followed by one letter, a–z or `_`, to list all the packages starting with that letter; even so, using `search` is better.

```
. ssc describe metan
```

---

```
package metan from http://fmwww.bc.edu/repec/bocode/m
```

---

**TITLE**

```
'METAN': module for fixed and random effects meta-analysis
```

**DESCRIPTION/AUTHOR(S)**

```
These routines provide facilities to conduct meta-analyses of
data from more than one study and to graph the results. Either
(output omitted)
```

**INSTALLATION FILES**

```
(type net install metan)
```

```
metan.ado
metan.hlp
../1/labbe.ado
../1/labbe.hlp
../1/labbe.dlg
metan7.ado
metan7.hlp
metan7.dlg
metannt.ado
metannt.hlp
metannt.dlg
../i/i2ci.ado
metan_examples.ado
```

**ANCILLARY FILES**

```
(type net get metan)
```

```
metan_example_data.dta
```

---

```
(type -ssc install metan- to install)
```

The default setting for the `saving()` option is for the output to be saved with the `.smcl` extension. You could also save the file with a `log` extension, and in this case, the file would be stored as an ASCII text file.

```
. ssc describe m, saving(m.index)
(output omitted)
```

```
. ssc describe metan, saving(metan.log)
(output omitted)
```

`ssc install pkgname` installs the specified package. You do not have to describe a package before installing it. There are ways of installing packages other than `ssc install`, such as `net`. It does not matter how a package is installed. For instance, a package can be installed using `net` and still be uninstalled using `ssc`.

```
. ssc install metan
checking metan consistency and verifying not already installed...
installing into C:\ado\plus\...
installation complete.
```

`ssc uninstall pkgname` removes the specified, previously installed package from your computer. You can uninstall immediately after installation or at any time in the future. (Technical note: `ssc uninstall` is a synonym for `ado uninstall`, so it can uninstall any installed package, not just packages obtained from the SSC.)

```
. ssc uninstall metan
```

```
package metan from http://fmwww.bc.edu/repec/bocode/m
```

```
'METAN': module for fixed and random effects meta-analysis
```

```
(package uninstalled)
```

`ssc type filename` types a specific file stored at the SSC. Although not shown in the syntax diagram, `ssc cat` is a synonym for `ssc type`, which may appeal to those familiar with Unix. To view only the `metan` help file for the `metan` package, you would type

```
. ssc type metan.hlp
```

---

```
help for metan, labbe
```

---

```
Fixed and random effects meta-analysis
```

---

```
(output omitted)
```

`ssc copy filename` copies a specific file stored at the SSC to your computer. By default, the file is copied to the current directory, but you can use options to change this. `ssc copy` is a rarely used alternative to `ssc install ...`, all. `ssc cp` is a synonym for `ssc copy`.

```
. ssc copy metan.ado
(file metan.ado copied to current directory)
```

◀

## Methods and formulas

`ssc` is implemented as an ado-file.

## Acknowledgments

`ssc` is based on `archutil` by Nicholas J. Cox, Durham University, and Christopher Baum, Boston College. The reworking of the original was done with their blessing and their participation.

## References

- Baum, C. F., and N. J. Cox. 1999. [ip29: Metadata for user-written contributions to the Stata programming language](#). *Stata Technical Bulletin* 52: 10–12. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 121–124. College Station, TX: Stata Press.
- Cox, N. J., and C. F. Baum. 2000. [ip29.1: Metadata for user-written contributions to the Stata programming language](#). *Stata Technical Bulletin* 54: 21–22. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 124–126. College Station, TX: Stata Press.

## Also see

- [R] [adoupdate](#) — Update user-written ado-files
- [R] [net](#) — Install and manage user-written additions from the Internet
- [R] [search](#) — Search Stata documentation
- [R] [sj](#) — Stata Journal and STB installation instructions
- [P] [sysdir](#) — Query and set system directories

Syntax

```
stem varname [if] [in] [, options]
```

| options           | description                                          |
|-------------------|------------------------------------------------------|
| Main              |                                                      |
| <u>prune</u>      | do not print stems that have no leaves               |
| <u>round</u> (#)  | round data to this value; default is round(1)        |
| <u>digits</u> (#) | digits per leaf; default is digits(1)                |
| <u>lines</u> (#)  | number of stems per interval of 10 <sup>digits</sup> |
| <u>width</u> (#)  | stem width; equal to 10 <sup>digits</sup> /width     |

by is allowed; see [\[D\]](#) `by`.

Menu

Statistics > Summaries, tables, and tests > Distributional plots and tests > Stem-and-leaf display

Description

stem displays stem-and-leaf plots.

Options

Main

- prune prevents printing any stems that have no leaves.
- round(#) rounds the data to this value and displays the plot in these units. If round() is not specified, noninteger data will be rounded automatically.
- digits(#) sets the number of digits per leaf. The default is 1.
- lines(#) sets the number of stems per every data interval of 10<sup>digits</sup>. The value of lines() must divide 10<sup>digits</sup>; that is, if digits(1) is specified, then lines() must divide 10. If digits(2) is specified, then lines() must divide 100, etc. Only one of lines() or width() may be specified. If neither is specified, an appropriate value will be set automatically.
- width(#) sets the width of a stem. lines() is equal to 10<sup>digits</sup>/width, and this option is merely an alternative way of setting lines(). The value of width() must divide 10<sup>digits</sup>. Only one of width() or lines() may be specified. If neither is specified, an appropriate value will be set automatically.

Note: If lines() or width() is not specified, digits() may be decreased in some circumstances to make a better-looking plot. If lines() or width() is set, the user-specified value of digits() will not be altered.



## Remarks

### ► Example 1

Stem-and-leaf displays are a compact way to present considerable information about a batch of data. For instance, using our automobile data (described in [U] [1.2.2 Example datasets](#)):

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. stem mpg
Stem-and-leaf plot for mpg (Mileage (mpg))

1t | 22
1f | 44444455
1s | 66667777
1. | 888888888999999999
2* | 00011111
2t | 22222333
2f | 444455555
2s | 666
2. | 8889
3* | 001
3t |
3f | 455
3s |
3. |
4* | 1
```

The stem-and-leaf display provides a way to list our data. The expression to the left of the vertical bar is called the stem; the digits to the right are called the leaves. All the stems that begin with the same digit and the corresponding leaves, written beside each other, reconstruct an observation of the data. Thus, if we look at the four stems that begin with the digit 1 and their corresponding leaves, we see that we have two cars rated at 12 mpg, 6 cars at 14, 2 at 15, and so on. The car with the highest mileage rating in our data is rated at 41 mpg.

The above plot is a five-line plot with `lines()` equal to 5 (five lines per interval of 10) and `width()` equal to 2 (two leaves per stem).

Instead, we could specify `lines(2)`:

```
. stem mpg, lines(2)
Stem-and-leaf plot for mpg (Mileage (mpg))

1* | 22444444
1. | 5566667777888888888999999999
2* | 00011111222223334444
2. | 5555566668889
3* | 0014
3. | 55
4* | 1
```

`stem mpg, width(5)` would produce the same plot as above.

The stem-and-leaf display provides a crude histogram of our data, one not so pretty as that produced by `histogram` (see [R] [histogram](#)), but one that is nonetheless informative.

## ► Example 2

The miles per gallon rating fits easily into a stem-and-leaf display because, in our data, it has two digits. However, `stem` does not require two digits.

```
. stem price, lines(1) digits(3)
Stem-and-leaf plot for price (Price)
 3*** | 291,299,667,748,798,799,829,895,955,984,995
 4*** | 010,060,082,099,172,181,187,195,296,389,424,425,453,482,499, ... (26)
 5*** | 079,104,172,189,222,379,397,705,719,788,798,799,886,899
 6*** | 165,229,295,303,342,486,850
 7*** | 140,827
 8*** | 129,814
 9*** | 690,735
10*** | 371,372
11*** | 385,497,995
12*** | 990
13*** | 466,594
14*** | 500
15*** | 906
```

The (26) at the right of the second stem shows that there were 26 leaves on this stem—too many to display on one line.

We can make a more compact stem-and-leaf plot by rounding. To display `stem` in units of 100, we could type

```
. stem price, round(100)
Stem-and-leaf plot for price (Price)
price rounded to nearest multiple of 100
plot in units of 100
 3* | 33778889
 4* | 0000111222234445555667777899
 5* | 11222447788899
 6* | 2233359
 7* | 18
 8* | 18
 9* | 77
10* | 44
11* | 45
12* | 0
13* | 056
14* | 5
15* | 9
```

`price`, in our data, has four or five digits. `stem` presented the display in terms of units of 100, so a car that cost \$3,291 was treated for display purposes as \$3,300.



## □ Technical note

Stem-and-leaf diagrams have been used in Japanese railway timetables, as shown in [Tufte \(1990, 46–47\)](#).



## Saved results

`stem` saves the following in `r()`:

Scalars

|                        |                                         |
|------------------------|-----------------------------------------|
| <code>r(width)</code>  | width of a stem                         |
| <code>r(digits)</code> | number of digits per leaf; default is 1 |

## Methods and formulas

`stem` is implemented as an ado-file.

## References

- Cox, N. J. 2007. [Speaking Stata: Turning over a new leaf](#). *Stata Journal* 7: 413–433.
- Emerson, J. D., and D. C. Hoaglin. 1983. Stem-and-leaf displays. In *Understanding Robust and Exploratory Data Analysis*, ed. D. C. Hoaglin, F. Mosteller, and J. W. Tukey, 7–32. New York: Wiley.
- Tufte, E. R. 1990. *Envisioning Information*. Cheshire, CT: Graphics Press.
- Tukey, J. W. 1972. Some graphic and semigraphic displays. In *Statistical Papers in Honor of George W. Snedecor*, ed. T. A. Bancroft and S. A. Brown, 293–316. Ames, IA: Iowa State University Press.
- . 1977. *Exploratory Data Analysis*. Reading, MA: Addison-Wesley.

## Also see

[R] [histogram](#) — Histograms for continuous and categorical variables

[R] [lv](#) — Letter-value displays

Syntax

stepwise [ , options ] : command

| options                                                                                                                                      | description                                        |
|----------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------|
| Model                                                                                                                                        |                                                    |
| *pr(#)                                                                                                                                       | significance level for removal from the model      |
| *pe(#)                                                                                                                                       | significance level for addition to the model       |
| Model2                                                                                                                                       |                                                    |
| forward                                                                                                                                      | perform forward-stepwise selection                 |
| hierarchical                                                                                                                                 | perform hierarchical selection                     |
| lockterm1                                                                                                                                    | keep the first term                                |
| lr                                                                                                                                           | perform likelihood-ratio test instead of Wald test |
| * At least one of pr(#) or pe(#) must be specified.                                                                                          |                                                    |
| by and xi are allowed; see [U] 11.1.10 Prefix commands.                                                                                      |                                                    |
| Weights are allowed if command allows them; see [U] 11.1.6 weight.                                                                           |                                                    |
| All postestimation commands behave as they would after command without the stepwise prefix; see the postestimation manual entry for command. |                                                    |
| See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.                                              |                                                    |

Menu

Statistics > Other > Stepwise estimation

Description

stepwise performs stepwise estimation. Typing

. stepwise, pr(#): command

performs backward-selection estimation for command. The stepwise selection method is determined by the following option combinations:

| options             | description                     |
|---------------------|---------------------------------|
| pr(#)               | backward selection              |
| pr(#) hierarchical  | backward hierarchical selection |
| pr(#) pe(#)         | backward stepwise               |
| pe(#)               | forward selection               |
| pe(#) hierarchical  | forward hierarchical selection  |
| pr(#) pe(#) forward | forward stepwise                |

command defines the estimation command to be executed. The following Stata commands are supported by stepwise:

|          |         |         |
|----------|---------|---------|
| clogit   | nbreg   | regress |
| cloglog  | ologit  | scobit  |
| glm      | oprobit | stcox   |
| intreg   | poisson | stcrreg |
| logistic | probit  | streg   |
| logit    | qreg    | tobit   |

**stepwise** expects *command* to have the following form:

```
command_name [depvar] term [term ...] [if] [in] [weight] [, command_options]
```

where *term* is either *varname* or (*varlist*) (a *varlist* in parentheses indicates that this group of variables is to be included or excluded together). *depvar* is not present when *command\_name* is **streg**, **stcox**, or **stcrreg**; otherwise, *depvar* is assumed to be present. For **intreg**, *depvar* is actually two dependent variable names (*depvar*<sub>1</sub> and *depvar*<sub>2</sub>).

**sw** is a synonym for **stepwise**.

## Options

### Model

**pr**(#) specifies the significance level for removal from the model; terms with  $p \geq \text{pr}()$  are eligible for removal.

**pe**(#) specifies the significance level for addition to the model; terms with  $p < \text{pe}()$  are eligible for addition.

### Model 2

**forward** specifies the forward-stepwise method and may be specified only when both **pr**() and **pe**() are also specified. Specifying both **pr**() and **pe**() without **forward** results in backward-stepwise selection. Specifying only **pr**() results in backward selection, and specifying only **pe**() results in forward selection.

**hierarchical** specifies hierarchical selection.

**lockterm1** specifies that the first term be included in the model and not be subjected to the selection criteria.

**lr** specifies that the test of term significance be the likelihood-ratio test. The default is the less computationally expensive Wald test; that is, the test is based on the estimated variance–covariance matrix of the estimators.

## Remarks

Remarks are presented under the following headings:

- [Introduction](#)
- [Search logic for a step](#)
- [Full search logic](#)
- [Examples](#)
- [Estimation sample considerations](#)
- [Messages](#)
- [Programming for stepwise](#)

## Introduction

### Typing

```
. stepwise, pr(.10): regress y1 x1 x2 d1 d2 d3 x4 x5
```

performs a backward-selection search for the regression model `y1` on `x1`, `x2`, `d1`, `d2`, `d3`, `x4`, and `x5`. In this search, each explanatory variable is said to be a term. Typing

```
. stepwise, pr(.10): regress y1 x1 x2 (d1 d2 d3) (x4 x5)
```

performs a similar backward-selection search, but the variables `d1`, `d2`, and `d3` are treated as one term, as are `x4` and `x5`. That is, `d1`, `d2`, and `d3` may or may not appear in the final model, but they appear or do not appear together.

### Example 1

Using the automobile dataset, we fit a backward-selection model of `mpg`:

```
. use http://www.stata-press.com/data/r11/auto
. gen weight2 = weight*weight
. stepwise, pr(.2): regress mpg weight weight2 displ gear turn headroom
> foreign price
```

begin with full model

p = 0.7116 >= 0.2000 removing headroom

p = 0.6138 >= 0.2000 removing displacement

p = 0.3278 >= 0.2000 removing price

|          |            |    |            |                 |        |
|----------|------------|----|------------|-----------------|--------|
| Source   | SS         | df | MS         | Number of obs = | 74     |
| Model    | 1736.31455 | 5  | 347.262911 | F( 5, 68) =     | 33.39  |
| Residual | 707.144906 | 68 | 10.3991898 | Prob > F =      | 0.0000 |
|          |            |    |            | R-squared =     | 0.7106 |
|          |            |    |            | Adj R-squared = | 0.6893 |
| Total    | 2443.45946 | 73 | 33.4720474 | Root MSE =      | 3.2248 |

| mpg        | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|------------|-----------|-----------|-------|-------|----------------------|-----------|
| weight     | -.0158002 | .0039169  | -4.03 | 0.000 | -.0236162            | -.0079842 |
| weight2    | 1.77e-06  | 6.20e-07  | 2.86  | 0.006 | 5.37e-07             | 3.01e-06  |
| foreign    | -3.615107 | 1.260844  | -2.87 | 0.006 | -6.131082            | -1.099131 |
| gear_ratio | 2.011674  | 1.468831  | 1.37  | 0.175 | -.9193321            | 4.94268   |
| turn       | -.3087038 | .1763099  | -1.75 | 0.084 | -.6605248            | .0431172  |
| _cons      | 59.02133  | 9.3903    | 6.29  | 0.000 | 40.28327             | 77.75938  |

This estimation treated each variable as its own term and thus considered each one separately. The engine displacement and gear ratio should really be considered together:

```
. stepwise, pr(.2): regress mpg weight weight2 (displ gear) turn headroom
> foreign price
```

```
begin with full model
```

```
p = 0.7116 >= 0.2000 removing headroom
```

```
p = 0.3944 >= 0.2000 removing displacement gear_ratio
```

```
p = 0.2798 >= 0.2000 removing price
```

| Source   | SS         | df | MS         | Number of obs = 74     |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | 1716.80842 | 4  | 429.202105 | F( 4, 69) = 40.76      |  |  |
| Residual | 726.651041 | 69 | 10.5311745 | Prob > F = 0.0000      |  |  |
|          |            |    |            | R-squared = 0.7026     |  |  |
|          |            |    |            | Adj R-squared = 0.6854 |  |  |
| Total    | 2443.45946 | 73 | 33.4720474 | Root MSE = 3.2452      |  |  |

| mpg     | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|---------|-----------|-----------|-------|-------|----------------------|-----------|
| weight  | -.0160341 | .0039379  | -4.07 | 0.000 | -.0238901            | -.0081782 |
| weight2 | 1.70e-06  | 6.21e-07  | 2.73  | 0.008 | 4.58e-07             | 2.94e-06  |
| foreign | -2.758668 | 1.101772  | -2.50 | 0.015 | -4.956643            | -.5606925 |
| turn    | -.2862724 | .176658   | -1.62 | 0.110 | -.6386955            | .0661508  |
| _cons   | 65.39216  | 8.208778  | 7.97  | 0.000 | 49.0161              | 81.76823  |

◀

## Search logic for a step

Before discussing the complete search logic, consider the logic for a step—the first step—in detail. The other steps follow the same logic. If you type

```
. stepwise, pr(.20): regress y1 x1 x2 (d1 d2 d3) (x4 x5)
```

the logic is

1. Fit the model  $y$  on  $x_1 x_2 d_1 d_2 d_3 x_4 x_5$ .
2. Consider dropping  $x_1$ .
3. Consider dropping  $x_2$ .
4. Consider dropping  $d_1 d_2 d_3$ .
5. Consider dropping  $x_4 x_5$ .
6. Find the term above that is least significant. If its significance level is  $\geq 0.20$ , remove that term.

If you type

```
. stepwise, pr(.20) hierarchical: regress y1 x1 x2 (d1 d2 d3) (x4 x5)
```

the logic would be different because the `hierarchical` option states that the terms are ordered. The initial logic would become

1. Fit the model  $y$  on  $x_1 x_2 d_1 d_2 d_3 x_4 x_5$ .
2. Consider dropping  $x_4 x_5$ —the last term.
3. If the significance of this last term is  $\geq 0.20$ , remove the term.

The process would then stop or continue. It would stop if  $x_4 x_5$  were not dropped, and otherwise, `stepwise` would continue to consider the significance of the next-to-last term,  $d_1 d_2 d_3$ .

Specifying `pe()` rather than `pr()` switches to forward estimation. If you type

```
. stepwise, pe(.20): regress y1 x1 x2 (d1 d2 d3) (x4 x5)
```

`stepwise` performs forward-selection search. The logic for the first step is

1. Fit a model of  $y$  on nothing (meaning a constant).
2. Consider adding  $x_1$ .
3. Consider adding  $x_2$ .
4. Consider adding  $d_1$   $d_2$   $d_3$ .
5. Consider adding  $x_4$   $x_5$ .
6. Find the term above that is most significant. If its significance level is  $< 0.20$ , add that term.

As with backward estimation, if you specify `hierarchical`,

```
. stepwise, pe(.20) hierarchical: regress y1 x1 x2 (d1 d2 d3) (x4 x5)
```

the search for the most significant term is restricted to the next term:

1. Fit a model of  $y$  on nothing (meaning a constant).
2. Consider adding  $x_1$ —the first term.
3. If the significance is  $< 0.20$ , add the term.

If  $x_1$  were added, `stepwise` would next consider  $x_2$ ; otherwise, the search process would stop.

`stepwise` can also use a stepwise selection logic that alternates between adding and removing terms. The full logic for all the possibilities is given below.



## Full search logic

| Option                                                              | Logic                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>pr()</code><br>(backward selection)                           | Fit the full model on all explanatory variables.<br>While the least-significant term is “insignificant”, remove it and reestimate.                                                                                                                                                                                                                                                                                                                                                 |
| <code>pr() hierarchical</code><br>(backward hierarchical selection) | Fit full model on all explanatory variables.<br>While the last term is “insignificant”, remove it and reestimate.                                                                                                                                                                                                                                                                                                                                                                  |
| <code>pr() pe()</code><br>(backward stepwise)                       | Fit full model on all explanatory variables.<br>If the least-significant term is “insignificant”, remove it and reestimate; otherwise, stop.<br>Do that again: if the least-significant term is “insignificant”, remove it and reestimate; otherwise, stop.<br>Repeatedly,<br>if the most-significant excluded term is “significant”, add it and reestimate;<br>if the least-significant included term is “insignificant”, remove it and reestimate;<br>until neither is possible. |
| <code>pe()</code><br>(forward selection)                            | Fit “empty” model.<br>While the most-significant excluded term is “significant”, add it and reestimate.                                                                                                                                                                                                                                                                                                                                                                            |
| <code>pe() hierarchical</code><br>(forward hierarchical selection)  | Fit “empty” model.<br>While the next term is “significant”, add it and reestimate.                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>pr() pe() forward</code><br>(forward stepwise)                | Fit “empty” model.<br>If the most-significant excluded term is “significant”, add it and reestimate; otherwise, stop.<br>Do that again: if the most-significant excluded term is “significant”, add it and reestimate; otherwise, stop.<br>Repeatedly,<br>if the least-significant included term is “insignificant”, remove it and reestimate;<br>if the most-significant excluded term is “significant”, add it and reestimate;<br>until neither is possible.                     |

## Examples

The following two statements are equivalent; both include solely single-variable terms:

```
. stepwise, pr(.2): reg price mpg weight displ
. stepwise, pr(.2): reg price (mpg) (weight) (displ)
```

The following two statements are equivalent; the last term in each is `r1, ..., r4`:

```
. stepwise, pr(.2) hierarchical: reg price mpg weight displ (r1-r4)
. stepwise, pr(.2) hierarchical: reg price (mpg) (weight) (displ) (r1-r4)
```

To group variables `weight` and `displ` into one term, type

```
. stepwise, pr(.2) hierarchical: reg price mpg (weight displ) (r1-r4)
```

`stepwise` can be used with commands other than `regress`; for instance,

```
. stepwise, pr(.2): logit outcome (sex weight) treated1 treated2
. stepwise, pr(.2): logistic outcome (sex weight) treated1 treated2
```

Either statement would fit the same model because `logistic` and `logit` both perform logistic regression; they differ only in how they report results; see [\[R\] logit](#) and [\[R\] logistic](#).

We use the `lockterm1` option to force the first term to be included in the model. To keep `treated1` and `treated2` in the model no matter what, we type

```
. stepwise, pr(.2) lockterm1: logistic outcome (treated1 treated2) ...
```

After `stepwise` estimation, we can type `stepwise` without arguments to redisplay results,

```
. stepwise
 (output from logistic appears)
```

or type the underlying estimation command:

```
. logistic
 (output from logistic appears)
```

At estimation time, we can specify options unique to the command being stepped:

```
. stepwise, pr(.2): logit outcome (sex weight) treated1 treated2, or
```

or is `logit`'s option to report odds ratios rather than coefficients; see [\[R\] logit](#).

## Estimation sample considerations

Whether you use backward or forward estimation, `stepwise` forms an estimation sample by taking observations with nonmissing values of all the variables specified (except for `depvar1` and `depvar2` for `intreg`). The estimation sample is held constant throughout the stepping. Thus if you type

```
. stepwise, pr(.2) hierarchical: regress amount sk edul sval
```

and variable `sval` is missing in half the data, that half of the data will not be used in the reported model, even if `sval` is not included in the final model.

The function `e(sample)` identifies the sample that was used. `e(sample)` contains 1 for observations used and 0 otherwise. For instance, if you type

```
. stepwise, pr(.2) pe(.10): logistic outcome x1 x2 (x3 x4) (x5 x6 x7)
```

and the final model is outcome on `x1`, `x5`, `x6`, and `x7`, you could re-create the final regression by typing

```
. logistic outcome x1 x5 x6 x7 if e(sample)
```

You could obtain summary statistics within the estimation sample of the independent variables by typing

```
. summarize x1 x5 x6 x7 if e(sample)
```

If you fit another model, `e(sample)` will automatically be redefined. Typing

```
. stepwise, lock pr(.2): logistic outcome (x1 x2) (x3 x4) (x5 x6 x7)
```

would automatically drop `e(sample)` and re-create it.

## Messages

**note: \_\_\_\_\_ dropped because of collinearity**

Each term is checked for collinearity, and variables within the term are dropped if collinearity is found. For instance, say that you type

```
. stepwise, pr(.2): regress y x1 x2 (r1-r4) (x3 x4)
```

and assume that variables `r1` through `r4` are mutually exclusive and exhaustive dummy variables—perhaps `r1`, ..., `r4` indicate in which of four regions the subject resides. One of the `r1`, ..., `r4` variables will be automatically dropped to identify the model.

This message should cause you no concern.

**Error message: between-term collinearity, variable \_\_\_\_\_**

After removing any within-term collinearity, if `stepwise` still finds collinearity between terms, it refuses to continue. For instance, assume that you type

```
. stepwise, pr(.2): regress y1 x1 x2 (d1-d8) (r1-r4)
```

Assume that `r1`, ..., `r4` identify in which of four regions the subject resides, and that `d1`, ..., `d8` identify the same sort of information, but more finely. `r1`, say, amounts to `d1` and `d2`; `r2` to `d3`, `d4`, and `d5`; `r3` to `d6` and `d7`; and `r4` to `d8`. You can estimate the `d*` variables or the `r*` variables, but not both.

It is your responsibility to specify noncollinear terms.

**note: \_\_\_\_\_ dropped because of estimability**

**note: \_\_\_\_\_ obs. dropped because of estimability**

You probably received this message in fitting a logistic or probit model. Regardless of estimation strategy, `stepwise` checks that the full model can be fit. The indicated variable had a 0 or infinite standard error.

For logistic, logit, and probit, this message is typically caused by one-way causation. Assume that you type

```
. stepwise, pr(.2): logistic outcome (x1 x2 x3) d1
```

and assume that variable `d1` is an indicator (dummy) variable. Further assume that whenever `d1 = 1`, `outcome = 1` in the data. Then the coefficient on `d1` is infinite. One (conservative) solution to this problem is to drop the `d1` variable and the `d1==1` observations. The underlying estimation commands `probit`, `logit`, and `logistic` report the details of the difficulty and solution; `stepwise` simply accumulates such problems and reports the above summary messages. Thus if you see this message, you could type

```
. logistic outcome x1 x2 x3 d1
```

to see the details. Although you should think carefully about such situations, Stata's solution of dropping the offending variables and observations is, in general, appropriate.

## Programming for stepwise

`stepwise` requires that *command\_name* follow standard Stata syntax and allow the `if` qualifier; see [U] **11 Language syntax**. Furthermore, *command\_name* must have `sw` or `swml` as a program property; see [P] **program properties**. If *command\_name* has `swml` as a property, *command\_name* must save the log-likelihood value in `e(ll)` and model degrees of freedom in `e(df_m)`.

## Saved results

`stepwise` saves whatever is saved by the underlying estimation command.

Also, `stepwise` saves `stepwise` in `e(stepwise)`.

## Methods and formulas

`stepwise` is implemented as an ado-file.

Some statisticians do not recommend stepwise procedures; see [Sribney \(1998\)](#) for a summary.

## References

- Beale, E. M. L. 1970. Note on procedures for variable selection in multiple regression. *Technometrics* 12: 909–914.
- Bendel, R. B., and A. A. Afifi. 1977. Comparison of stopping rules in forward “stepwise” regression. *Journal of the American Statistical Association* 72: 46–53.
- Berk, K. N. 1978. Comparing subset regression procedures. *Technometrics* 20: 1–6.
- Draper, N., and H. Smith. 1998. *Applied Regression Analysis*. 3rd ed. New York: Wiley.
- Efroymson, M. A. 1960. Multiple regression analysis. In *Mathematical Methods for Digital Computers*, ed. A. Ralston and H. S. Wilf, 191–203. New York: Wiley.
- Gorman, J. W., and R. J. Toman. 1966. Selection of variables for fitting equations to data. *Technometrics* 8: 27–51.
- Hocking, R. R. 1976. The analysis and selection of variables in linear regression. *Biometrics* 32: 1–49.
- Hosmer Jr., D. W., and S. Lemeshow. 2000. *Applied Logistic Regression*. 2nd ed. New York: Wiley.
- Kennedy Jr., W. J., and T. A. Bancroft. 1971. Model-building for prediction in regression based on repeated significance tests. *Annals of Mathematical Statistics* 42: 1273–1284.
- Mantel, N. 1970. Why stepdown procedures in variable selection. *Technometrics* 12: 621–625.
- . 1971. More on variable selection and an alternative approach (letter to the editor). *Technometrics* 13: 455–457.
- Sribney, W. M. 1998. FAQ: What are some of the problems with stepwise regression?  
<http://www.stata.com/support/faqs/stat/stepwise.html>.

- Wang, Z. 2000. [sg134: Model selection using the Akaike information criterion](#). *Stata Technical Bulletin* 54: 47–49.  
Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 335–337. College Station, TX: Stata Press.
- Williams, R. 2007. [Stata tip 46: Step we gaily, on we go](#). *Stata Journal* 7: 272–274.

## Also see

[R] [nestreg](#) — Nested model statistics

Syntax

```
suest namelist [, options]
```

where *namelist* is a list of one or more names under which estimation results were saved via `estimates store`. Wildcards may be used. `*` and `_all` refer to all stored results. A period (`.`) may be used to refer to the last estimation results, even if they have not (yet) been stored.

| <i>options</i>          | description                                                                       |
|-------------------------|-----------------------------------------------------------------------------------|
| <hr/>                   |                                                                                   |
| SE/Robust               |                                                                                   |
| svy                     | survey data estimation                                                            |
| vce( <i>vcetype</i> )   | <i>vcetype</i> may be <code>robust</code> or <code>cluster <i>clustvar</i></code> |
| Reporting               |                                                                                   |
| level(#)                | set confidence level; default is <code>level(95)</code>                           |
| dir                     | display a table describing the models                                             |
| eform( <i>string</i> )  | report exponentiated coefficients and label as <i>string</i>                      |
| display_options         | control spacing and display of omitted variables and base and empty cells         |
| <sup>†</sup> coeflegend | display coefficients' legend instead of coefficient table                         |
| <hr/>                   |                                                                                   |

<sup>†</sup>coeflegend does not appear in the dialog box.

Menu

Statistics > Postestimation > Tests > Seemingly unrelated estimation

Description

`suest` is a postestimation command; see [\[U\] 20 Estimation and postestimation commands](#).

`suest` combines the estimation results—parameter estimates and associated (co)variance matrices—stored under *namelist* into one parameter vector and *simultaneous* (co)variance matrix of the sandwich/robust type. This (co)variance matrix is appropriate even if the estimates were obtained on the same or on overlapping data.

Typical applications of `suest` are tests for intramodel and cross-model hypotheses using `test` or `testnl`, for example, a generalized Hausman specification test. `lincom` and `nlcom` may be used after `suest` to estimate linear combinations and nonlinear functions of coefficients. `suest` may also be used to adjust a standard VCE for clustering or survey design effects.

Different estimators are allowed, for example, a `regress` model and a `probit` model. The only requirement is that `predict` produce equation-level scores with the `score` option after an estimation command. The models may be estimated on different samples, due either to explicit `if` or `in` selection or to missing values. If weights are applied, the same weights (type and values) should be applied to all models in *namelist*. The estimators should be estimated without `vce(robust)` or `vce(cluster clustvar)` options. `suest` returns the robust VCE, allows the `vce(cluster clustvar)` option, and automatically works with results from the `svy` prefix command (only for `vce(linearized)`). See [\[SVY\] svy postestimation](#) for an example using `suest` with `svy: ologit`.

Because `suest` posts its results like a proper estimation command, its results can be stored via `estimates store`. Moreover, like other estimation commands, `suest` typed without arguments replays the results.

## Options

### SE/Robust

`svy` specifies that estimation results should be modified to reflect the survey design effects according to the `svyset` specifications, see [SVY] [svyset](#).

The `svy` option is implied when `suest` encounters survey estimation results from the `svy` prefix. Poststratification is allowed only with survey estimation results from the `svy` prefix.

`vce(vcetype)` specifies the type of standard error reported, which includes types that are robust to some kinds of misspecification and that allow for intragroup correlation; see [R] [vce\\_option](#).

The `vce()` option may not be combined with the `svy` option or estimation results from the `svy` prefix.

### Reporting

`level(#)` specifies the confidence level, as a percentage, for confidence intervals of the coefficients; see [R] [level](#).

`dir` displays a table describing the models in `namelist` just like `estimates dir namelist`.

`eform(str)` displays the coefficient table in exponentiated form: for each coefficient,  $\exp(b)$  rather than  $b$  is displayed, and standard errors and confidence intervals are transformed. Display of the intercept, if any, is suppressed. `str` is the table header that will be displayed above the transformed coefficients and must be 11 characters or fewer, for example, `eform("Odds ratio")`.

*display\_options*: [noomitted](#), [vsquish](#), [noemptycells](#), [baselevels](#), [allbaselevels](#); see [R] [estimation options](#).

The following option is available with `suest` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

Remarks are presented under the following headings:

*Using suest*

*Remarks on regress*

*Testing the assumption of the independence of irrelevant alternatives*

*Testing proportionality*

*Testing cross-model hypotheses*

## Using suest

If you plan to use `suest`, you must take precautions when fitting the original models. These restrictions are relaxed when using `svy` commands; see [SVY] [svy postestimation](#).

1. **suest** works with estimation commands that allow **predict** to generate equation-level score variables when supplied with the **score** (or **scores**) option. For example, equation-level score variables are generated after running **mlogit** by typing

```
. predict sc*, scores
```

2. Estimation should take place *without* the **vce(robust)** or **vce(cluster clustvar)** option. **suest** always computes the robust estimator of the (co)variance, and **suest** has a **vce(cluster clustvar)** option.

The within-model covariance matrices computed by **suest** are identical to those obtained by specifying a **vce(robust)** or **vce(cluster clustvar)** option during estimation. **suest**, however, also estimates the between-model covariances of parameter estimates.

3. Finally, the estimation results to be combined should be stored by **estimates store**; see [\[R\] estimates store](#).

After estimating and storing a series of estimation results, you are ready to combine the estimation results with **suest**,

```
. suest name1 [name2 ...] [, vce(cluster clustvar)]
```

and you can subsequently use postestimation commands, such as **test**, to test hypotheses. Here an important issue is how **suest** assigns names to the equations. If you specify one model *name*, the original equation names are left unchanged; otherwise, **suest** constructs new equation names. The coefficients of a single-equation model (such as **logit** and **poisson**) that was **estimate** stored under name *X* are collected under equation *X*. With a multiequation model stored under name *X*, **suest** prefixes *X\_* to an original equation name *eq*, forming equation name, *X\_eq*.

## □ Technical note

Earlier we said that standard errors from **suest** are identical to those obtained by specifying the **vce(robust)** option with each command individually. Thus if you fit a logistic model using **logit** with the **vce(robust)** option, you will get the same standard errors when you type

```
. suest .
```

directly after **logit** using the same data without the **vce(robust)** option.

This is not true for multiple estimation results when the estimation samples are not all the same. The standard errors from **suest** will be slightly smaller than those from individual model fits using the **vce(robust)** option because **suest** uses a larger number of observations to estimate the simultaneous (co)variance matrix.

□

## □ Technical note

In rare circumstances, **suest** may have to truncate equation names to 32 characters. When equation names are not unique because of truncation, **suest** numbers the equations within models, using equations named *X\_#*.

□



## Remarks on regress

`regress` does not include its ancillary parameter, the residual variance, in its coefficient vector and (co)variance matrix. Moreover, while the `score` option is allowed with `predict` after `regress`, a score variable is generated for the mean but not for the variance parameter. `suest` contains special code that assigns the equation name `mean` to the coefficients for the mean, adds the equation `lnvar` for the log variance, and computes the appropriate two score variables itself.

## Testing the assumption of the independence of irrelevant alternatives

The multinomial logit model and the closely related conditional logit model satisfy a probabilistic version of the assumption of the independence of irrelevant alternatives (IIA), implying that the ratio of the probabilities for two alternatives does not depend on what other alternatives are available. Hausman and McFadden (1984) proposed a test for this assumption that is implemented in the `hausman` command. The standard Hausman test has several limitations. First, the test statistic may be undefined because the estimated VCE does not satisfy the required asymptotic properties of the test. Second, the classic Hausman test applies only to the test of the equality of two estimators. Third, the test requires access to a fully efficient estimator; such an estimator may not be available, for example, if you are analyzing complex survey data. Using `suest` can overcome these three limitations.

### ► Example 1

In our first example, we follow the analysis of the type of health insurance reported in [R] `mprobit` and demonstrate the `hausman` command with the `suest/test` combination. We fit the full multinomial logit model for all three alternatives and two restricted multinomial models in which one alternative is excluded. After fitting each of these models, we store the results by using the `store` subcommand of `estimates`. `title()` simply documents the models.

```
. use http://www.stata-press.com/data/r11/sysdsn4
(Health insurance data)

. mprobit insure age male

Iteration 0: log likelihood = -552.21924
Iteration 1: log likelihood = -551.3306
Iteration 2: log likelihood = -551.32871
Iteration 3: log likelihood = -551.32871

Multinomial probit regression Number of obs = 615
 Wald chi2(4) = 8.97
Log likelihood = -551.32871 Prob > chi2 = 0.0618
```

| insure    | Coef.          | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-----------|----------------|-----------|-------|-------|----------------------|-----------|
| Indemnity | (base outcome) |           |       |       |                      |           |
| Prepaid   |                |           |       |       |                      |           |
| age       | -.0086267      | .0051707  | -1.67 | 0.095 | -.0187612            | .0015078  |
| male      | .4368331       | .1686449  | 2.59  | 0.010 | .1062951             | .7673711  |
| _cons     | .2262309       | .2396841  | 0.94  | 0.345 | -.2435413            | .6960031  |
| Uninsure  |                |           |       |       |                      |           |
| age       | -.0042211      | .0073813  | -0.57 | 0.567 | -.0186883            | .010246   |
| male      | .352813        | .234675   | 1.50  | 0.133 | -.1071415            | .8127675  |
| _cons     | -1.232167      | .3452424  | -3.57 | 0.000 | -1.90883             | -.5555044 |

```
. estimates store m1, title(all three insurance forms)
. quietly mprobit insure age male if insure != "Uninsure":insure
```

```
. estimates store m2, title(insure != "Uninsure":insure)
. quietly mprobit insure age male if insure != "Prepaid":insure
. estimates store m3, title(insure != "Prepaid":insure)
```

Having performed the three estimations, we inspect the results. `estimates dir` provides short descriptions of the models that were stored using `estimates store`. Typing `estimates table` lists the coefficients, displaying blanks for a coefficient not contained in a model.

```
. estimates dir

. estimates table m1 m2 m3, star stats(N ll)
```

| name | command | depvar | npar | title                             |
|------|---------|--------|------|-----------------------------------|
| m1   | mprobit | insure | 6    | <i>all three insurance forms</i>  |
| m2   | mprobit | insure | 3    | <i>insure != Uninsure :insure</i> |
| m3   | mprobit | insure | 3    | <i>insure != Prepaid :insure</i>  |

| Variable   | m1           | m2          | m3            |
|------------|--------------|-------------|---------------|
| Prepaid    |              |             |               |
| age        | -.0086267    | -.00900756  |               |
| male       | .43683313**  | .45451955** |               |
| _cons      | .22623091    | .23774451   |               |
| Uninsure   |              |             |               |
| age        | -.00422114   |             | -.00302811    |
| male       | .35281303    |             | .35281917     |
| _cons      | -1.232167*** |             | -1.5205246*** |
| Statistics |              |             |               |
| N          | 615          | 570         | 338           |
| ll         | -551.32871   | -390.48251  | -131.77197    |

legend: \* p<0.05; \*\* p<0.01; \*\*\* p<0.001

Comparing the coefficients between models does not suggest substantial differences. We can formally test that coefficients are the same for the full model `m1` and the restricted models `m2` and `m3` by using the `hausman` command. `hausman` expects the models to be specified in the order “always consistent” first and “efficient under  $H_0$ ” second.

```
. hausman m2 m1, alleq constant

----- Coefficients -----
 (b) (B) (b-B) sqrt(diag(V_b-V_B))
 m2 m1 Difference S.E.

age -.0090076 -.0086267 -.0003809 .0012156
male .4545195 .4368331 .0176864 .0442633
_cons .2377445 .2262309 .0115136 .0542036

b = consistent under Ho and Ha; obtained from mprobit
B = inconsistent under Ha, efficient under Ho; obtained from mprobit
Test: Ho: difference in coefficients not systematic
 chi2(3) = (b-B)'[(V_b-V_B)^(-1)](b-B)
 = 0.21
 Prob>chi2 = 0.9767
```

```
. hausman m3 m1, alleqs constant
```

|       | Coefficients |           | (b-B)<br>Difference | sqrt(diag(V <sub>b</sub> -V <sub>B</sub> ))<br>S.E. |
|-------|--------------|-----------|---------------------|-----------------------------------------------------|
|       | (b)<br>m3    | (B)<br>m1 |                     |                                                     |
| age   | -.0030281    | -.0042211 | .001193             | .0047968                                            |
| male  | .3528192     | .352813   | 6.14e-06            | .1545937                                            |
| _cons | -1.520525    | -1.232167 | -.2883576           | .2357049                                            |

b = consistent under Ho and Ha; obtained from mprobit

B = inconsistent under Ha, efficient under Ho; obtained from mprobit

Test: Ho: difference in coefficients not systematic

chi2(3) = (b-B)'[(V<sub>b</sub>-V<sub>B</sub>)<sup>-1</sup>](b-B)

= 13.16

Prob>chi2 = 0.0043

According to the test of m1 against m2, we cannot reject the hypothesis that the coefficients of m1 and m2 are the same. The second Hausman test is not well defined—something that happens fairly often. The problem is due to the estimator of the variance  $V(b-B)$  as  $V(b)-V(B)$ , which is a feasible estimator only asymptotically. Here it simply is not a proper variance matrix, and the Hausman test becomes undefined.

suest m1 m2 estimates the simultaneous (co)variance of the coefficients of models m1 and m2. Although suest is technically a postestimation command, it acts like an estimation command in that it stores the simultaneous coefficients in  $e(b)$  and the full (co)variance matrix in  $e(V)$ . We use the estat vce command to display this full (co)variance matrix to show that the cross-model covariances have indeed been estimated. Typically, we do not expect to have a direct interest in  $e(V)$ .

```
. suest m1 m2
```

Simultaneous results for m1, m2

Number of obs = 615

|             | Coef.     | Robust<br>Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-------------|-----------|---------------------|-------|-------|----------------------|-----------|
| m1_Prepaid  |           |                     |       |       |                      |           |
| age         | -.0086267 | .0051181            | -1.69 | 0.092 | -.0186579            | .0014045  |
| male        | .4368331  | .1692997            | 2.58  | 0.010 | .1050119             | .7686544  |
| _cons       | .2262309  | .2387678            | 0.95  | 0.343 | -.2417455            | .6942073  |
| m1_Uninsure |           |                     |       |       |                      |           |
| age         | -.0042211 | .0071001            | -0.59 | 0.552 | -.018137             | .0096947  |
| male        | .352813   | .2370142            | 1.49  | 0.137 | -.1117262            | .8173523  |
| _cons       | -1.232167 | .3267826            | -3.77 | 0.000 | -1.872649            | -.5916849 |
| m2_Prepaid  |           |                     |       |       |                      |           |
| age         | -.0090076 | .0052361            | -1.72 | 0.085 | -.0192702            | .0012551  |
| male        | .4545195  | .1753731            | 2.59  | 0.010 | .1107946             | .7982445  |
| _cons       | .2377445  | .2434992            | 0.98  | 0.329 | -.2395051            | .7149941  |

(Continued on next page)

. estat vce

Covariance matrix of coefficients of suest model

| e(V)        | m1_Prepaid<br>age   | male              | _cons      | m1_Unins~e<br>age | male       |
|-------------|---------------------|-------------------|------------|-------------------|------------|
| m1_Prepaid  |                     |                   |            |                   |            |
| age         | .00002619           |                   |            |                   |            |
| male        | -.00008615          | .02866237         |            |                   |            |
| _cons       | -.00114502          | -.00319028        | .05701008  |                   |            |
| m1_Uninsure |                     |                   |            |                   |            |
| age         | .00001225           | -.00004954        | -.00053868 | .00005041         |            |
| male        | -.00004619          | .01482174         | -.00131192 | -.0003288         | .05617571  |
| _cons       | -.00053868          | -.00118073        | .02704676  | -.00215773        | -.00021218 |
| m2_Prepaid  |                     |                   |            |                   |            |
| age         | .00002678           | -.00009876        | -.00116793 | .00001271         | -.00006696 |
| male        | -.0000977           | .02967604         | -.00288912 | -.00006734        | .0158878   |
| _cons       | -.00116801          | -.0028486         | .05810244  | -.00055873        | -.00044666 |
| e(V)        | m1_Unins~e<br>_cons | m2_Prepaid<br>age | male       | _cons             |            |
| m1_Uninsure |                     |                   |            |                   |            |
| _cons       | .10678688           |                   |            |                   |            |
| m2_Prepaid  |                     |                   |            |                   |            |
| age         | -.00055858          | .00002742         |            |                   |            |
| male        | -.00041519          | -.00011188        | .03075572  |                   |            |
| _cons       | .0279299            | -.00119326        | -.00248963 | .05929184         |            |

suest created equation names by combining the name under which we stored the results using `estimates store` with the original equation names. Thus, in the simultaneous estimation result, equation `Prepaid` originating in model `m1` is named `m1_Prepaid`. According to the McFadden–Hausman specification of a test for IIA, the coefficients of the equations `m1_PrePaid` and `m2_PrePaid` should be equal. This equality can be tested easily with the `test` command. The `cons` option specifies that the intercept `_cons` be included in the test.

```
. test [m1_Prepaid = m2_Prepaid], cons
(1) [m1_Prepaid]age - [m2_Prepaid]age = 0
(2) [m1_Prepaid]male - [m2_Prepaid]male = 0
(3) [m1_Prepaid]_cons - [m2_Prepaid]_cons = 0
 chi2(3) = 5.80
 Prob > chi2 = 0.1218
```

The Hausman test via `suest` is comparable to that computed by `hausman`, but they use different estimators of the variance of the difference of the estimates. The `hausman` command estimates  $V(b - B)$  by  $V(b) - V(B)$ , whereas `suest` estimates  $V(b - B)$  by  $V(b) - \text{cov}(b, B) - \text{cov}(B, b) + V(B)$ . One advantage of the second estimator is that it is always admissible, so the resulting test is always well defined. This quality is illustrated in the Hausman-type test of IIA comparing models `m1` and `m3`.

```
. suest m1 m3
```

```
Simultaneous results for m1, m3
```

```
Number of obs = 615
```

|             | Coef.     | Robust<br>Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-------------|-----------|---------------------|-------|-------|----------------------|-----------|
| m1_Prepaid  |           |                     |       |       |                      |           |
| age         | -.0086267 | .0051181            | -1.69 | 0.092 | -.0186579            | .0014045  |
| male        | .4368331  | .1692997            | 2.58  | 0.010 | .1050119             | .7686544  |
| _cons       | .2262309  | .2387678            | 0.95  | 0.343 | -.2417455            | .6942073  |
| m1_Uninsure |           |                     |       |       |                      |           |
| age         | -.0042211 | .0071001            | -0.59 | 0.552 | -.018137             | .0096947  |
| male        | .352813   | .2370142            | 1.49  | 0.137 | -.1117262            | .8173523  |
| _cons       | -1.232167 | .3267826            | -3.77 | 0.000 | -1.872649            | -.5916849 |
| m3_Uninsure |           |                     |       |       |                      |           |
| age         | -.0030281 | .008452             | -0.36 | 0.720 | -.0195937            | .0135374  |
| male        | .3528192  | .2812216            | 1.25  | 0.210 | -.1983649            | .9040033  |
| _cons       | -1.520525 | .4001354            | -3.80 | 0.000 | -2.304776            | -.7362735 |

```
. test [m1_Uninsure = m3_Uninsure], cons
```

```
(1) [m1_Uninsure]age - [m3_Uninsure]age = 0
```

```
(2) [m1_Uninsure]male - [m3_Uninsure]male = 0
```

```
(3) [m1_Uninsure]_cons - [m3_Uninsure]_cons = 0
```

```
 chi2(3) = 119.86
```

```
 Prob > chi2 = 0.0000
```

Although the classic Hausman test computed by `hausman` is not defined here, the `suest`-based test is just fine. We cannot reject the equality of the common coefficients across `m1` and `m3`.

A second advantage of the `suest` approach is that we can estimate the (co)variance matrix of the multivariate normal distribution of the estimators of the three models `m1`, `m2`, and `m3` and test that the common coefficients are equal.

*(Continued on next page)*

```
. suest m*
Simultaneous results for m1, m2, m3

Number of obs = 615


```

|             | Coef.     | Robust Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-------------|-----------|------------------|-------|-------|----------------------|-----------|
| m1_Prepaid  |           |                  |       |       |                      |           |
| age         | -.0086267 | .0051181         | -1.69 | 0.092 | -.0186579            | .0014045  |
| male        | .4368331  | .1692997         | 2.58  | 0.010 | .1050119             | .7686544  |
| _cons       | .2262309  | .2387678         | 0.95  | 0.343 | -.2417455            | .6942073  |
| m1_Uninsure |           |                  |       |       |                      |           |
| age         | -.0042211 | .0071001         | -0.59 | 0.552 | -.018137             | .0096947  |
| male        | .352813   | .2370142         | 1.49  | 0.137 | -.1117262            | .8173523  |
| _cons       | -1.232167 | .3267826         | -3.77 | 0.000 | -1.872649            | -.5916849 |
| m2_Prepaid  |           |                  |       |       |                      |           |
| age         | -.0090076 | .0052361         | -1.72 | 0.085 | -.0192702            | .0012551  |
| male        | .4545195  | .1753731         | 2.59  | 0.010 | .1107946             | .7982445  |
| _cons       | .2377445  | .2434992         | 0.98  | 0.329 | -.2395051            | .7149941  |
| m3_Uninsure |           |                  |       |       |                      |           |
| age         | -.0030281 | .008452          | -0.36 | 0.720 | -.0195937            | .0135374  |
| male        | .3528192  | .2812216         | 1.25  | 0.210 | -.1983649            | .9040033  |
| _cons       | -1.520525 | .4001354         | -3.80 | 0.000 | -2.304776            | -.7362735 |

```

. test [m1_Prepaid = m2_Prepaid], cons notest
(1) [m1_Prepaid]age - [m2_Prepaid]age = 0
(2) [m1_Prepaid]male - [m2_Prepaid]male = 0
(3) [m1_Prepaid]_cons - [m2_Prepaid]_cons = 0
. test [m1_Uninsure = m3_Uninsure], cons acc
(1) [m1_Prepaid]age - [m2_Prepaid]age = 0
(2) [m1_Prepaid]male - [m2_Prepaid]male = 0
(3) [m1_Prepaid]_cons - [m2_Prepaid]_cons = 0
(4) [m1_Uninsure]age - [m3_Uninsure]age = 0
(5) [m1_Uninsure]male - [m3_Uninsure]male = 0
(6) [m1_Uninsure]_cons - [m3_Uninsure]_cons = 0
 chi2(6) = 201.57
 Prob > chi2 = 0.0000

```

Again we do not find evidence against the correct specification of the multinomial logit for type of insurance. The classic Hausman test assumes that one of the estimators (named B in `hausman`) is efficient, that is, it has minimal (asymptotic) variance. This assumption ensures that  $V(b) - V(B)$  is an admissible, viable estimator for  $V(b - B)$ . The assumption that we have an efficient estimator is a restrictive one. It is violated, for instance, if our data are clustered. We want to adjust for clustering via a `vce(cluster clustvar)` option by requesting the cluster-adjusted sandwich estimator of variance. Consequently, in such a case, `hausman` cannot be used. This problem does not exist with the `suest` version of the Hausman test. To illustrate this feature, we suppose that the data are clustered by city—we constructed an imaginary variable `cityid` for this illustration. If we plan to apply `suest`, we would not specify the `vce(cluster clustvar)` option at the time of estimation. `suest` has a `vce(cluster clustvar)` option. Thus we do not need to refit the models; we can call `suest` and `test` right away.

```
. suest m1 m2, vce(cluster cityid)
```

```
Simultaneous results for m1, m2
```

```
Number of obs = 615
```

```
(Std. Err. adjusted for 260 clusters in cityid)
```

|             | Coef.     | Robust<br>Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-------------|-----------|---------------------|-------|-------|----------------------|-----------|
| m1_Prepaid  |           |                     |       |       |                      |           |
| age         | -.0086267 | .0049389            | -1.75 | 0.081 | -.0183068            | .0010534  |
| male        | .4368331  | .1624828            | 2.69  | 0.007 | .1183727             | .7552936  |
| _cons       | .2262309  | .2325479            | 0.97  | 0.331 | -.2295545            | .6820164  |
| m1_Uninsure |           |                     |       |       |                      |           |
| age         | -.0042211 | .00683              | -0.62 | 0.537 | -.0176077            | .0091654  |
| male        | .352813   | .2441603            | 1.45  | 0.148 | -.1257324            | .8313585  |
| _cons       | -1.232167 | .3235257            | -3.81 | 0.000 | -1.866266            | -.5980683 |
| m2_Prepaid  |           |                     |       |       |                      |           |
| age         | -.0090076 | .0050738            | -1.78 | 0.076 | -.0189521            | .000937   |
| male        | .4545195  | .1688441            | 2.69  | 0.007 | .1235911             | .785448   |
| _cons       | .2377445  | .2380297            | 1.00  | 0.318 | -.2287852            | .7042742  |

```
. test [m1_Prepaid = m2_Prepaid], cons
```

```
(1) [m1_Prepaid]age - [m2_Prepaid]age = 0
```

```
(2) [m1_Prepaid]male - [m2_Prepaid]male = 0
```

```
(3) [m1_Prepaid]_cons - [m2_Prepaid]_cons = 0
```

```
 chi2(3) = 5.66
```

```
 Prob > chi2 = 0.1292
```

suest provides some descriptive information about the clustering on cityid. Like any other estimation command, suest informs us that the standard errors are adjusted for clustering. The Hausman-type test obtained from the test command uses a simultaneous (co)variance of m1 and m2 appropriately adjusted for clustering. In this example, we still do not have reason to conclude that the multinomial logit model in this application is misspecified, that is, that IIA is violated.

◀

The multinomial logistic regression model is a special case of the conditional logistic regression model; see [R] [clogit](#). Like the multinomial logistic regression model, the conditional logistic regression model also makes the IIA assumption. Consider an example, introduced in [R] [asclogit](#), in which the demand for American, Japanese, and European cars is modeled in terms of the number of local dealers of the respective brands and of some individual attributes incorporated in interaction with the nationality of cars. We want to perform a Hausman-type test for IIA comparing the decision between all nationalities with the decision between non-American cars. The following code fragment demonstrates how to conduct a Hausman test for IIA via suest in this case.

```
. clogit choice japan europe maleJap maleEur incJap incEur dealer, group(id)
. estimates store allcars
. clogit choice japan maleJap incJap dealer if car!=1 , group(id)
. estimates store foreign
. suest allcars foreign
. test [allcars_choice=foreign_choice], common
```

## Testing proportionality

The applications of `suest` that we have discussed so far concern Hausman-type tests for misspecification. To test such a hypothesis, we compared two estimators that have the same probability limit if the hypothesis holds true, but otherwise have different limits. We may also want to compare the coefficients of models (estimators) for other substantive reasons. Although we most often want to test whether coefficients differ between models or estimators, we may occasionally want to test other constraints (see [Hausman and Ruud \[1987\]](#)).

### ➤ Example 2

In this example, using simulated labor market data for siblings, we consider two dependent variables, income (`inc`) and whether a person was promoted in the last year (`promo`). We apply familiar economic arguments regarding human capital, according to which employees have a higher income and a higher probability of being promoted, by having more human capital. Human capital is acquired through formal education (`edu`) and on-the-job training experience (`exp`). We study whether income and promotion are “two sides of the same coin”, that is, whether they reflect a common latent variable, “human capital”. Accordingly, we want to compare the effects of different aspects of human capital on different outcome variables.

We estimate fairly simple labor market equations. The income model is estimated with `regress`, and the estimation results are stored under the name `Inc`.

```
. use http://www.stata-press.com/data/r11/income
. regress inc edu exp male
```

| Source   | SS         | df  | MS         |
|----------|------------|-----|------------|
| Model    | 2058.44672 | 3   | 686.148908 |
| Residual | 4424.05183 | 273 | 16.2053181 |
| Total    | 6482.49855 | 276 | 23.4873136 |

Number of obs = 277

F( 3, 273) = 42.34

Prob > F = 0.0000

R-squared = 0.3175

Adj R-squared = 0.3100

Root MSE = 4.0256

| inc   | Coef.    | Std. Err. | t    | P> t  | [95% Conf. Interval] |
|-------|----------|-----------|------|-------|----------------------|
| edu   | 2.213707 | .243247   | 9.10 | 0.000 | 1.734828 2.692585    |
| exp   | 1.47293  | .231044   | 6.38 | 0.000 | 1.018076 1.927785    |
| male  | .5381153 | .4949466  | 1.09 | 0.278 | -.436282 1.512513    |
| _cons | 1.255497 | .3115808  | 4.03 | 0.000 | .642091 1.868904     |

```
. est store Inc
```

Being sibling data, the observations are clustered on family of origin, `famid`. In the estimation of the regression parameters, we did not specify a `vce(cluster famid)` option to adjust standard errors for clustering on family (`famid`). Thus the standard errors reported by `regress` are potentially flawed. This problem will, however, be corrected by specifying a `vce(cluster clustvar)` option with `suest`.

Next we estimate the promotion equation with `probit` and again store the results under an appropriate name.



```
. probit promo edu exp male, nolog
```

```
Probit regression
```

```
Number of obs = 277
LR chi2(3) = 49.76
Prob > chi2 = 0.0000
Pseudo R2 = 0.1357
```

```
Log likelihood = -158.43888
```

|  | promo | Coef.    | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|--|-------|----------|-----------|-------|-------|----------------------|-----------|
|  | edu   | .4593002 | .0898537  | 5.11  | 0.000 | .2831901             | .6354102  |
|  | exp   | .3593023 | .0805774  | 4.46  | 0.000 | .2013735             | .5172312  |
|  | male  | .2079983 | .1656413  | 1.26  | 0.209 | -.1166527            | .5326494  |
|  | _cons | -.464622 | .1088166  | -4.27 | 0.000 | -.6778985            | -.2513454 |

```
. est store Promo
```

The coefficients in the income and promotion equations definitely seem to be different. However, because the scales of the two variables are different, we would not expect the coefficients to be equal. The correct hypothesis here is that the proportionality of the coefficients of the two models, apart from the constant, are equal. This formulation would still reflect that the relative effects of the different aspects of human capital do not differ between the dependent variables. We can obtain a nonlinear Wald test for the hypothesis of proportionality by using the `testnl` command on the combined estimation results of the two estimators. Thus we first have to form the combined estimation results. At this point, we specify the `vce(cluster famid)` option to adjust for the clustering of observations on `famid`.

```
. suest Inc Promo, vce(cluster famid)
```

```
Simultaneous results for Inc, Promo
```

```
Number of obs = 277
```

```
(Std. Err. adjusted for 135 clusters in famid)
```

|             |       | Robust<br>Coef. | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-------------|-------|-----------------|-----------|-------|-------|----------------------|-----------|
| Inc_mean    |       |                 |           |       |       |                      |           |
|             | edu   | 2.213707        | .2483907  | 8.91  | 0.000 | 1.72687              | 2.700543  |
|             | exp   | 1.47293         | .1890583  | 7.79  | 0.000 | 1.102383             | 1.843478  |
|             | male  | .5381153        | .4979227  | 1.08  | 0.280 | -.4377952            | 1.514026  |
|             | _cons | 1.255497        | .3374977  | 3.72  | 0.000 | .594014              | 1.916981  |
| Inc_lnvar   |       |                 |           |       |       |                      |           |
|             | _cons | 2.785339        | .079597   | 34.99 | 0.000 | 2.629332             | 2.941347  |
| Promo_promo |       |                 |           |       |       |                      |           |
|             | edu   | .4593002        | .0886982  | 5.18  | 0.000 | .2854549             | .6331454  |
|             | exp   | .3593023        | .079772   | 4.50  | 0.000 | .2029522             | .5156525  |
|             | male  | .2079983        | .1691053  | 1.23  | 0.219 | -.1234419            | .5394386  |
|             | _cons | -.464622        | .1042169  | -4.46 | 0.000 | -.6688833            | -.2603607 |

(Continued on next page)

The standard errors reported by **suest** are identical to those reported by the respective estimation commands when invoked with the `vce(cluster famid)` option. We are now ready to test for proportionality:

$$H_0 : \frac{\beta_{\text{edu}}^{\text{Income}}}{\beta_{\text{edu}}^{\text{Promotion}}} = \frac{\beta_{\text{exp}}^{\text{Income}}}{\beta_{\text{exp}}^{\text{Promotion}}} = \frac{\beta_{\text{male}}^{\text{Income}}}{\beta_{\text{male}}^{\text{Promotion}}}$$

It is straightforward to translate this into syntax suitable for **testnl**, recalling that the coefficient of variable *v* in equation *eq* is denoted by `[eq]v`.

```
. testnl [Inc_mean]edu/[Promo_promo]edu =
> [Inc_mean]exp/[Promo_promo]exp =
> [Inc_mean]male/[Promo_promo]male
(1) [Inc_mean]edu/[Promo_promo]edu = [Inc_mean]exp/[Promo_promo]exp
(2) [Inc_mean]edu/[Promo_promo]edu = [Inc_mean]male/[Promo_promo]male
 chi2(2) = 0.61
 Prob > chi2 = 0.7385
```

From the evidence, we fail to reject the hypotheses that the coefficients of the income and promotion equations are proportional. Thus it is not unreasonable to assume that income and promotion can be explained by the same latent variable, “labor market success”.

A disadvantage of the nonlinear Wald test is that it is not invariant with respect to representation: a Wald test for a mathematically equivalent formulation of the nonlinear constraint usually leads to a different test result. An equivalent formulation of the proportionality hypothesis is

$$H_0: \beta_{\text{edu}}^{\text{Income}} \beta_{\text{exp}}^{\text{Promotion}} = \beta_{\text{edu}}^{\text{Promotion}} \beta_{\text{exp}}^{\text{Income}} \quad \text{and} \\ \beta_{\text{edu}}^{\text{Income}} \beta_{\text{male}}^{\text{Promotion}} = \beta_{\text{edu}}^{\text{Promotion}} \beta_{\text{male}}^{\text{Income}}$$

This formulation is “more linear” in the coefficients. The asymptotic  $\chi^2$  distribution of the nonlinear Wald statistic can be expected to be more accurate for this representation.

```
. testnl ([Inc_mean]edu*[Promo_promo]exp = [Inc_mean]exp*[Promo_promo]edu)
> ([Inc_mean]edu*[Promo_promo]male = [Inc_mean]male*[Promo_promo]edu)
(1) [Inc_mean]edu*[Promo_promo]exp = [Inc_mean]exp*[Promo_promo]edu
(2) [Inc_mean]edu*[Promo_promo]male = [Inc_mean]male*[Promo_promo]edu
 chi2(2) = 0.46
 Prob > chi2 = 0.7936
```

Here the two representations lead to similar test statistics and *p*-values. As before, we fail to reject the hypothesis of proportionality of the coefficients of the two models.

◀

## Testing cross-model hypotheses

### ► Example 3

In this example, we demonstrate how some cross-model hypotheses can be tested using the facilities already available in most estimation commands. This demonstration will explain the intricate relationship between the cluster adjustment of the robust estimator of variance and the **suest** command. It will also be made clear that a new facility is required to perform more general cross-model testing.

We want to test whether the effect of  $x_1$  on the binary variable  $y_1$  is the same as the effect of  $x_2$  on the binary  $y_2$ ; see [Clogg, Petkova, and Haritou \(1995\)](#). In this setting,  $x_1$  may equal  $x_2$ , and  $y_1$  may equal  $y_2$ . We assume that logistic regression models can be used to model the responses, and for simplicity, we ignore further predictor variables in these models. If the two logit models are fit on independent samples so that the estimators are (stochastically) independent, a Wald test for `_b[x1] = _b[x2]` rejects the null hypothesis if

$$\frac{\widehat{b}(x_1) - \widehat{b}(x_2)}{\left[ \widehat{\sigma}^2 \left\{ \widehat{b}(x_1) \right\} + \widehat{\sigma}^2 \left\{ \widehat{b}(x_2) \right\} \right]^{1/2}}$$

is larger than the appropriate  $\chi_1^2$  threshold. If the models are fit on the *same* sample (or on dependent samples), so that the estimators are stochastically dependent, the above test that ignores the covariance between the estimators is not appropriate.

It is instructive to see how this problem can be tackled by “stacking” data. In the stacked format, we doubled the number of observations. The dependent variable is  $y_1$  in the first half of the data and is  $y_2$  in the second half of the data. The predictor variable  $z_1$  is set to  $x_1$  in the first half of the expanded data and to 0 in the rest. Similarly,  $z_2$  is 0 in the first half and  $x_2$  in the second half. The following diagram illustrates the transformation, in the terminology of the `reshape` command, from wide to long format.

$$\left( \begin{array}{ccccc} \text{id} & y_1 & y_2 & x_1 & x_2 \\ \hline 1 & y_{11} & y_{21} & x_{11} & x_{21} \\ 2 & y_{12} & y_{22} & x_{12} & x_{22} \\ 3 & y_{13} & y_{23} & x_{13} & x_{23} \end{array} \right) \Rightarrow \left( \begin{array}{ccccc} \text{id} & y & z_1 & z_2 & \text{model} \\ \hline 1 & y_{11} & x_{11} & 0 & 1 \\ 2 & y_{12} & x_{12} & 0 & 1 \\ 3 & y_{13} & x_{13} & 0 & 1 \\ 1 & y_{21} & 0 & x_{21} & 2 \\ 2 & y_{22} & 0 & x_{22} & 2 \\ 3 & y_{23} & 0 & x_{23} & 2 \end{array} \right)$$

The observations in the long format data organization are *clustered* on the original subjects and are identified with the identifier `id`. The clustering on `id` has to be accounted for when fitting a simultaneous model. The simplest way to deal with clustering is to use the cluster adjustment of the robust or sandwich estimator; see [\[P\] `\_robust`](#). The data manipulation can be accomplished easily with the `stack` command; see [\[D\] `stack`](#). Subsequently, we fit a simultaneous logit model and perform a Wald test for the hypothesis that the coefficients of `z1` and `z2` are the same. A full setup to obtain the cross-model Wald test could then be as follows:

```
. generate zero = 0 // a variable that is always 0
. generate one = 1 // a variable that is always 1
. generate two = 2 // a variable that is always 2
. stack id y1 x1 zero one id y2 zero x2 two, into(id y z1 z2 model)
. generate model2 = (model==2)
. logit y model2 z1 z2, vce(cluster id)
. test _b[z1] = _b[z2]
```

The coefficient of `z1` represents the effect of `x1` on `y1`, and similarly, `z2` for the effect of `x2` on `y2`. The variable `model2` is a dummy for the “second model”, which is included to allow the intercept in the second model to differ from that in the first model. The estimates of the coefficient of `z1` and its standard error in the combined model are the same as the estimates of the coefficient of `z1` and its standard error if we fit the model on the unstacked data.

```
. logit y1 x1, vce(robust)
```

The `vce(cluster clustvar)` option specified with the `logit` command for the stacked data ensures that the covariances of `_b[z1]` and `_b[z2]` are indeed estimated. This estimation ensures that the Wald test for the equality of the coefficients is correct. If we had not specified the `vce(cluster clustvar)` option, the (co)variance matrix of the coefficients would have been block-diagonal; i.e., the covariances of `_b[z1]` and `_b[z2]` would have been 0. Then `test` would have effectively used the invalid formula for the Wald test for two independent samples.

In this example, the two logit models were fit on the same data. The same setup would apply, without modification, when the two logit models were fit on overlapping data that resulted, for instance, if the *y* or *x* variables were missing in some observations.

The `suest` command allows us to obtain the above Wald test more efficiently by avoiding the data manipulation, obviating the need to fit a model with twice the number of coefficients. The test statistic produced by the above code fragment is *identical* to that obtained via `suest` on the original (unstacked) data:

```
. logit y1 x1
. estimates store M1
. logit y2 x2
. estimates store M2
. suest M1 M2
. test [M1]x1=[M2]x2
```

The stacking method can be applied not only to the testing of cross-model hypotheses for logit models but also to any estimation command that supports the `vce(cluster clustvar)` option. The stacking approach clearly generalizes to stacking more than two logit or other models, testing more general linear hypotheses, and testing nonlinear cross-model hypotheses (see [R] [testnl](#)). In all these cases, `suest` would yield identical statistical results but at smaller costs in terms of data management, computer storage, and computer time.

Is `suest` nothing but a convenience command? No, there are two disadvantages to the stacking method, both of which are resolved via `suest`. First, if the models include ancillary parameters (in a regression model, the residual variance; in an ordinal response model, the cutpoints; and in lognormal survival-time regression, the time scale parameter), these parameters are constrained to be equal between the stacked models. In `suest`, this constraint is relaxed. Second, the stacking method does not generalize to compare different statistical models, such as a probit model and a regression model. As demonstrated in the previous section, `suest` can deal with this situation.

## Saved results

`suest` saves the following in `e()`:

|                          |                                                |
|--------------------------|------------------------------------------------|
| Scalars                  |                                                |
| <code>e(N)</code>        | number of observations                         |
| <code>e(N_clust)</code>  | number of clusters                             |
| <code>e(rank)</code>     | rank of <code>e(V)</code>                      |
| Macros                   |                                                |
| <code>e(cmd)</code>      | <code>suest</code>                             |
| <code>e(eqnames#)</code> | original names of equations of model #         |
| <code>e(names)</code>    | list of model names                            |
| <code>e(wtype)</code>    | weight type                                    |
| <code>e(wexp)</code>     | weight expression                              |
| <code>e(clustvar)</code> | name of cluster variable                       |
| <code>e(vce)</code>      | <i>vcetype</i> specified in <code>vce()</code> |
| <code>e(vcetype)</code>  | title used to label Std. Err.                  |
| Matrices                 |                                                |
| <code>e(b)</code>        | stacked coefficient vector of the models       |
| <code>e(V)</code>        | (co)variance matrix                            |
| Functions                |                                                |
| <code>e(sample)</code>   | marks estimation sample                        |

## Methods and formulas

`suest` is implemented as an ado-file.

The estimation of the simultaneous (co)variance of a series of  $k$  estimators is a nonstandard application of the sandwich estimator, as implemented by the command [\[P\] \\_robust](#). You may want to read this manual entry before reading further.

The starting point is that we have fit  $k$  different models on the *same* data—partially overlapping or nonoverlapping data are special cases. We want to derive the *simultaneous* distribution of these  $k$  estimators, for instance, to test a cross-estimator hypothesis  $H_0$ . As in the framework of Hausman testing,  $H_0$  will often be of the form that different estimators have the same probability limit under some hypothesis, while the estimators have different limits if the hypothesis is violated.

We consider (vector) estimators  $\hat{\beta}_i$  to be defined as “the” solution of the estimation equations  $\mathbf{G}_i$ ,

$$\mathbf{G}_i(\mathbf{b}_i) = \sum_j w_{ij} \mathbf{u}_{ij}(\mathbf{b}_i) = \mathbf{0}, \quad i = 1, \dots, k$$

We refer to the  $\mathbf{u}_{ij}$  as the “scores”. Specifying some weights  $w_{ij} = 0$  trivially accommodates for partially overlapping or even disjointed data. Under “suitable regularity conditions” (see [White 1982](#) and [1996](#) for details), the  $\hat{\beta}_i$  are asymptotically normally distributed, with the variance estimated consistently by the sandwich estimator

$$V_i = \text{Var}(\hat{\beta}_i) = \mathbf{D}_i^{-1} \sum_j w_{ij} \mathbf{u}_{ij} \mathbf{u}_{ij}' \mathbf{D}_i^{-1}$$

where  $\mathbf{D}_i$  is the Jacobian of  $\mathbf{G}_i$  evaluated at  $\hat{\beta}_i$ . In the context of maximum likelihood estimation,  $\mathbf{D}_i$  can be estimated consistently by (minus) the Hessian of the log likelihood or by the Fisher information matrix. If the model is also well specified, the sandwiched term  $(\sum_j w_{ij} \mathbf{u}_{ij} \mathbf{u}_{ij}')'$  converges in probability to  $\mathbf{D}_i$ , so  $V_i$  may be consistently estimated by  $\mathbf{D}_i^{-1}$ .

To derive the simultaneous distribution of the estimators, we consider the “stacked” estimation equation,

$$\mathbf{G}(\hat{\beta}) = \begin{bmatrix} \mathbf{G}_1(\hat{\beta}_1)' & \mathbf{G}_1(\hat{\beta}_2)' & \dots & \mathbf{G}_k(\hat{\beta}_k)' \end{bmatrix}' = \mathbf{0}$$

Under “suitable regularity conditions” (see [White \[1996\]](#) for details),  $\hat{\beta}$  is asymptotically *jointly* normally distributed. The Jacobian and scores of the simultaneous equation are easily expressed in the Jacobian and scores of the separate equations. The Jacobian of  $\mathbf{G}$ ,

$$\mathbf{D}(\hat{\beta}) = \left. \frac{d\mathbf{G}(\beta)}{d\beta} \right|_{\beta=\hat{\beta}}$$

is block diagonal with blocks  $\mathbf{D}_1, \dots, \mathbf{D}_k$ . The inverse of  $\mathbf{D}(\hat{\beta})$  is again block diagonal, with the inverses of  $\mathbf{D}_i$  on the diagonal. The scores  $\mathbf{u}$  of  $\mathbf{G}$  are simply obtained as the *concatenated* scores of the separate equations:

$$\mathbf{u}_j = (\mathbf{u}_{1j}' \quad \mathbf{u}_{2j}' \quad \dots \quad \mathbf{u}_{kj}')'$$

Out-of-sample (i.e., where  $w_{ij} = 0$ ) values of the score variables are defined as 0 (thus we drop the  $i$  subscript from the common weight variable). The sandwich estimator for the asymptotic variance of  $\hat{\beta}$  reads

$$V = \text{Var}(\hat{\beta}) = \mathbf{D}(\hat{\beta})^{-1} \left( \sum_j w_j \mathbf{u}_j \mathbf{u}_j' \right) \mathbf{D}(\hat{\beta})^{-1}$$

Taking a “partitioned” look at this expression, we see that  $V(\hat{\beta}_i)$  is estimated by

$$\mathbf{D}_i^{-1} \left( \sum_j w_j \mathbf{u}_{ij} \mathbf{u}_{ij}' \right) \mathbf{D}_i^{-1}$$

which is, yet again, the familiar sandwich estimator for  $\hat{\beta}_i$  based on the separate estimation equation  $\mathbf{G}_i$ . Thus considering several estimators simultaneously in this way does not affect the estimators of the asymptotic variances of these estimators. However, as a bonus of stacking, we obtained a sandwich-type estimate of the *covariance*  $V_{ih}$  of estimators  $\hat{\beta}_i$  and  $\hat{\beta}_h$ ,

$$V_{ih} = \text{Cov}(\hat{\beta}_i, \hat{\beta}_h) = \mathbf{D}_i^{-1} \left( \sum_j w_j \mathbf{u}_{ij} \mathbf{u}_{ih}' \right) \mathbf{D}_h^{-1}$$

which is also obtained by [White \(1982\)](#).

This estimator for the covariance of estimators is an application of the cluster modification of the sandwich estimator proposed by [Rogers \(1993\)](#). Consider the stacked data format as discussed in the logit example, and assume that Stata would be able to estimate a “stacked model” in which different models apply to different observations, for example, a probit model for the first half, a regression model for the second half, and a one-to-one cluster relation between the first and second half. If there are no common parameters to both models, the score statistics of parameters for the stacked models are zero in the half of the data in which they do not occur. In Rogers’ method, we have to sum the score statistics over the observations within a cluster. This step boils down to concatenating the score statistics at the level of the cluster.

We compare the sandwich estimator of the (co)variance  $V_{12}$  of two estimators with the estimator of variance  $\tilde{V}_{12}$  applied in the classic Hausman test. Hausman (1978) showed that if  $\hat{\beta}_1$  is consistent under  $H_0$  and  $\hat{\beta}_2$  is efficient under  $H_0$ , then asymptotically

$$\text{Cov}(\hat{\beta}_1, \hat{\beta}_2) = \text{Var}(\hat{\beta}_2)$$

and so  $\text{var}(\hat{\beta}_1 - \hat{\beta}_2)$  is consistently estimated by  $V_1 - V_2$ .

## Acknowledgment

`suest` was written by Jeroen Weesie, Department of Sociology, Utrecht University, The Netherlands. This research is supported by grant PGS 50-370 by The Netherlands Organization for Scientific Research.

An earlier version of `suest` was published in the *Stata Technical Bulletin* (1999). The current version of `suest` is not backward compatible with the STB version because of the introduction of new ways to manage estimation results via the `estimates` command.

## References

- Arminger, G. 1990. Testing against misspecification in parametric rate models. In *Event History Analysis in Life Course Research*, ed. K. U. Mayer and N. B. Tuma, 146–158. Madison: University of Wisconsin Press.
- Clogg, C. C., E. Petkova, and A. Haritou. 1995. Statistical methods for comparing regression coefficients between models. *American Journal of Sociology* 100: 1261–1312. (With comments by P. D. Allison and a reply by C. C. Clogg, E. Petkova, and T. Cheng).
- Gourieroux, C., and A. Monfort. 1997. *Time Series and Dynamic Models*. Trans. ed. G. M. Gallo. Cambridge: Cambridge University Press.
- Hausman, J. A. 1978. Specification tests in econometrics. *Econometrica* 46: 1251–1271.
- Hausman, J. A., and D. L. McFadden. 1984. Specification tests for the multinomial logit model. *Econometrica* 52: 1219–1240.
- Hausman, J. A., and P. A. Ruud. 1987. Specifying and testing econometric models for rank-ordered data. *Journal of Econometrics* 34: 83–104.
- Huber, P. J. 1967. The behavior of maximum likelihood estimates under nonstandard conditions. In Vol. 1 of *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 221–233. Berkeley: University of California Press.
- Rogers, W. H. 1993. [sg16.4: Comparison of nbreg and glm for negative binomial](#). *Stata Technical Bulletin* 16: 7. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 82–84. College Station, TX: Stata Press.
- Weesie, J. 1999. [sg121: Seemingly unrelated estimation and the cluster-adjusted sandwich estimator](#). *Stata Technical Bulletin* 52: 34–47. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 231–248. College Station, TX: Stata Press.
- White, H. 1982. Maximum likelihood estimation of misspecified models. *Econometrica* 50: 1–25.
- . 1996. *Estimation, Inference and Specification Analysis*. Cambridge: Cambridge University Press.

## Also see

- [R] **estimates** — Save and manipulate estimation results
- [R] **hausman** — Hausman specification test
- [R] **lincom** — Linear combinations of estimators
- [R] **nlcom** — Nonlinear combinations of estimators
- [R] **test** — Test linear hypotheses after estimation
- [R] **testnl** — Test nonlinear hypotheses after estimation
- [P] **\_robust** — Robust variance estimates



Syntax

```
summarize [varlist] [if] [in] [weight] [, options]
```

| options | description |
|---------|-------------|
|---------|-------------|

|                              |                                                                                   |
|------------------------------|-----------------------------------------------------------------------------------|
| Main                         |                                                                                   |
| <code>detail</code>          | display additional statistics                                                     |
| <code>meanonly</code>        | suppress the display; calculate only the mean; programmer's option                |
| <code>format</code>          | use variable's display format                                                     |
| <code>separator(#)</code>    | draw separator line after every # variables; default is <code>separator(5)</code> |
| <code>display_options</code> | control spacing and base and empty cells                                          |

`varlist` may contain factor variables; see [U] 11.4.3 Factor variables.

`varlist` may contain time-series operators; see [U] 11.4.4 Time-series varlists.

`by` is allowed; see [D] `by`.

`aweight`s, `fweight`s, and `iweight`s are allowed. However, `iweight`s may not be used with the `detail` option; see [U] 11.1.6 `weight`.

Menu

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Summary statistics

Description

`summarize` calculates and displays a variety of univariate summary statistics. If no `varlist` is specified, summary statistics are calculated for all the variables in the dataset.

Also see [R] `ci` for calculating the standard error and confidence intervals of the mean.

Options

|                                                                                                                                                                                                                                                                                                                                |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Main                                                                                                                                                                                                                                                                                                                           |
| <code>detail</code> produces additional statistics, including skewness, kurtosis, the four smallest and largest values, and various percentiles.                                                                                                                                                                               |
| <code>meanonly</code> , which is allowed only when <code>detail</code> is not specified, suppresses the display of results and calculation of the variance. Ado-file writers will find this useful for fast calls.                                                                                                             |
| <code>format</code> requests that the summary statistics be displayed using the display formats associated with the variables rather than the default <code>g</code> display format; see [U] 12.5 Formats: Controlling how data are displayed.                                                                                 |
| <code>separator(#)</code> specifies how often to insert separation lines into the output. The default is <code>separator(5)</code> , meaning that a line is drawn after every five variables. <code>separator(10)</code> would draw a line after every 10 variables. <code>separator(0)</code> suppresses the separation line. |

*display\_options*: vsquish, noemptycells, baselevels, allbaselevels; see [\[R\] estimation options](#).

## Remarks

`summarize` can produce two different sets of summary statistics. Without the `detail` option, the number of nonmissing observations, the mean and standard deviation, and the minimum and maximum values are presented. With `detail`, the same information is presented along with the variance, skewness, and kurtosis; the four smallest and four largest values; and the 1st, 5th, 10th, 25th, 50th (median), 75th, 90th, 95th, and 99th percentiles.

### ► Example 1: summarize with the separator() option

We have data containing information on various automobiles, among which is the variable `mpg`, the mileage rating. We can obtain a quick summary of the `mpg` variable by typing

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. summarize mpg
```

| Variable | Obs | Mean    | Std. Dev. | Min | Max |
|----------|-----|---------|-----------|-----|-----|
| mpg      | 74  | 21.2973 | 5.785503  | 12  | 41  |

We see that we have 74 observations. The mean of `mpg` is 21.3 miles per gallon, and the standard deviation is 5.79. The minimum is 12, and the maximum is 41.

If we had not specified the variable (or variables) we wanted to summarize, we would have obtained summary statistics on all the variables in the dataset:

```
. summarize, separator(4)
```

| Variable     | Obs | Mean     | Std. Dev. | Min  | Max   |
|--------------|-----|----------|-----------|------|-------|
| make         | 0   |          |           |      |       |
| price        | 74  | 6165.257 | 2949.496  | 3291 | 15906 |
| mpg          | 74  | 21.2973  | 5.785503  | 12   | 41    |
| rep78        | 69  | 3.405797 | .9899323  | 1    | 5     |
| headroom     | 74  | 2.993243 | .8459948  | 1.5  | 5     |
| trunk        | 74  | 13.75676 | 4.277404  | 5    | 23    |
| weight       | 74  | 3019.459 | 777.1936  | 1760 | 4840  |
| length       | 74  | 187.9324 | 22.26634  | 142  | 233   |
| turn         | 74  | 39.64865 | 4.399354  | 31   | 51    |
| displacement | 74  | 197.2973 | 91.83722  | 79   | 425   |
| gear_ratio   | 74  | 3.014865 | .4562871  | 2.19 | 3.89  |
| foreign      | 74  | .2972973 | .4601885  | 0    | 1     |

There are only 69 observations on `rep78`, so some of the observations are missing. There are no observations on `make` because it is a string variable.

The idea of the mean is quite old (Plackett 1958), but its extension to a scheme of moment-based measures was not done until the end of the 19th century. Between 1893 and 1905, Pearson discussed and named the standard deviation, skewness, and kurtosis, but he was not the first to use any of these. Thiele (1889), in contrast, had earlier firmly grasped the notion that the  $m_r$  provide a systematic basis for discussing distributions. However, even earlier anticipations can also be found. For example, Euler in 1778 used  $m_2$  and  $m_3$  in passing in a treatment of estimation (Hald 1998, 87), but seemingly did not build on that.

Similarly, the idea of the median is quite old. The history of the interquartile range is tangled up with that of the probable error, a long-popular measure. Extending this in various ways to a more general approach based on quantiles (to use a later term) occurred to several people in the nineteenth century. Galton (1875) is a nice example, particularly because he seems so close to the key idea of the quantiles as a function, which took another century to reemerge strongly.

Thorvald Nicolai Thiele (1838–1910) was a Danish scientist who worked in astronomy, mathematics, actuarial science, and statistics. He made many pioneering contributions to statistics, several of which were overlooked until recently. Thiele advocated graphical analysis of residuals checking for trends, symmetry of distributions, and changes of sign, and he even warned against overinterpreting such graphs.

► Example 2: summarize with the detail option

The `detail` option provides all the information of a normal `summarize` and more. The format of the output also differs, as shown here:

```
. summarize mpg, detail
```

| Mileage (mpg) |    |          |             |          |
|---------------|----|----------|-------------|----------|
| Percentiles   |    | Smallest |             |          |
| 1%            | 12 | 12       |             |          |
| 5%            | 14 | 12       |             |          |
| 10%           | 14 | 14       | Obs         | 74       |
| 25%           | 18 | 14       | Sum of Wgt. | 74       |
| 50%           | 20 | Largest  | Mean        | 21.2973  |
|               |    |          | Std. Dev.   | 5.785503 |
| 75%           | 25 | 34       |             |          |
| 90%           | 29 | 35       | Variance    | 33.47205 |
| 95%           | 34 | 35       | Skewness    | .9487176 |
| 99%           | 41 | 41       | Kurtosis    | 3.975005 |

As in the previous example, we see that the mean of `mpg` is 21.3 miles per gallon and that the standard deviation is 5.79. We also see the various percentiles. The median of `mpg` (the 50th percentile) is 20 miles per gallon. The 25th percentile is 18, and the 75th percentile is 25.

When we performed `summarize`, we learned that the minimum and maximum were 12 and 41, respectively. We now see that the four smallest values in our dataset are 12, 12, 14, and 14. The four largest values are 34, 35, 35, and 41. The skewness of the distribution is 0.95, and the kurtosis is 3.98. (A normal distribution would have a skewness of 0 and a kurtosis of 3.)

*Skewness* is a measure of the lack of symmetry of a distribution. If the distribution is symmetric, the coefficient of skewness is 0. If the coefficient is negative, the median is usually greater than the mean and the distribution is said to be skewed left. If the coefficient is positive, the median is usually less than the mean and the distribution is said to be skewed right. *Kurtosis* (from the Greek *kyrtosis*, meaning curvature) is a measure of peakedness of a distribution. The smaller the coefficient of kurtosis, the flatter the distribution. The normal distribution has a coefficient of kurtosis of 3 and provides a convenient benchmark.

➤ Example 3: summarize with the by prefix

summarize can usefully be combined with the `by varlist:` prefix. In our dataset, we have a variable, `foreign`, that distinguishes foreign and domestic cars. We can obtain summaries of `mpg` and `weight` within each subgroup by typing

```
. by foreign: summarize mpg weight
```

---

```
-> foreign = Domestic
```

| Variable | Obs | Mean     | Std. Dev. | Min  | Max  |
|----------|-----|----------|-----------|------|------|
| mpg      | 52  | 19.82692 | 4.743297  | 12   | 34   |
| weight   | 52  | 3317.115 | 695.3637  | 1800 | 4840 |

---

```
-> foreign = Foreign
```

| Variable | Obs | Mean     | Std. Dev. | Min  | Max  |
|----------|-----|----------|-----------|------|------|
| mpg      | 22  | 24.77273 | 6.611187  | 14   | 41   |
| weight   | 22  | 2315.909 | 433.0035  | 1760 | 3420 |

Domestic cars in our dataset average 19.8 miles per gallon, whereas foreign cars average 24.8.

Because `by varlist:` can be combined with `summarize`, it can also be combined with `summarize, detail`:

```
. by foreign: summarize mpg, detail
```

---

```
-> foreign = Domestic
```

|             |      | Mileage (mpg) |                    |
|-------------|------|---------------|--------------------|
| Percentiles |      | Smallest      |                    |
| 1%          | 12   | 12            |                    |
| 5%          | 14   | 12            |                    |
| 10%         | 14   | 14            | Obs 52             |
| 25%         | 16.5 | 14            | Sum of Wgt. 52     |
| 50%         | 19   |               | Mean 19.82692      |
|             |      | Largest       | Std. Dev. 4.743297 |
| 75%         | 22   | 28            |                    |
| 90%         | 26   | 29            | Variance 22.49887  |
| 95%         | 29   | 30            | Skewness .7712432  |
| 99%         | 34   | 34            | Kurtosis 3.441459  |

---

```
-> foreign = Foreign
```

|             |      | Mileage (mpg) |                    |
|-------------|------|---------------|--------------------|
| Percentiles |      | Smallest      |                    |
| 1%          | 14   | 14            |                    |
| 5%          | 17   | 17            |                    |
| 10%         | 17   | 17            | Obs 22             |
| 25%         | 21   | 18            | Sum of Wgt. 22     |
| 50%         | 24.5 |               | Mean 24.77273      |
|             |      | Largest       | Std. Dev. 6.611187 |
| 75%         | 28   | 31            |                    |
| 90%         | 35   | 35            | Variance 43.70779  |
| 95%         | 35   | 35            | Skewness .657329   |
| 99%         | 41   | 41            | Kurtosis 3.10734   |

## □ Technical note

`summarize` respects display formats if we specify the `format` option. When we type `summarize price weight`, we obtain

```
. summarize price weight
```

| Variable | Obs | Mean     | Std. Dev. | Min  | Max   |
|----------|-----|----------|-----------|------|-------|
| price    | 74  | 6165.257 | 2949.496  | 3291 | 15906 |
| weight   | 74  | 3019.459 | 777.1936  | 1760 | 4840  |

The display is accurate but is not as aesthetically pleasing as we may wish, particularly if we plan to use the output directly in published work. By placing formats on the variables, we can control how the table appears:

```
. format price weight %9.2fc
```

```
. summarize price weight, format
```

| Variable | Obs | Mean     | Std. Dev. | Min      | Max       |
|----------|-----|----------|-----------|----------|-----------|
| price    | 74  | 6,165.26 | 2,949.50  | 3,291.00 | 15,906.00 |
| weight   | 74  | 3,019.46 | 777.19    | 1,760.00 | 4,840.00  |

□

If you specify a weight (see [\[U\] 11.1.6 weight](#)), each observation is multiplied by the value of the weighting expression before the summary statistics are calculated so that the weighting expression is interpreted as the discrete density of each observation.

## ➤ Example 4: summarize with factor variables

You can also use `summarize` to obtain summary statistics for factor variables. For example, if you type

```
. summarize i.rep78
```

| Variable | Obs | Mean     | Std. Dev. | Min | Max |
|----------|-----|----------|-----------|-----|-----|
| rep78    |     |          |           |     |     |
| 2        | 69  | .115942  | .3225009  | 0   | 1   |
| 3        | 69  | .4347826 | .4993602  | 0   | 1   |
| 4        | 69  | .2608696 | .4423259  | 0   | 1   |
| 5        | 69  | .1594203 | .3687494  | 0   | 1   |

you obtain the sample proportions for four of the five levels of the `rep78` variable. For example, 11.6% of the 69 cars with nonmissing values of `rep78` fall into repair category two. When you use factor-variable notation, the base category is suppressed by default. If you type

```
. summarize bn.rep78
```

| Variable | Obs | Mean     | Std. Dev. | Min | Max |
|----------|-----|----------|-----------|-----|-----|
| rep78    |     |          |           |     |     |
| 1        | 69  | .0289855 | .1689948  | 0   | 1   |
| 2        | 69  | .115942  | .3225009  | 0   | 1   |
| 3        | 69  | .4347826 | .4993602  | 0   | 1   |
| 4        | 69  | .2608696 | .4423259  | 0   | 1   |
| 5        | 69  | .1594203 | .3687494  | 0   | 1   |

the notation `bn.rep78` indicates that Stata should not suppress the base category so that we see the proportions for all five levels.

We could have used `tabulate oneway rep78` to obtain the sample proportions along with the cumulative proportions. Alternatively, we could have used `proportions rep78` to obtain the sample proportions along with the standard errors of the proportions instead of the standard deviations of the proportions.



➤ Example 5: summarize with weights

We have 1980 census data on each of the 50 states. Included in our variables is `medage`, the median age of the population of each state. If we type `summarize medage`, we obtain unweighted statistics:

```
. use http://www.stata-press.com/data/r11/census
(1980 Census data by state)
. summarize medage
```

| Variable | Obs | Mean  | Std. Dev. | Min  | Max  |
|----------|-----|-------|-----------|------|------|
| medage   | 50  | 29.54 | 1.693445  | 24.2 | 34.7 |

Also among our variables is `pop`, the population in each state. Typing `summarize medage [w=pop]` produces population-weighted statistics:

```
. summarize medage [w=pop]
(analytic weights assumed)
```

| Variable | Obs | Weight    | Mean     | Std. Dev. | Min  | Max  |
|----------|-----|-----------|----------|-----------|------|------|
| medage   | 50  | 225907472 | 30.11047 | 1.66933   | 24.2 | 34.7 |

The number listed under `Weight` is the sum of the weighting variable, `pop`, indicating that there are roughly 226 million people in the United States. The `pop`-weighted mean of `medage` is 30.11 (compared with 29.54 for the unweighted statistic), and the weighted standard deviation is 1.67 (compared with 1.69).



➤ Example 6: summarize with weights and the detail option

We can obtain detailed summaries of weighted data as well. When we do this, *all* the statistics are weighted, including the percentiles.

```
. summarize medage [w=pop], detail
(analytic weights assumed)
```

| Median age  |      |  |             |           |  |
|-------------|------|--|-------------|-----------|--|
| Percentiles |      |  | Smallest    |           |  |
| 1%          | 27.1 |  | 24.2        |           |  |
| 5%          | 27.7 |  | 26.1        |           |  |
| 10%         | 28.2 |  | 27.1        |           |  |
| 25%         | 29.2 |  | 27.4        |           |  |
| 50%         | 29.9 |  |             |           |  |
|             |      |  | Largest     |           |  |
| 75%         | 30.9 |  | 32          |           |  |
| 90%         | 32.1 |  | 32.1        |           |  |
| 95%         | 32.2 |  | 32.2        |           |  |
| 99%         | 34.7 |  | 34.7        |           |  |
|             |      |  | Obs         | 50        |  |
|             |      |  | Sum of Wgt. | 225907472 |  |
|             |      |  | Mean        | 30.11047  |  |
|             |      |  | Std. Dev.   | 1.66933   |  |
|             |      |  | Variance    | 2.786661  |  |
|             |      |  | Skewness    | .5281972  |  |
|             |      |  | Kurtosis    | 4.494223  |  |



□ Technical note

If you are writing a program and need to access the mean of a variable, the `meanonly` option provides for fast calls. For example, suppose that your program reads as follows:

```

program mean
 summarize '1', meanonly
 display " mean = " r(mean)
end

```

The result of executing this is

```

. mean price
 mean = 6165.2568

```



# Saved results

`summarize` saves the following in `r()`:

|                          |                               |                          |                               |
|--------------------------|-------------------------------|--------------------------|-------------------------------|
| Scalars                  |                               |                          |                               |
| <code>r(N)</code>        | number of observations        | <code>r(p50)</code>      | 50th percentile (detail only) |
| <code>r(mean)</code>     | mean                          | <code>r(p75)</code>      | 75th percentile (detail only) |
| <code>r(skewness)</code> | skewness (detail only)        | <code>r(p90)</code>      | 90th percentile (detail only) |
| <code>r(min)</code>      | minimum                       | <code>r(p95)</code>      | 95th percentile (detail only) |
| <code>r(max)</code>      | maximum                       | <code>r(p99)</code>      | 99th percentile (detail only) |
| <code>r(sum_w)</code>    | sum of the weights            | <code>r(Var)</code>      | variance                      |
| <code>r(p1)</code>       | 1st percentile (detail only)  | <code>r(kurtosis)</code> | kurtosis (detail only)        |
| <code>r(p5)</code>       | 5th percentile (detail only)  | <code>r(sum)</code>      | sum of variable               |
| <code>r(p10)</code>      | 10th percentile (detail only) | <code>r(sd)</code>       | standard deviation            |
| <code>r(p25)</code>      | 25th percentile (detail only) |                          |                               |

# Methods and formulas

Let  $x$  denote the variable on which we want to calculate summary statistics, and let  $x_i, i = 1, \dots, n$ , denote an individual observation on  $x$ . Let  $v_i$  be the weight, and if no weight is specified, define  $v_i = 1$  for all  $i$ .

Define  $V$  as the *sum of the weight*:

$$V = \sum_{i=1}^n v_i$$

Define  $w_i$  to be  $v_i$  normalized to sum to  $n$ ,  $w_i = v_i(n/V)$ .

The *mean*,  $\bar{x}$ , is defined as

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n w_i x_i$$

The *variance*,  $s^2$ , is defined as

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n w_i (x_i - \bar{x})^2$$

The *standard deviation*,  $s$ , is defined as  $\sqrt{s^2}$ .

Define  $m_r$  as the  $r$ th moment about the mean  $\bar{x}$ :

$$m_r = \frac{1}{n} \sum_{i=1}^n w_i (x_i - \bar{x})^r$$

The *coefficient of skewness* is then defined as  $m_3 m_2^{-3/2}$ . The *coefficient of kurtosis* is defined as  $m_4 m_2^{-2}$ .

Let  $x_{(i)}$  refer to the  $x$  in ascending order, and let  $w_{(i)}$  refer to the corresponding weights of  $x_{(i)}$ . The four smallest values are  $x_{(1)}$ ,  $x_{(2)}$ ,  $x_{(3)}$ , and  $x_{(4)}$ . The four largest values are  $x_{(n)}$ ,  $x_{(n-1)}$ ,  $x_{(n-2)}$ , and  $x_{(n-3)}$ .

To obtain the  $p$ th percentile, which we will denote as  $x_{[p]}$ , let  $P = np/100$ . Let

$$W_{(i)} = \sum_{j=1}^i w_{(j)}$$

Find the first index  $i$  such that  $W_{(i)} > P$ . The  $p$ th percentile is then

$$x_{[p]} = \begin{cases} \frac{x_{(i-1)} + x_{(i)}}{2} & \text{if } W_{(i-1)} = P \\ x_{(i)} & \text{otherwise} \end{cases}$$

## References

- David, H. A. 2001. First (?) occurrence of common terms in statistics and probability. In *Annotated Readings in the History of Statistics*, ed. H. A. David and A. W. F. Edwards, 209–246. New York: Springer.
- Galton, F. 1875. Statistics by intercomparison, with remarks on the law of frequency of error. *Philosophical Magazine* 49: 33–46.
- Gleason, J. R. 1997. [sg67: Univariate summaries with boxplots](#). *Stata Technical Bulletin* 36: 23–25. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 179–183. College Station, TX: Stata Press.
- . 1999. [sg67.1: Update to univar](#). *Stata Technical Bulletin* 51: 27–28. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 159–161. College Station, TX: Stata Press.
- Hald, A. 1998. *A History of Mathematical Statistics from 1750 to 1930*. New York: Wiley.
- Hamilton, L. C. 1996. *Data Analysis for Social Scientists*. Belmont, CA: Duxbury.
- . 2009. [Statistics with Stata \(Updated for Version 10\)](#). Belmont, CA: Brooks/Cole.
- Kirkwood, B. R., and J. A. C. Sterne. 2003. *Essential Medical Statistics*. 2nd ed. Malden, MA: Blackwell.
- Lauritzen, S. L. 2002. *Thiele: Pioneer in Statistics*. Oxford: Oxford University Press.
- Plackett, R. L. 1958. Studies in the history of probability and statistics: VII. The principle of the arithmetic mean. *Biometrika* 45: 130–135.
- Stuart, A., and J. K. Ord. 1994. *Kendall's Advanced Theory of Statistics: Distribution Theory, Vol I*. 6th ed. London: Arnold.
- Thiele, T. N. 1889. *Forelæsninger over Almindelig Igttagelseslære: Sandsynlighedsregning og mindste Kvadraters Methode*. Kjøbenhavn: C.A. Reitzel. (English translation included in Lauritzen 2002).
- Weisberg, H. F. 1992. *Central Tendency and Variability*. Newbury Park, CA: Sage.



## Also see

- [R] **ameans** — Arithmetic, geometric, and harmonic means
- [R] **centile** — Report centile and confidence interval
- [R] **mean** — Estimate means
- [R] **proportion** — Estimate proportions
- [R] **ratio** — Estimate ratios
- [R] **table** — Tables of summary statistics
- [R] **tabstat** — Display table of summary statistics
- [R] **tabulate, summarize()** — One- and two-way tables of summary statistics
- [R] **total** — Estimate totals
- [D] **codebook** — Describe data contents
- [D] **describe** — Describe data in memory or in file
- [D] **inspect** — Display simple summary of data's attributes
- [ST] **stsum** — Summarize survival-time data
- [SVY] **svy estimation** — Estimation commands for survey data
- [XT] **xtsum** — Summarize xt data

Syntax

```
sunflower yvar xvar [if] [in] [weight] [, options]
```

| options                                     | description                                                                  |
|---------------------------------------------|------------------------------------------------------------------------------|
| Main                                        |                                                                              |
| <code>nograph</code>                        | do not show graph                                                            |
| <code>notable</code>                        | do not show summary table; implied when <code>by()</code> is specified       |
| <code>marker_options</code>                 | affect rendition of markers drawn at the plotted points                      |
| Bins/Petals                                 |                                                                              |
| <code>binwidth(#)</code>                    | width of the hexagonal bins                                                  |
| <code>binar(#)</code>                       | aspect ratio of the hexagonal bins                                           |
| <code>bin_options</code>                    | affect rendition of hexagonal bins                                           |
| <code>light(#)</code>                       | minimum observations for a light sunflower; default is <code>light(3)</code> |
| <code>dark(#)</code>                        | minimum observations for a dark sunflower; default is <code>dark(13)</code>  |
| <code>xcenter(#)</code>                     | <i>x</i> -coordinate of the reference bin                                    |
| <code>ycenter(#)</code>                     | <i>y</i> -coordinate of the reference bin                                    |
| <code>petalweight(#)</code>                 | observations in a dark sunflower petal                                       |
| <code>petallength(#)</code>                 | length of sunflower petal as a percentage                                    |
| <code>petal_options</code>                  | affect rendition of sunflower petals                                         |
| <code>floweronly</code>                     | show petals only; do not render bins                                         |
| <code>nosinglepetal</code>                  | suppress single petals                                                       |
| Add plots                                   |                                                                              |
| <code>addplot(plot)</code>                  | add other plots to generated graph                                           |
| Y axis, X axis, Titles, Legend, Overall, By |                                                                              |
| <code>twoway_options</code>                 | any options documented in <a href="#">[G] twoway_options</a>                 |

| <i>bin_options</i>                         | description                    |
|--------------------------------------------|--------------------------------|
| <code>[1 d] bstyle(areastyle)</code>       | overall look of hexagonal bins |
| <code>[1 d] bcolor(colorstyle)</code>      | outline and fill color         |
| <code>[1 d] bfillcolor(colorstyle)</code>  | fill color                     |
| <code>[1 d] blstyle(linestyle)</code>      | overall look of outline        |
| <code>[1 d] blcolor(colorstyle)</code>     | outline color                  |
| <code>[1 d] blwidth(linewidthstyle)</code> | thickness of outline           |

| <i>petal_options</i>                       | description                       |
|--------------------------------------------|-----------------------------------|
| <code>[1 d] flstyle(linestyle)</code>      | overall style of sunflower petals |
| <code>[1 d] flcolor(colorstyle)</code>     | color of sunflower petals         |
| <code>[1 d] flwidth(linewidthstyle)</code> | thickness of sunflower petals     |

All options are *rightmost*; see [G] [concept: repeated options](#).  
`fweights` are allowed; see [U] [11.1.6 weight](#).

## Menu

Graphics > Smoothing and densities > Density-distribution sunflower plot

## Description

`sunflower` draws density-distribution sunflower plots (Plummer Jr. and Dupont 2003). These plots are useful for displaying bivariate data whose density is too great for conventional scatterplots to be effective.

A sunflower is several line segments of equal length, called petals, that radiate from a central point. There are two varieties of sunflowers: light and dark. Each petal of a light sunflower represents 1 observation. Each petal of a dark sunflower represents several observations. Dark and light sunflowers represent high- and medium-density regions of the data, and marker symbols represent individual observations in low-density regions.

The plane defined by the variables `yvar` and `xvar` is divided into contiguous hexagonal bins. The number of observations contained within a bin determines how the bin will be represented.

- When there are fewer than `light(#)` observations in a bin, each point is plotted using the usual marker symbols in a scatterplot.
- Bins with at least `light(#)` but fewer than `dark(#)` observations are represented by a light sunflower.
- Bins with at least `dark(#)` observations are represented by a dark sunflower.

## Options

Main

`nograph` prevents the graph from being generated.

`notable` prevents the summary table from being displayed. This option is implied when the `by()` option is specified.

`marker_options` affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [G] [marker\\_options](#).

---

#### Bins/Petals

---

`binwidth(#)` specifies the horizontal width of the hexagonal bins in the same units as *xvar*. By default,

$$\text{binwidth} = \max(\text{rbw}, \text{nbw})$$

where

$$\text{rbw} = \text{range of } xvar / 40$$

$$\text{nbw} = \text{range of } xvar / \max(1, \text{nb})$$

and

$$\text{nb} = \text{int}(\min(\sqrt{n}, 10 * \log_{10}(n)))$$

where

$$n = \text{the number of observations in the dataset}$$

`binar(#)` specifies the aspect ratio for the hexagonal bins. The height of the bins is given by

$$\text{binheight} = \text{binwidth} \times \# \times 2 / \sqrt{3}$$

where *binheight* and *binwidth* are specified in the units of *yvar* and *xvar*, respectively. The default is `binar(r)`, where *r* results in the rendering of regular hexagons.

`bin_options` affect how the hexagonal bins are rendered.

`lbstyle(areastyle)` and `dbstyle(areastyle)` specify the look of the light and dark hexagonal bins, respectively. The options listed below allow you to change each attribute, but `lbstyle()` and `dbstyle()` provide the starting points. See [G] [areastyle](#) for a list of available area styles.

`lbcolor(colorstyle)` and `dbcolor(colorstyle)` specify one color to be used both to outline the shape and to fill the interior of the light and dark hexagonal bins, respectively. See [G] [colorstyle](#) for a list of color choices.

`lbfc(colorstyle)` and `dbfc(colorstyle)` specify the color to be used to fill the interior of the light and dark hexagonal bins, respectively. See [G] [colorstyle](#) for a list of color choices.

`lbstyle(linestyle)` and `dbstyle(linestyle)` specify the overall style of the line used to outline the area, which includes its pattern (solid, dashed, etc.), thickness, and color. The other options listed below allow you to change the line's attributes, but `lbstyle()` and `dbstyle()` are the starting points. See [G] [linestyle](#) for a list of choices.

`lbcolor(colorstyle)` and `dbcolor(colorstyle)` specify the color to be used to outline the light and dark hexagonal bins, respectively. See [G] [colorstyle](#) for a list of color choices.

`lbwidth(linewidthstyle)` and `dbwidth(linewidthstyle)` specify the thickness of the line to be used to outline the light and dark hexagonal bins, respectively. See [G] [linewidthstyle](#) for a list of choices.

`light(#)` specifies the minimum number of observations needed for a bin to be represented by a light sunflower. The default is `light(3)`.

`dark(#)` specifies the minimum number of observations needed for a bin to be represented by a dark sunflower. The default is `dark(13)`.

`xcenter(#)` and `ycenter(#)` specify the center of the reference bin. The default values are the median values of `xvar` and `yvar`, respectively. The centers of the other bins are implicitly defined by the location of the reference bin together with the common bin width and height.

`petalweight(#)` specifies the number of observations represented by each petal of a dark sunflower. The default value is chosen so that the maximum number of petals on a dark sunflower is 14.

`petallength(#)` specifies the length of petals in the sunflowers. The value specified is interpreted as a percentage of half the bin width. The default is 100%.

`petal_options` affect how the sunflower petals are rendered.

`lflstyle(linestyle)` and `dflstyle(linestyle)` specify the overall style of the light and dark sunflower petals, respectively.

`lflcolor(colorstyle)` and `dflcolor(colorstyle)` specify the color of the light and dark sunflower petals, respectively.

`lflwidth(linewidthstyle)` and `dflwidth(linewidthstyle)` specify the width of the light and dark sunflower petals, respectively.

`floweronly` suppresses rendering of the bins. This option is equivalent to specifying `lbcolor(none)` and `dbcolor(none)`.

`nosinglepetal` suppresses flowers from being drawn in light bins that contain only 1 observation and dark bins that contain as many observations as the petal weight (see the `petalweight()` option).

#### Add plots

`addplot(plot)` provides a way to add other plots to the generated graph; see [G] [addplot\\_option](#).

#### Y axis, X axis, Titles, Legend, Overall, By

`twoway_options` are any of the options documented in [G] [twoway\\_options](#). These include options for titling the graph (see [G] [title\\_options](#)), options for saving the graph to disk (see [G] [saving\\_option](#)), and the `by()` option (see [G] [by\\_option](#)).

## Remarks

See Dupont (2009, 87–92) for a discussion of sunflower plots and how to create them using Stata.

### ► Example 1

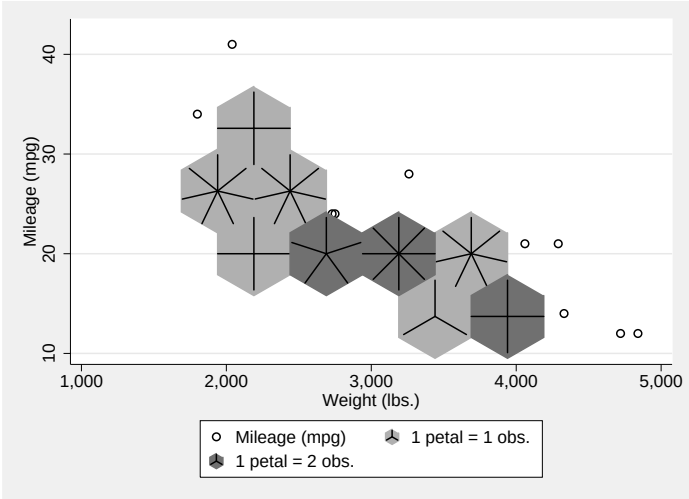
Using the auto dataset, we want to examine the relationship between `weight` and `mpg`. To do that, we type

(Continued on next page)

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. sunflower mpg weight, binwid(500) petalw(2) dark(8)
Bin width = 500
Bin height = 8.38703
Bin aspect ratio = .0145268
Max obs in a bin = 15
Light = 3
Dark = 8
X-center = 3190
Y-center = 20
Petal weight = 2
```

| flower type | petal weight | No. of petals | No. of flowers | estimated obs. | actual obs. |
|-------------|--------------|---------------|----------------|----------------|-------------|
| none        |              |               |                | 10             | 10          |
| light       | 1            | 3             | 1              | 3              | 3           |
| light       | 1            | 4             | 2              | 8              | 8           |
| light       | 1            | 7             | 3              | 21             | 21          |
| dark        | 2            | 4             | 1              | 8              | 8           |
| dark        | 2            | 5             | 1              | 10             | 9           |
| dark        | 2            | 8             | 1              | 16             | 15          |
|             |              |               |                | 76             | 74          |



The three darkly shaded sunflowers immediately catch our eyes, indicating a group of eight cars that are heavy (nearly 4,000 pounds) and fuel inefficient and two groups of cars that get about 20 miles per gallon and weight in the neighborhood of 3,000 pounds, one with 10 cars and one with 8 cars. The lighter sunflowers with seven petals each indicate groups of seven cars that share similar weight and fuel economy characteristics. To obtain the number of cars in each group, we counted the number of petals in each flower and consulted the graph legend to see how many observations each petal represents.

## Methods and formulas

`sunflower` is implemented as an ado-file.

## Acknowledgments

We thank William D. Dupont and W. Dale Plummer Jr. (Vanderbilt University), authors of the original `sunflower` command, for their assistance in producing this version.

## References

- Cleveland, W. S., and R. McGill. 1984. The many faces of a scatterplot. *Journal of the American Statistical Association* 79: 807–822.
- Dupont, W. D. 2009. *Statistical Modeling for Biomedical Researchers: A Simple Introduction to the Analysis of Complex Data*. 2nd ed. Cambridge: Cambridge University Press.
- Dupont, W. D., and W. D. Plummer Jr. 2005. Using density-distribution sunflower plots to explore bivariate relationships in dense data. *Stata Journal* 5: 371–384.
- Huang, C., J. A. McDonald, and W. Stuetzle. 1997. Variable resolution bivariate plots. *Journal of Computational and Graphical Statistics* 6: 383–396.
- Levy, D. 1999. *50 years of discovery: Medical milestones from the National Heart, Lung, and Blood Institute's Framingham Heart Study*. Hoboken, NJ: Center for Bio-Medical Communication.
- Plummer Jr., W. D., and W. D. Dupont. 2003. Density distribution sunflower plots. *Journal of Statistical Software* 8: 1–11.
- Steichen, T. J., and N. J. Cox. 1999. flower: Stata module to draw sunflower plots. Boston College Department of Economics, Statistical Software Components S393001. <http://ideas.repec.org/c/boc/bocode/s393001.html>.

Syntax

Basic syntax

```
sureg (depvar1 varlist1) (depvar2 varlist2) ... (depvarN varlistN)
[if] [in] [weight]
```

Full syntax

```
sureg ([eqname1:]depvar1a [depvar1b ...=]varlist1 [, noconstant])
([eqname2:]depvar2a [depvar2b ...=]varlist2 [, noconstant])
...
([eqnameN:]depvarNa [depvarNb ...=]varlistN [, noconstant])
[if] [in] [weight] [, options]
```

Explicit equation naming (*eqname*:) cannot be combined with multiple dependent variables in an equation specification.

| <i>options</i>                            | description                                                               |
|-------------------------------------------|---------------------------------------------------------------------------|
| Model                                     |                                                                           |
| <u>is</u> ure                             | iterate until estimates converge                                          |
| <u>con</u> straints( <i>constraints</i> ) | apply specified linear constraints                                        |
| df adj.                                   |                                                                           |
| <u>s</u> mall                             | report small-sample statistics                                            |
| dfk                                       | use small-sample adjustment                                               |
| dfk2                                      | use alternate adjustment                                                  |
| Reporting                                 |                                                                           |
| <u>l</u> evel(#)                          | set confidence level; default is level(95)                                |
| <u>co</u> rr                              | perform Breusch–Pagan test                                                |
| <u>no</u> cn <u>s</u> report              | do not display constraints                                                |
| <i>display_options</i>                    | control spacing and display of omitted variables and base and empty cells |
| Optimization                              |                                                                           |
| <i>optimization_options</i>               | control the optimization process; seldom used                             |
| <sup>†</sup> <u>no</u> header             | suppress header table from above coefficient table                        |
| <sup>†</sup> <u>no</u> table              | suppress coefficient table                                                |
| <sup>†</sup> <u>co</u> ef <u>l</u> egend  | display coefficients’ legend instead of coefficient table                 |



<sup>†</sup>`noheader`, `notable`, and `coeflegend` do not appear in the dialog box.

`varlist1`, ..., `varlistN` may contain factor variables; see [U] 11.4.3 **Factor variables**. You must have the same levels of factor variables in all equations that have factor variables.

`depvars` and the `varlists` may contain time-series operators; see [U] 11.4.4 **Time-series varlists**.

`bootstrap`, `by`, `jackknife`, `rolling`, and `statsby` are allowed; see [U] 11.1.10 **Prefix commands**.

Weights are not allowed with the `bootstrap` prefix.

`aweight`s are not allowed with the `jackknife` prefix.

`aweight`s and `fweight`s are allowed; see [U] 11.1.6 **weight**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

## Menu

Statistics > Linear models and related > Multiple-equation models > Seemingly unrelated regression

## Description

`sureg` fits seemingly unrelated regression models (Zellner 1962; Zellner and Huang 1962; Zellner 1963). The acronyms SURE and SUR are often used for the estimator.

## Options

### Model

`isure` specifies that `sureg` iterate over the estimated disturbance covariance matrix and parameter estimates until the parameter estimates converge. Under seemingly unrelated regression, this iteration converges to the maximum likelihood results. If this option is not specified, `sureg` produces two-step estimates.

`constraints`(*constraints*); see [R] **estimation options**.

### df adj.

`small` specifies that small-sample statistics be computed. It shifts the test statistics from chi-squared and *z* statistics to *F* statistics and *t* statistics. Although the standard errors from each equation are computed using the degrees of freedom for the equation, the degrees of freedom for the *t* statistics are all taken to be those for the first equation.

`dfk` specifies the use of an alternate divisor in computing the covariance matrix for the equation residuals. As an asymptotically justified estimator, `sureg` by default uses the number of sample observations (*n*) as a divisor. When the `dfk` option is set, a small-sample adjustment is made and the divisor is taken to be  $\sqrt{(n - k_i)(n - k_j)}$ , where *k<sub>i</sub>* and *k<sub>j</sub>* are the numbers of parameters in equations *i* and *j*, respectively.

`dfk2` specifies the use of an alternate divisor in computing the covariance matrix for the equation residuals. When the `dfk2` option is set, the divisor is taken to be the mean of the residual degrees of freedom from the individual equations.

### Reporting

`level`(#); see [R] **estimation options**.

`corr` displays the correlation matrix of the residuals between equations and performs a Breusch–Pagan test for independent equations; that is, the disturbance covariance matrix is diagonal.

`nocnsreport`; see [R] [estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

### Optimization

`optimization_options` control the iterative process that minimizes the sum of squared errors when `isure` is specified. These options are seldom used.

`iterate(#)` specifies the maximum number of iterations. When the number of iterations equals `#`, the optimizer stops and presents the current results, even if the convergence tolerance has not been reached. The default value of `iterate()` is the current value of `set maxiter`, which is `iterate(16000)` if `maxiter` has not been changed.

`trace` adds to the iteration log a display of the current parameter vector

`nolog` suppresses the display of the iteration log.

`tolerance(#)` specifies the tolerance for the coefficient vector. When the relative change in the coefficient vector from one iteration to the next is less than or equal to `#`, the optimization process is stopped. `tolerance(1e-6)` is the default.

The following options are available with `sureg` but are not shown in the dialog box:

`noheader` suppresses display of the table reporting  $F$  statistics,  $R$ -squared, and root mean squared error above the coefficient table.

`notable` suppresses display of the coefficient table.

`coeflegend`; see [R] [estimation options](#).

## Remarks

Seemingly unrelated regression models are so called because they appear to be joint estimates from several regression models, each with its own error term. The regressions are related because the (contemporaneous) errors associated with the dependent variables may be correlated. Chapter 5 of [Cameron and Trivedi \(2009\)](#) contains a discussion of the seemingly unrelated regression model and the feasible generalized least-squares estimator underlying it.

### ► Example 1

When we fit models with the same set of right-hand-side variables, the seemingly unrelated regression results (in terms of coefficients and standard errors) are the same as fitting the models separately (using, say, `regress`). The same is true when the models are nested. Even in such cases, `sureg` is useful when we want to perform joint tests. For instance, let us assume that we think

$$\begin{aligned}\text{price} &= \beta_0 + \beta_1 \text{foreign} + \beta_2 \text{length} + u_1 \\ \text{weight} &= \gamma_0 + \gamma_1 \text{foreign} + \gamma_2 \text{length} + u_2\end{aligned}$$

Because the models have the same set of explanatory variables, we could estimate the two equations separately. Yet, we might still choose to estimate them with `sureg` because we want to perform the joint test  $\beta_1 = \gamma_1 = 0$ .

We use the `small` and `dfk` options to obtain small-sample statistics comparable with `regress` or `mvreg`.

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. sureg (price foreign length) (weight foreign length), small dfk

Seemingly unrelated regression
```

| Equation | Obs | Parms | RMSE     | "R-sq" | F-Stat | P      |
|----------|-----|-------|----------|--------|--------|--------|
| price    | 74  | 2     | 2474.593 | 0.3154 | 16.35  | 0.0000 |
| weight   | 74  | 2     | 250.2515 | 0.8992 | 316.54 | 0.0000 |

|         | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|---------|-----------|-----------|-------|-------|----------------------|-----------|
| price   |           |           |       |       |                      |           |
| foreign | 2801.143  | 766.117   | 3.66  | 0.000 | 1286.674             | 4315.611  |
| length  | 90.21239  | 15.83368  | 5.70  | 0.000 | 58.91219             | 121.5126  |
| _cons   | -11621.35 | 3124.436  | -3.72 | 0.000 | -17797.77            | -5444.93  |
| weight  |           |           |       |       |                      |           |
| foreign | -133.6775 | 77.47615  | -1.73 | 0.087 | -286.8332            | 19.4782   |
| length  | 31.44455  | 1.601234  | 19.64 | 0.000 | 28.27921             | 34.60989  |
| _cons   | -2850.25  | 315.9691  | -9.02 | 0.000 | -3474.861            | -2225.639 |

These two equations have a common set of regressors, and we could have used a shorthand syntax to specify the equations:

```
. sureg (price weight = foreign length), small dfk
```

Here the results presented by `sureg` are the same as if we had estimated the equations separately:

```
. regress price foreign length
(output omitted)

. regress weight foreign length
(output omitted)
```

There is, however, a difference. We have allowed  $u_1$  and  $u_2$  to be correlated and have estimated the full variance–covariance matrix of the coefficients. `sureg` has estimated the correlations, but it does not report them unless we specify the `corr` option. We did not remember to specify `corr` when we fit the model, but we can redisplay the results:

```
. sureg, notable noheader corr
```

Correlation matrix of residuals:

```
 price weight
price 1.0000
weight 0.5840 1.0000
```

```
Breusch-Pagan test of independence: chi2(1) = 25.237, Pr = 0.0000
```

The `notable` and `noheader` options prevented `sureg` from redisplaying the header and coefficient tables. We find that, for the same cars, the correlation of the residuals in the `price` and `weight` equations is 0.5840 and that we can reject the hypothesis that this correlation is zero.

We can test that the coefficients on `foreign` are jointly zero in both equations—as we set out to do—by typing `test foreign`; see [\[R\] test](#). When we type a variable without specifying the equation, that variable is tested for zero in all equations in which it appears:

```
. test foreign
(1) [price]foreign = 0
(2) [weight]foreign = 0
 F(2, 142) = 17.99
 Prob > F = 0.0000
```



➤ Example 2

When the models do not have the same set of explanatory variables and are not nested, `sureg` may lead to more efficient estimates than running the models separately as well as allowing joint tests. This time, let us assume that we believe

$$\begin{aligned} \text{price} &= \beta_0 + \beta_1\text{foreign} + \beta_2\text{mpg} + \beta_3\text{displ} + u_1 \\ \text{weight} &= \gamma_0 + \gamma_1\text{foreign} + \gamma_2\text{length} + u_2 \end{aligned}$$

To fit this model, we type

```
. sureg (price foreign mpg displ) (weight foreign length), corr
Seemingly unrelated regression
```

| Equation | Obs | Parms | RMSE     | "R-sq" | chi2   | P      |
|----------|-----|-------|----------|--------|--------|--------|
| price    | 74  | 3     | 2165.321 | 0.4537 | 49.64  | 0.0000 |
| weight   | 74  | 2     | 245.2916 | 0.8990 | 661.84 | 0.0000 |

|              | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|--------------|-----------|-----------|-------|-------|----------------------|-----------|
| price        |           |           |       |       |                      |           |
| foreign      | 3058.25   | 685.7357  | 4.46  | 0.000 | 1714.233             | 4402.267  |
| mpg          | -104.9591 | 58.47209  | -1.80 | 0.073 | -219.5623            | 9.644042  |
| displacement | 18.18098  | 4.286372  | 4.24  | 0.000 | 9.779842             | 26.58211  |
| _cons        | 3904.336  | 1966.521  | 1.99  | 0.047 | 50.0263              | 7758.645  |
| weight       |           |           |       |       |                      |           |
| foreign      | -147.3481 | 75.44314  | -1.95 | 0.051 | -295.2139            | .517755   |
| length       | 30.94905  | 1.539895  | 20.10 | 0.000 | 27.93091             | 33.96718  |
| _cons        | -2753.064 | 303.9336  | -9.06 | 0.000 | -3348.763            | -2157.365 |

Correlation matrix of residuals:

```
 price weight
price 1.0000
weight 0.3285 1.0000
```

Breusch-Pagan test of independence: chi2(1) = 7.984, Pr = 0.0047

In comparison, if we had fit the price model separately,

```
. regress price foreign mpg displ
```

| Source   | SS        | df | MS         | Number of obs = 74     |  |  |
|----------|-----------|----|------------|------------------------|--|--|
| Model    | 294104790 | 3  | 98034929.9 | F( 3, 70) = 20.13      |  |  |
| Residual | 340960606 | 70 | 4870865.81 | Prob > F = 0.0000      |  |  |
|          |           |    |            | R-squared = 0.4631     |  |  |
|          |           |    |            | Adj R-squared = 0.4401 |  |  |
| Total    | 635065396 | 73 | 8699525.97 | Root MSE = 2207        |  |  |

| price        | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|--------------|-----------|-----------|-------|-------|----------------------|----------|
| foreign      | 3545.484  | 712.7763  | 4.97  | 0.000 | 2123.897             | 4967.072 |
| mpg          | -98.88559 | 63.17063  | -1.57 | 0.122 | -224.8754            | 27.10426 |
| displacement | 22.40416  | 4.634239  | 4.83  | 0.000 | 13.16146             | 31.64686 |
| _cons        | 2796.91   | 2137.873  | 1.31  | 0.195 | -1466.943            | 7060.763 |

The coefficients are slightly different, but the standard errors are uniformly larger. This would still be true if we specified the `dfk` option to make a small-sample adjustment to the estimated covariance of the disturbances.

◀

## □ Technical note

Constraints can be applied to SURE models using Stata's standard syntax for constraints. For a general discussion of constraints, see [R] [constraint](#); for examples similar to seemingly unrelated regression models, see [R] [reg3](#).

□

Arnold Zellner (1927– ) was born in New York. He studied physics at Harvard and economics at Berkeley, and then he taught economics at the Universities of Washington and Wisconsin before settling in Chicago in 1966. Among his many major contributions to econometrics and statistics are his work on seemingly unrelated regression, three-stage least squares, and Bayesian econometrics.

(Continued on next page)

## Saved results

sureg saves the following in `e()`:

### Scalars

|                           |                                                                       |
|---------------------------|-----------------------------------------------------------------------|
| <code>e(N)</code>         | number of observations                                                |
| <code>e(k)</code>         | number of parameters in system                                        |
| <code>e(k_eq)</code>      | number of equations                                                   |
| <code>e(k_autoCns)</code> | number of base, empty, and omitted constraints                        |
| <code>e(mss_#)</code>     | model sum of squares for equation #                                   |
| <code>e(df_m#)</code>     | model degrees of freedom for equation #                               |
| <code>e(rss_#)</code>     | residual sum of squares for equation #                                |
| <code>e(df_r)</code>      | residual degrees of freedom                                           |
| <code>e(r2_#)</code>      | <i>R</i> -squared for equation #                                      |
| <code>e(F_#)</code>       | <i>F</i> statistic for equation # ( <code>small</code> only)          |
| <code>e(rmse_#)</code>    | root mean squared error for equation #                                |
| <code>e(dfk2_adj)</code>  | divisor used with VCE when <code>dfk2</code> specified                |
| <code>e(ll)</code>        | log likelihood                                                        |
| <code>e(chi2_#)</code>    | $\chi^2$ for equation #                                               |
| <code>e(p_#)</code>       | significance for equation #                                           |
| <code>e(cons_#)</code>    | 1 if equation # has a constant, 0 otherwise                           |
| <code>e(k_eform)</code>   | number of leading equations appropriate for <code>eform</code> output |
| <code>e(chi2_bp)</code>   | Breusch–Pagan $\chi^2$                                                |
| <code>e(df_bp)</code>     | degrees of freedom for Breusch–Pagan $\chi^2$ test                    |
| <code>e(cons_#)</code>    | 1 when equation # has a constant; 0, otherwise                        |
| <code>e(rank)</code>      | rank of <code>e(V)</code>                                             |
| <code>e(ic)</code>        | number of iterations                                                  |

### Macros

|                              |                                                                 |
|------------------------------|-----------------------------------------------------------------|
| <code>e(cmd)</code>          | <code>sureg</code>                                              |
| <code>e(cmdline)</code>      | command as typed                                                |
| <code>e(method)</code>       | <code>sure</code> or <code>isure</code>                         |
| <code>e(depvar)</code>       | names of dependent variables                                    |
| <code>e(exog)</code>         | names of exogenous variables                                    |
| <code>e(eqnames)</code>      | names of equations                                              |
| <code>e(wtype)</code>        | weight type                                                     |
| <code>e(wexp)</code>         | weight expression                                               |
| <code>e(corr)</code>         | correlation structure                                           |
| <code>e(small)</code>        | <code>small</code>                                              |
| <code>e(dfk)</code>          | alternate divisor ( <code>dfk</code> or <code>dfk2</code> only) |
| <code>e(properties)</code>   | <code>b V</code>                                                |
| <code>e(predict)</code>      | program used to implement <code>predict</code>                  |
| <code>e(marginsok)</code>    | predictions allowed by <code>margins</code>                     |
| <code>e(marginsnotok)</code> | predictions disallowed by <code>margins</code>                  |
| <code>e(asbalanced)</code>   | factor variables <code>fvset</code> as <code>asbalanced</code>  |
| <code>e(asobserved)</code>   | factor variables <code>fvset</code> as <code>asobserved</code>  |

### Matrices

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>e(b)</code>     | coefficient vector                           |
| <code>e(Cns)</code>   | constraints matrix                           |
| <code>e(Sigma)</code> | $\hat{\Sigma}$ matrix                        |
| <code>e(V)</code>     | variance–covariance matrix of the estimators |

### Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`sureg` is implemented as an ado-file.

`sureg` uses the asymptotically efficient, feasible, generalized least-squares algorithm described in Greene (2008, 254–267). The computing formulas are given on page 256.

The  $R$ -squared reported is the percent of variance explained by the predictors. It may be used for descriptive purposes, but  $R$ -squared is not a well-defined concept when GLS is used.

`sureg` will refuse to compute the estimators if the same equation is named more than once or the covariance matrix of the residuals is singular.

The Breusch and Pagan (1980)  $\chi^2$  statistic—a Lagrange multiplier statistic—is given by

$$\lambda = T \sum_{m=1}^M \sum_{n=1}^{m-1} r_{mn}^2$$

where  $r_{mn}$  is the estimated correlation between the residuals of the  $M$  equations and  $T$  is the number of observations. It is distributed as  $\chi^2$  with  $M(M-1)/2$  degrees of freedom.

## References

- Breusch, T. S., and A. R. Pagan. 1980. The Lagrange multiplier test and its applications to model specification in econometrics. *Review of Economic Studies* 47: 239–253.
- Cameron, A. C., and P. K. Trivedi. 2009. *Microeconometrics Using Stata*. College Station, TX: Stata Press.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- McDowell, A. W. 2004. From the help desk: Seemingly unrelated regression with unbalanced equations. *Stata Journal* 4: 442–448.
- Weesie, J. 1999. `sg121`: Seemingly unrelated estimation and the cluster-adjusted sandwich estimator. *Stata Technical Bulletin* 52: 34–47. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 231–248. College Station, TX: Stata Press.
- Zellner, A. 1962. An efficient method of estimating seemingly unrelated regressions and tests for aggregation bias. *Journal of the American Statistical Association* 57: 348–368.
- . 1963. Estimators for seemingly unrelated regression equations: Some exact finite sample results. *Journal of the American Statistical Association* 58: 977–992.
- Zellner, A., and D. S. Huang. 1962. Further properties of efficient estimators for seemingly unrelated regression equations. *International Economic Review* 3: 300–313.

## Also see

- [R] [sureg postestimation](#) — Postestimation tools for `sureg`
- [R] [mvreg](#) — Multivariate regression
- [R] [nlsur](#) — Estimation of nonlinear systems of equations
- [R] [reg3](#) — Three-stage estimation for systems of simultaneous equations
- [R] [regress](#) — Linear regression
- [TS] [dfactor](#) — Dynamic-factor models
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `sureg`:

| command                | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>     | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estimates</code> | cataloging estimation results                                                                       |
| <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>margins</code>   | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>   | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code> | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details.

Syntax for predict

```
predict [type] newvar [if] [in] [, equation(eqno[,eqno]) statistic]
```

| statistic               | description                                                |
|-------------------------|------------------------------------------------------------|
| Main                    |                                                            |
| <code>xb</code>         | linear prediction; the default                             |
| <code>stdp</code>       | standard error of the linear prediction                    |
| <code>residuals</code>  | residuals                                                  |
| <code>difference</code> | difference between the linear predictions of two equations |
| <code>stddp</code>      | standard error of the difference in linear predictions     |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

Menu

Statistics > Postestimation > Predictions, residuals, etc.



## Options for predict

Main

`equation(eqno [, eqno])` specifies to which equation(s) you are referring.

`equation()` is filled in with one *eqno* for the `xb`, `stdp`, and `residuals` options. `equation(#1)` would mean that the calculation is to be made for the first equation, `equation(#2)` would mean the second, and so on. You could also refer to the equations by their names. `equation(income)` would refer to the equation named `income` and `equation(hours)` to the equation named `hours`.

If you do not specify `equation()`, the results are the same as if you specified `equation(#1)`.

`difference` and `stdp` refer to between-equation concepts. To use these options, you must specify two equations, for example, `equation(#1,#2)` or `equation(income,hours)`. When two equations must be specified, `equation()` is required.

`xb`, the default, calculates the linear prediction (fitted values)—the prediction of  $\mathbf{x}_j\mathbf{b}$  for the specified equation.

`stdp` calculates the standard error of the prediction for the specified equation. It can be thought of as the standard error of the predicted expected value or mean for the observation's covariate pattern. The standard error of the prediction is also referred to as the standard error of the fitted value.

`residuals` calculates the residuals.

`difference` calculates the difference between the linear predictions of two equations in the system.

With `equation(#1,#2)`, `difference` computes the prediction of `equation(#1)` minus the prediction of `equation(#2)`.

`stdp` is allowed only after you have previously fit a multiple-equation model. The standard error of the difference in linear predictions ( $\mathbf{x}_{1j}\mathbf{b} - \mathbf{x}_{2j}\mathbf{b}$ ) between equations 1 and 2 is calculated.

For more information on using `predict` after multiple-equation estimation commands, see [\[R\] predict](#).

## Remarks

For an example of cross-equation testing of parameters using the `test` command, see [example 1](#) in [\[R\] sureg](#).

### ► Example 1

In [example 1](#) of [\[R\] sureg](#), we fit a seemingly unrelated regressions model of `price` and `weight`. Here we obtain the fitted values.

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. sureg (price foreign length) (weight foreign length), small dfk
(output omitted)
. predict phat, equation(price)
(option xb assumed; fitted values)
. predict what, equation(weight)
(option xb assumed; fitted values)
```

```
. summarize price phat weight what
```

| Variable | Obs | Mean     | Std. Dev. | Min      | Max      |
|----------|-----|----------|-----------|----------|----------|
| price    | 74  | 6165.257 | 2949.496  | 3291     | 15906    |
| phat     | 74  | 6165.257 | 1656.407  | 1639.872 | 9398.138 |
| weight   | 74  | 3019.459 | 777.1936  | 1760     | 4840     |
| what     | 74  | 3019.459 | 736.9666  | 1481.199 | 4476.331 |

Just as in single-equation OLS regression, in a SURE model the sample mean of the fitted values for an equation equals the sample mean of the dependent variable.



➤ Example 2

Suppose that for whatever reason we were interested in the difference between the predicted values of price and weight. predict has an option to compute this difference in one step:

```
. predict diff, equation(price, weight) difference
```

diff is the same as phat - what:

```
. generate mydiff = phat - what
. summarize diff mydiff
```

| Variable | Obs | Mean     | Std. Dev. | Min       | Max      |
|----------|-----|----------|-----------|-----------|----------|
| diff     | 74  | 3145.797 | 1233.26   | -132.2275 | 5505.914 |
| mydiff   | 74  | 3145.797 | 1233.26   | -132.2275 | 5505.914 |



Methods and formulas

All postestimation commands listed above are implemented as ado-files.

Also see

[R] [sureg](#) — Zellner’s seemingly unrelated regression

[U] [20 Estimation and postestimation commands](#)

Syntax

*Shapiro–Wilk normality test*

```
swilk varlist [if] [in] [, options]
```

*Shapiro–Francia normality test*

```
sfrancia varlist [if] [in]
```

| options                           | description                                                |
|-----------------------------------|------------------------------------------------------------|
| Main                              |                                                            |
| <u>generate</u> ( <i>newvar</i> ) | create <i>newvar</i> containing <i>W</i> test coefficients |
| <u>lnnormal</u>                   | test for three-parameter lognormality                      |
| <u>noties</u>                     | do not use average ranks for tied values                   |

by is allowed with `swilk` and `sfrancia`; see [\[D\]](#) `by`.

Menu

swilk

Statistics > Summaries, tables, and tests > Distributional plots and tests > Shapiro-Wilk normality test

sfrancia

Statistics > Summaries, tables, and tests > Distributional plots and tests > Shapiro-Francia normality test

Description

`swilk` performs the Shapiro–Wilk *W* test for normality, and `sfrancia` performs the Shapiro–Francia *W'* test for normality. `swilk` can be used with  $4 \leq n \leq 2,000$  observations, and `sfrancia` can be used with  $5 \leq n \leq 5,000$  observations; see [\[R\]](#) `sktest` for a test allowing more observations.

Options

|                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Main                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>generate</code> ( <i>newvar</i> ) creates new variable <i>newvar</i> containing the <i>W</i> test coefficients.                                                                                                                                                                                                                                                                     |
| <code>lnnormal</code> specifies that the test be for three-parameter lognormality, meaning that $\ln(X - k)$ is tested for normality, where <i>k</i> is calculated from the data as the value that makes the skewness coefficient zero. When simply testing $\ln(X)$ for normality, do not specify this option. See <a href="#">[R]</a> <code>lnskew0</code> for estimation of <i>k</i> . |
| <code>noties</code> suppresses use of averaged ranks for tied values when calculating the <i>W</i> test coefficients.                                                                                                                                                                                                                                                                     |

Remarks

➤ Example 1

Using our automobile dataset, we will test whether the variables `mpg` and `trunk` are normally distributed:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. swilk mpg trunk
```

| Shapiro–Wilk W test for normal data |     |         |       |       |         |
|-------------------------------------|-----|---------|-------|-------|---------|
| Variable                            | Obs | W       | V     | z     | Prob>z  |
| mpg                                 | 74  | 0.94821 | 3.335 | 2.627 | 0.00430 |
| trunk                               | 74  | 0.97921 | 1.339 | 0.637 | 0.26215 |

```
. sfrancia mpg trunk
```

| Shapiro–Francia W' test for normal data |     |         |       |       |         |
|-----------------------------------------|-----|---------|-------|-------|---------|
| Variable                                | Obs | W'      | V'    | z     | Prob>z  |
| mpg                                     | 74  | 0.94872 | 3.629 | 2.490 | 0.00639 |
| trunk                                   | 74  | 0.98446 | 1.100 | 0.190 | 0.42477 |

We can reject the hypothesis that `mpg` is normally distributed, but we cannot reject that `trunk` is normally distributed.

The values reported under  $W$  and  $W'$  are the Shapiro–Wilk and Shapiro–Francia test statistics. The tests also report  $V$  and  $V'$ , which are more appealing indexes for departure from normality. The median values of  $V$  and  $V'$  are 1 for samples from normal populations. Large values indicate nonnormality. The 95% critical values of  $V$  ( $V'$ ), which depend on the sample size, are between 1.2 and 2.4 (2.0 and 2.8); see [Royston \(1991b\)](#). There is no more information in  $V$  ( $V'$ ) than in  $W$  ( $W'$ )—one is just the transform of the other.



➤ Example 2

We have data on a variable called `studytime`, which we suspect is distributed lognormally:

```
. use http://www.stata-press.com/data/r11/cancer
(Patient Survival in Drug Trial)

. generate lnstudytime = ln(studytime)

. swilk lnstudytime
```

| Shapiro–Wilk W test for normal data |     |         |       |       |         |
|-------------------------------------|-----|---------|-------|-------|---------|
| Variable                            | Obs | W       | V     | z     | Prob>z  |
| lnstudytime                         | 48  | 0.92731 | 3.311 | 2.547 | 0.00543 |

We can reject the lognormal assumption. We do *not* specify the `lnnormal` option when testing for lognormality. The `lnnormal` option is for three-parameter lognormality.



➤ Example 3

Having discovered that `ln(studytime)` is not distributed normally, we now test that `ln(studytime - k)` is normally distributed, where  $k$  is chosen so that the resulting skewness is zero. We obtain the estimate for  $k$  from `lnskew0`; see [R] [lnskew0](#):

```
. lnskew0 lnstudytimek = studytime, level(95)
```

| Transform      | k         | [95% Conf. Interval] |           | Skewness  |
|----------------|-----------|----------------------|-----------|-----------|
| ln(studytim-k) | -11.01181 | -infinity            | -.9477328 | -.0000173 |

```
. swilk lnstudytimek, lnnormal
```

Shapiro-Wilk W test for 3-parameter lognormal data

| Variable     | Obs | W       | V     | z     | Prob>z  |
|--------------|-----|---------|-------|-------|---------|
| lnstudytimek | 48  | 0.97064 | 1.337 | 1.261 | 0.10363 |

We cannot reject the hypothesis that `ln(studytime + 11.01181)` is distributed normally. We do specify the `lnnormal` option when using an estimated value of  $k$ .

Saved results

`swilk` and `sfrancia` save the following in `r()`:

| Scalars           |                        |                   |             |
|-------------------|------------------------|-------------------|-------------|
| <code>r(N)</code> | number of observations | <code>r(W)</code> | $W$ or $W'$ |
| <code>r(p)</code> | significance           | <code>r(V)</code> | $V$ or $V'$ |
| <code>r(z)</code> | $z$ statistic          |                   |             |

Methods and formulas

`swilk` and `sfrancia` are implemented as ado-files.

The Shapiro–Wilk test is based on [Shapiro and Wilk \(1965\)](#) with a new approximation accurate for  $4 \leq n \leq 2,000$  ([Royston 1992](#)). The calculations made by `swilk` are based on [Royston \(1982, 1992, 1993\)](#).

The Shapiro–Francia test ([Shapiro and Francia 1972](#); [Royston 1983](#)) is an approximate test that is similar to the Shapiro–Wilk test for very large samples.

Samuel Sanford Shapiro (1930– ) earned degrees in statistics and engineering from City College of New York, Columbia, and Rutgers. After employment in the U.S. army and industry, he joined the faculty at Florida International University in 1972. Shapiro has coauthored various texts in statistics and published several papers on distributional testing and other statistical topics.

## Acknowledgment

`swilk` and `sfrancia` were written by Patrick Royston of the MRC Clinical Trials Unit, London.

## References

- Gould, W. W. 1992. [sg3.7: Final summary of tests of normality](#). *Stata Technical Bulletin* 5: 10–11. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 114–115. College Station, TX: Stata Press.
- Royston, P. 1982. An extension of Shapiro and Wilks’s  $W$  test for normality to large samples. *Applied Statistics* 31: 115–124.
- . 1983. A simple method for evaluating the Shapiro–Francia  $W'$  test of non-normality. *Statistician* 32: 297–300.
- . 1991a. [sg3.2: Shapiro–Wilk and Shapiro–Francia tests](#). *Stata Technical Bulletin* 3: 19. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, p. 105. College Station, TX: Stata Press.
- . 1991b. Estimating departure from normality. *Statistics in Medicine* 10: 1283–1293.
- . 1992. Approximating the Shapiro–Wilk  $W$ -test for non-normality. *Statistics and Computing* 2: 117–119.
- . 1993. A toolkit for testing for non-normality in complete and censored samples. *Statistician* 42: 37–43.
- Shapiro, S. S., and R. S. Francia. 1972. An approximate analysis of variance test for normality. *Journal of the American Statistical Association* 67: 215–216.
- Shapiro, S. S., and M. B. Wilk. 1965. An analysis of variance test for normality (complete samples). *Biometrika* 52: 591–611.

## Also see

- [\[R\] `lnskew0`](#) — Find zero-skewness log or Box–Cox transform
- [\[R\] `lv`](#) — Letter-value displays
- [\[R\] `sktest`](#) — Skewness and kurtosis test for normality

Syntax

*Symmetry and marginal homogeneity tests*

```
symmetry casevar controlvar [if] [in] [weight] [, options]
```

*Immediate form of symmetry and marginal homogeneity tests*

```
symmi #11 #12 [...] \ #21 #22 [...] [\...] [if] [in] [, options]
```

| options        | description                                                      |
|----------------|------------------------------------------------------------------|
| Main           |                                                                  |
| <u>notable</u> | suppress output of contingency table                             |
| <u>contrib</u> | report contribution of each off-diagonal cell pair               |
| <u>exact</u>   | perform exact test of table symmetry                             |
| <u>mh</u>      | perform two marginal homogeneity tests                           |
| <u>trend</u>   | perform a test for linear trend in the (log) relative risk (RR)  |
| <u>cc</u>      | use continuity correction when calculating test for linear trend |

fweights are allowed; see [U] 11.1.6 weight.

Menu

symmetry

Statistics > Epidemiology and related > Other > Symmetry and marginal homogeneity test

symmi

Statistics > Epidemiology and related > Other > Symmetry and marginal homogeneity test calculator

Description

`symmetry` performs asymptotic symmetry and marginal homogeneity tests, as well as an exact symmetry test on  $K \times K$  tables where there is a 1-to-1 matching of cases and controls (nonindependence). This testing is used to analyze matched-pair case–control data with multiple discrete levels of the exposure (outcome) variable. In genetics, the test is known as the transmission/disequilibrium test (TDT) and is used to test the association between transmitted and nontransmitted parental marker alleles to an affected child (Spieldman, McGinnis, and Ewens 1993). For  $2 \times 2$  tables, the asymptotic test statistics reduce to the McNemar test statistic, and the exact symmetry test produces an exact McNemar test; see [ST] `epitab`. For many exposure variables, `symmetry` can optionally perform a test for linear trend in the log relative risk.

`symmetry` expects the data to be in the wide format; that is, each observation contains the matched case and control values in variables `casevar` and `controlvar`. Variables can be numeric or string.

`symmi` is the immediate form of `symmetry`. The `symmi` command uses the values specified on the command line; rows are separated by ‘\’, and options are the same as for `symmetry`. See [U] 19 Immediate commands for a general introduction to immediate commands.

Options

Main

`notable` suppresses the output of the contingency table. By default, `symmetry` displays the  $n \times n$  contingency table at the top of the output.

`contrib` reports the contribution of each off-diagonal cell pair to the overall symmetry  $\chi^2$ .

`exact` performs an exact test of table symmetry. This option is recommended for sparse tables.  
CAUTION: The exact test requires substantial amounts of time and computer memory for large tables.

`mh` performs two marginal homogeneity tests that do not require the inversion of the variance–covariance matrix.

By default, `symmetry` produces the Stuart–Maxwell test statistic, which requires the inversion of the nondiagonal variance–covariance matrix,  $V$ . When the table is sparse, the matrix may not be of full rank, and then the command substitutes a generalized inverse  $V^*$  for  $V^{-1}$ . `mh` calculates optional marginal homogeneity statistics that do not require the inversion of the variance–covariance matrix. These tests may be preferred in certain situations. See [Methods and formulas](#) and [Bickeböller and Clerget-Darpoux \(1995\)](#) for details on these test statistics.

`trend` performs a test for linear trend in the (log) relative risk (RR). This option is allowed only for numeric exposure (outcome) variables, and its use should be restricted to measurements on the ordinal or the interval scales.

`cc` specifies that the continuity correction be used when calculating the test for linear trend. This correction should be specified only when the levels of the exposure variable are equally spaced.

Remarks

`symmetry` and `symmi` may be used to analyze 1-to-1 matched case–control data with multiple discrete levels of the exposure (outcome) variable.

Example 1

Consider a survey of 344 individuals (BMDP 1990, 267–270) who were asked in October 1986 whether they agreed with President Reagan’s handling of foreign affairs. In January 1987, after the Iran-Contra affair became public, these same individuals were surveyed again and asked the same question. We would like to know if public opinion changed over this period.

We first describe the dataset and list a few observations.

```
. use http://www.stata-press.com/data/r11/iran
. describe
Contains data from http://www.stata-press.com/data/r11/iran.dta
 obs: 344
 vars: 2 29 Jan 2009 02:37
 size: 2,064 (99.9% of memory free)
```

| variable name | storage type | display format | value label | variable label           |
|---------------|--------------|----------------|-------------|--------------------------|
| before        | byte         | %8.0g          | vlab        | Public Opinion before IC |
| after         | byte         | %8.0g          | vlab        | Public Opinion after IC  |

Sorted by:



```
. list in 1/5
```

|    | before   | after    |
|----|----------|----------|
| 1. | agree    | agree    |
| 2. | agree    | disagree |
| 3. | agree    | unsure   |
| 4. | disagree | agree    |
| 5. | disagree | disagree |

Each observation corresponds to one of the 344 individuals. The data are in wide form so that each observation has a before and an after measurement. We now perform the test without options.

```
. symmetry before after
```

| Public<br>Opinion<br>before IC | Public Opinion after IC |          |        | Total |
|--------------------------------|-------------------------|----------|--------|-------|
|                                | agree                   | disagree | unsure |       |
| agree                          | 47                      | 56       | 38     | 141   |
| disagree                       | 28                      | 61       | 31     | 120   |
| unsure                         | 26                      | 47       | 10     | 83    |
| Total                          | 101                     | 164      | 79     | 344   |

|                                       | chi2  | df | Prob>chi2 |
|---------------------------------------|-------|----|-----------|
| Symmetry (asymptotic)                 | 14.87 | 3  | 0.0019    |
| Marginal homogeneity (Stuart-Maxwell) | 14.78 | 2  | 0.0006    |

The test first tabulates the data in a  $K \times K$  table and then performs Bowker's (1948) test for table symmetry and the Stuart–Maxwell (Stuart 1955; Maxwell 1970) test for marginal homogeneity.

Both the symmetry test and the marginal homogeneity test are highly significant, thus indicating a shift in public opinion.

An exact test of symmetry is provided for use on sparse tables. This test is computationally intensive, so it should not be used on large tables. Because we are working on a fast computer, we will run the symmetry test again and this time include the `exact` option. We will suppress the output of the contingency table by specifying `notable` and include the `contrib` option so that we may further examine the cells responsible for the significant result.

```
. symmetry before after, contrib exact mh notable
```

| Cells       | Contribution<br>to symmetry<br>chi-squared |  |  |  |
|-------------|--------------------------------------------|--|--|--|
| n1_2 & n2_1 | 9.3333                                     |  |  |  |
| n1_3 & n3_1 | 2.2500                                     |  |  |  |
| n2_3 & n3_2 | 3.2821                                     |  |  |  |

|                                       | chi2  | df | Prob>chi2 |
|---------------------------------------|-------|----|-----------|
| Symmetry (asymptotic)                 | 14.87 | 3  | 0.0019    |
| Marginal homogeneity (Stuart-Maxwell) | 14.78 | 2  | 0.0006    |
| Marginal homogeneity (Bickenboller)   | 13.53 | 2  | 0.0012    |
| Marginal homogeneity (no diagonals)   | 15.25 | 2  | 0.0005    |

|                                           |  |  |        |
|-------------------------------------------|--|--|--------|
| Symmetry (exact significance probability) |  |  | 0.0018 |
|-------------------------------------------|--|--|--------|

The largest contribution to the symmetry  $\chi^2$  is due to cells  $n_{12}$  and  $n_{21}$ . These correspond to changes between the agree and disagree categories. Of the 344 individuals, 56 (16.3%) changed from the agree to the disagree response, whereas only 28 (8.1%) changed in the opposite direction.

For these data, the results from the exact test are similar to those from the asymptotic test.



➤ Example 2

Breslow and Day (1980, 163) reprinted data from Mack et al. (1976) from a case–control study of the effect of exogenous estrogen on the risk of endometrial cancer. The data consist of 59 elderly women diagnosed with endometrial cancer and 59 disease-free control subjects living in the same community as the cases. Cases and controls were matched on age, marital status, and time living in the community. The data collected included information on the daily dose of conjugated estrogen therapy. Breslow and Day analyzed these data by creating four levels of the dose variable. Here are the data as entered into a Stata dataset:

```
. use http://www.stata-press.com/data/r11/bd163
. list, noobs divider
```

| case      | control   | count |
|-----------|-----------|-------|
| 0         | 0         | 6     |
| 0         | 0.1–0.299 | 2     |
| 0         | 0.3–0.625 | 3     |
| 0         | 0.626+    | 1     |
| 0.1–0.299 | 0         | 9     |
| 0.1–0.299 | 0.1–0.299 | 4     |
| 0.1–0.299 | 0.3–0.625 | 2     |
| 0.1–0.299 | 0.626+    | 1     |
| 0.3–0.625 | 0         | 9     |
| 0.3–0.625 | 0.1–0.299 | 2     |
| 0.3–0.625 | 0.3–0.625 | 3     |
| 0.3–0.625 | 0.626+    | 1     |
| 0.626+    | 0         | 12    |
| 0.626+    | 0.1–0.299 | 1     |
| 0.626+    | 0.3–0.625 | 2     |
| 0.626+    | 0.626+    | 1     |

This dataset is in a different format from that of the previous example. Instead of each observation representing one matched pair, each observation represents possibly multiple pairs indicated by the count variable. For instance, the first observation corresponds to six matched pairs where neither the case nor the control was on estrogen, the second observation corresponds to two matched pairs where the case was not on estrogen and the control was on 0.1 to 0.299 mg/day, etc.

To use `symmetry` to analyze this dataset, we must specify `fweight` to indicate that in our data there are observations corresponding to more than one matched pair.

```
. symmetry case control [fweight=count]
```

| case      | control |           |           |        | Total |
|-----------|---------|-----------|-----------|--------|-------|
|           | 0       | 0.1-0.299 | 0.3-0.625 | 0.626+ |       |
| 0         | 6       | 2         | 3         | 1      | 12    |
| 0.1-0.299 | 9       | 4         | 2         | 1      | 16    |
| 0.3-0.625 | 9       | 2         | 3         | 1      | 15    |
| 0.626+    | 12      | 1         | 2         | 1      | 16    |
| Total     | 36      | 9         | 10        | 4      | 59    |

|                                       | chi2  | df | Prob>chi2 |
|---------------------------------------|-------|----|-----------|
| Symmetry (asymptotic)                 | 17.10 | 6  | 0.0089    |
| Marginal homogeneity (Stuart-Maxwell) | 16.96 | 3  | 0.0007    |

Both the test of symmetry and the test of marginal homogeneity are highly significant, thus leading us to reject the null hypothesis that there is no effect of exposure to estrogen on the risk of endometrial cancer.

Breslow and Day perform a test for trend assuming that the estrogen exposure levels were equally spaced by recoding the exposure levels as 1, 2, 3, and 4.

We can easily reproduce their results by recoding our data in this way and by specifying the `trend` option. Two new numeric variables were created, `ca` and `co`, corresponding to the variables `case` and `control`, respectively. Below we list some of the data and our results from `symmetry`:

```
. encode case, gen(ca)
. encode control, gen(co)
. label values ca
. label values co
. list in 1/4
```

|    | case | control   | count | ca | co |
|----|------|-----------|-------|----|----|
| 1. | 0    | 0         | 6     | 1  | 1  |
| 2. | 0    | 0.1-0.299 | 2     | 1  | 2  |
| 3. | 0    | 0.3-0.625 | 3     | 1  | 3  |
| 4. | 0    | 0.626+    | 1     | 1  | 4  |

```
. symmetry ca co [fw=count], notable trend cc
```

|                                       | chi2  | df | Prob>chi2 |
|---------------------------------------|-------|----|-----------|
| Symmetry (asymptotic)                 | 17.10 | 6  | 0.0089    |
| Marginal homogeneity (Stuart-Maxwell) | 16.96 | 3  | 0.0007    |
| Linear trend in the (log) RR          | 14.43 | 1  | 0.0001    |

We requested the continuity correction by specifying `cc`. Doing so is appropriate because our coded exposure levels are equally spaced.

The test for trend was highly significant, indicating an increased risk of endometrial cancer with increased dosage of conjugated estrogen.

You must be cautious: the way in which you code the exposure variable affects the linear trend statistic. If instead of coding the levels as 1, 2, 3, and 4, we had instead used 0, 0.2, 0.46, and 0.7

(roughly the midpoint in the range of each level), we would have obtained a  $\chi^2$  statistic of 11.19 for these data.



## Saved results

`symmetry` saves the following in `r()`:

|                         |                                        |
|-------------------------|----------------------------------------|
| Scalars                 |                                        |
| <code>r(N_pair)</code>  | number of matched pairs                |
| <code>r(chi2)</code>    | asymptotic symmetry $\chi^2$           |
| <code>r(df)</code>      | asymptotic symmetry degrees of freedom |
| <code>r(p)</code>       | asymptotic symmetry $p$ -value         |
| <code>r(chi2_sm)</code> | MH (Stuart–Maxwell) $\chi^2$           |
| <code>r(df_sm)</code>   | MH (Stuart–Maxwell) degrees of freedom |
| <code>r(p_sm)</code>    | MH (Stuart–Maxwell) $p$ -value         |
| <code>r(chi2_b)</code>  | MH (Bickenböllner) $\chi^2$            |
| <code>r(df_b)</code>    | MH (Bickenböllner) degrees of freedom  |
| <code>r(p_b)</code>     | MH (Bickenböllner) $p$ -value          |
| <code>r(chi2_nd)</code> | MH (no diagonals) $\chi^2$             |
| <code>r(df_nd)</code>   | MH (no diagonals) degrees of freedom   |
| <code>r(p_nd)</code>    | MH (no diagonals) $p$ -value           |
| <code>r(chi2_t)</code>  | $\chi^2$ for linear trend              |
| <code>r(p_trend)</code> | $p$ -value for linear trend            |
| <code>r(p_exact)</code> | exact symmetry $p$ -value              |

## Methods and formulas

`symmetry` and `symmi` are implemented as ado-files.

Methods and formulas are presented under the following headings:

- [Asymptotic tests](#)
- [Exact symmetry test](#)

## Asymptotic tests

Consider a square table with  $K$  exposure categories, that is,  $K$  rows and  $K$  columns. Let  $n_{ij}$  be the count corresponding to row  $i$  and column  $j$  of the table,  $N_{ij} = n_{ij} + n_{ji}$ , for  $i, j = 1, 2, \dots, K$ , and  $n_{i.}$ , and let  $n_{.j}$  be the marginal totals for row  $i$  and column  $j$ , respectively. Asymptotic tests for symmetry and marginal homogeneity for this  $K \times K$  table are calculated as follows:

The null hypothesis of complete symmetry  $p_{ij} = p_{ji}$ ,  $i \neq j$ , is tested by calculating the test statistic (Bowker 1948)

$$T_{cs} = \sum_{i < j} \frac{(n_{ij} - n_{ji})^2}{n_{ij} + n_{ji}}$$

which is asymptotically distributed as  $\chi^2$  with  $K(K - 1)/2 - R$  degrees of freedom, where  $R$  is the number of off-diagonal cells with  $N_{ij} = 0$ .

The null hypothesis of marginal homogeneity,  $p_{i.} = p_{.i}$ , is tested by calculating the Stuart–Maxwell test statistic (Stuart 1955; Maxwell 1970),

$$T_{\text{sm}} = \mathbf{d}'\mathbf{V}^{-1}\mathbf{d}$$

where  $\mathbf{d}$  is a column vector with elements equal to the differences  $d_i = n_{i.} - n_{.i}$  for  $i = 1, 2, \dots, K$ , and  $\mathbf{V}$  is the variance–covariance matrix with elements

$$\begin{aligned} v_{ii} &= n_{i.} + n_{.i} - 2n_{ii} \\ v_{ij} &= -(n_{ij} + n_{ji}), \quad i \neq j \end{aligned}$$

$T_{\text{sm}}$  is asymptotically  $\chi^2$  with  $K - 1$  degrees of freedom.

This test statistic properly accounts for the dependence between the table's rows and columns. When the matrix  $\mathbf{V}$  is not of full rank, a generalized inverse  $\mathbf{V}^*$  is substituted for  $\mathbf{V}^{-1}$ .

The Bickelboller and Clerget-Darpoux (1995) marginal homogeneity test statistic is calculated by

$$T_{\text{mh}} = \sum_i \frac{(n_{i.} - n_{.i})^2}{n_{i.} + n_{.i}}$$

This statistic is asymptotically distributed, under the assumption of marginal independence, as  $\chi^2$  with  $K - 1$  degrees of freedom.

The marginal homogeneity (no diagonals) test statistic  $T_{\text{mh}}^0$  is calculated in the same way as  $T_{\text{mh}}$ , except that the diagonal elements do not enter into the calculation of the marginal totals. Unlike the previous test statistic,  $T_{\text{mh}}^0$  reduces to a McNemar test statistic for  $2 \times 2$  tables. The test statistic  $\{(K - 1)/2\}T_{\text{mh}}^0$  is asymptotically distributed as  $\chi^2$  with  $K - 1$  degrees of freedom (Cleves, Olson, and Jacobs 1997; Spielman and Ewens 1996).

Breslow and Day's test statistic for linear trend in the (log) of RR is

$$\frac{\left\{ \sum_{i < j} (n_{ij} - n_{ji})(X_j - X_i) - cc \right\}^2}{\sum_{i < j} (n_{ij} + n_{ji})(X_j - X_i)^2}$$

where the  $X_j$  are the doses associated with the various levels of exposure and  $cc$  is the continuity correction; it is asymptotically distributed as  $\chi^2$  with 1 degree of freedom.

The continuity correction option is applicable only when the levels of the exposure variable are equally spaced.

## Exact symmetry test

The exact test is based on a permutation algorithm applied to the null distribution. The distribution of the off-diagonal elements  $n_{ij}$ ,  $i \neq j$ , conditional on the sum of the complementary off-diagonal cells,  $N_{ij} = n_{ij} + n_{ji}$ , can be written as the product of  $K(K - 1)/2$  binomial random variables,

$$P(\mathbf{n}) = \prod_{i < j} \binom{N_{ij}}{n_{ij}} \pi_{ij}^{n_{ij}} (1 - \pi_{ij})^{n_{ji}}$$

where  $\mathbf{n}$  is a vector with elements  $n_{ij}$  and  $\pi_{ij} = E(n_{ij}/N_{ij}|N_{ij})$ . Under the null hypothesis of complete symmetry,  $\pi_{ij} = \pi_{ji} = 1/2$ , and thus the permutation distribution is given by

$$P_0(\mathbf{n}) = \prod_{i < j} \binom{N_{ij}}{n_{ij}} \left(\frac{1}{2}\right)^{N_{ij}}$$

The exact significance test is performed by evaluating

$$P_{cs} = \sum_{\mathbf{n} \in p} P_0(\mathbf{n})$$

where  $p = \{\mathbf{n} : P_0(\mathbf{n}) < P_0(\mathbf{n}^*)\}$  and  $\mathbf{n}^*$  is the observed contingency table data vector. The algorithm evaluates  $p_{cs}$  exactly. For information about permutation tests, see Good (2005, 2006).

## References

- Bickeböller, H., and F. Clerget-Darpoux. 1995. Statistical properties of the allelic and genotypic transmission/disequilibrium test for multiallelic markers. *Genetic Epidemiology* 12: 865–870.
- BMDP. 1990. *BMDP Statistical Software Manual*. Oakland, CA: University of California Press.
- Bowker, A. H. 1948. A test for symmetry in contingency tables. *Journal of the American Statistical Association* 43: 572–574.
- Breslow, N. E., and N. E. Day. 1980. *Statistical Methods in Cancer Research: Vol. 1—The Analysis of Case-Control Studies*. Lyon: IARC.
- Cleves, M. A. 1997. [sg74: Symmetry and marginal homogeneity test/Transmission-Disequilibrium Test \(TDT\)](#). *Stata Technical Bulletin* 40: 23–27. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 193–197. College Station, TX: Stata Press.
- . 1999. [sg110: Hardy–Weinberg equilibrium test and allele frequency estimation](#). *Stata Technical Bulletin* 48: 34–37. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 280–284. College Station, TX: Stata Press.
- Cleves, M. A., J. M. Olson, and K. B. Jacobs. 1997. Exact transmission-disequilibrium tests with multiallelic markers. *Genetic Epidemiology* 14: 337–347.
- Cui, J. 2000. [sg150: Hardy–Weinberg equilibrium test in case-control studies](#). *Stata Technical Bulletin* 57: 17–19. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 218–220. College Station, TX: Stata Press.
- Good, P. I. 2005. *Permutation, Parametric, and Bootstrap Tests of Hypotheses: A Practical Guide to Resampling Methods for Testing Hypotheses*. 3rd ed. New York: Springer.
- . 2006. *Resampling Methods: A Practical Guide to Data Analysis*. 3rd ed. Boston: Birkhäuser.
- Mack, T. M., M. C. Pike, B. E. Henderson, R. I. Pfeffer, V. R. Gerkins, M. Arthur, and S. E. Brown. 1976. Estrogens and endometrial cancer in a retirement community. *New England Journal of Medicine* 294: 1262–1267.
- Mander, A. 2000. [sbe38: Haplotype frequency estimation using an EM algorithm and log-linear modeling](#). *Stata Technical Bulletin* 57: 5–7. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 104–107. College Station, TX: Stata Press.
- Maxwell, A. E. 1970. Comparing the classification of subjects by two independent judges. *British Journal of Psychiatry* 116: 651–655.
- Spieldman, R. S., and W. J. Ewens. 1996. The TDT and other family-based tests for linkage disequilibrium and association. *American Journal of Human Genetics* 59: 983–989.
- Spieldman, R. S., R. E. McGinnis, and W. J. Ewens. 1993. Transmission test for linkage disequilibrium: The insulin gene region and insulin-dependence diabetes mellitus (IDDM). *American Journal of Human Genetics* 52: 506–516.
- Stuart, A. 1955. A test for homogeneity of the marginal distributions in a two-way classification. *Biometrika* 42: 412–416.

## Also see

[ST] [epitab](#) — Tables for epidemiologists

**table** — Tables of summary statistics

Syntax

`table rowvar [ colvar [ supercolvar ] ] [ if ] [ in ] [ weight ] [ , options ]`

| <i>options</i>                          | description                                                                                   |
|-----------------------------------------|-----------------------------------------------------------------------------------------------|
| Main                                    |                                                                                               |
| <code>contents(<i>clist</i>)</code>     | contents of table cells; select up to five statistics; default is <code>contents(freq)</code> |
| <code>by(<i>superrowvarlist</i>)</code> | superrow variables                                                                            |
| Options                                 |                                                                                               |
| <code>cellwidth(#)</code>               | cell width                                                                                    |
| <code>csepxwidth(#)</code>              | column-separation width                                                                       |
| <code>stubwidth(#)</code>               | stub width                                                                                    |
| <code>scsepxwidth(#)</code>             | supercolumn-separation width                                                                  |
| <code>center</code>                     | center-align table cells; default is right-align                                              |
| <code>left</code>                       | left-align table cells; default is right-align                                                |
| <code>cw</code>                         | perform casewise deletion                                                                     |
| <code>row</code>                        | add row totals                                                                                |
| <code>col</code>                        | add column totals                                                                             |
| <code>scol</code>                       | add supercolumn totals                                                                        |
| <code>concise</code>                    | suppress rows with all missing entries                                                        |
| <code>missing</code>                    | show missing statistics with period                                                           |
| <code>replace</code>                    | replace current data with table statistics                                                    |
| <code>name(<i>string</i>)</code>        | name new variables with prefix <i>string</i>                                                  |
| <code>format(<i>%fmt</i>)</code>        | display format for numbers in cells; default is <code>format(%9.0g)</code>                    |

`by` is allowed; see [D] [by](#).  
`fweights`, `awweights`, `iweights`, and `pweights` are allowed; see [U] [11.1.6 weight](#).  
`pweights` may not be used with `sd`, `semean`, `sebinomial`, or `sepoisson`. `iweights` may not be used with `semean`, `sebinomial`, or `sepoisson`. `awweights` may not be used with `sebinomial` or `sepoisson`.

where the elements of *clist* may be

|                                 |                                                                                      |                             |                          |
|---------------------------------|--------------------------------------------------------------------------------------|-----------------------------|--------------------------|
| <code>freq</code>               | frequency                                                                            | <code>n varname</code>      | same as count            |
| <code>mean varname</code>       | mean of <i>varname</i>                                                               | <code>max varname</code>    | maximum                  |
| <code>sd varname</code>         | standard deviation                                                                   | <code>min varname</code>    | minimum                  |
| <code>semean varname</code>     | standard error of the mean<br>( <code>sd/sqrt(n)</code> )                            | <code>median varname</code> | median                   |
| <code>sebinomial varname</code> | standard error of the mean, binomial<br>distribution ( <code>sqrt(p(1-p)/n)</code> ) | <code>p1 varname</code>     | 1st percentile           |
| <code>sepoisson varname</code>  | standard error of the mean, Poisson<br>distribution ( <code>sqrt(mean)</code> )      | <code>p2 varname</code>     | 2nd percentile           |
|                                 |                                                                                      | <code>...</code>            | 3rd–49th percentiles     |
|                                 |                                                                                      | <code>p50 varname</code>    | 50th percentile (median) |
|                                 |                                                                                      | <code>...</code>            | 51st–97th percentiles    |
| <code>sum varname</code>        | sum                                                                                  | <code>p98 varname</code>    | 98th percentile          |
| <code>rawsum varname</code>     | sums ignoring optionally specified weight                                            | <code>p99 varname</code>    | 99th percentile          |
| <code>count varname</code>      | count of nonmissing observations                                                     | <code>iqr varname</code>    | interquartile range      |



Rows, columns, supercolumns, and superrows are thus defined as

[illegible]

## Menu

Statistics > Summaries, tables, and tests > Tables > Table of summary statistics (table)

## Description

**table** calculates and displays tables of statistics.

## Options

[Main](#)

`contents(clist)` specifies the contents of the table's cells; if not specified, `contents(freq)` is used by default. `contents(freq)` produces a table of frequencies. `contents(mean mpg)` produces a table of the means of variable `mpg`. `contents(freq mean mpg sd mpg)` produces a table of frequencies together with the mean and standard deviation of variable `mpg`. Up to five statistics may be specified.

by(*superrowvarlist*) specifies that numeric or string variables be treated as superrows. Up to four variables may be specified in *superrowvarlist*. The `by()` option may be specified with the `by` prefix.

## Options

`cellwidth(#)` specifies the width of the cell in units of digit widths; 10 means the space occupied by 10 digits, which is 1234567890. The default `cellwidth()` is not a fixed number, but a number chosen by `table` to spread the table out while presenting a reasonable number of columns across the page.

`csepxwidth(#)` specifies the separation between columns in units of digit widths. The default is not a fixed number, but a number chosen by `table` according to what it thinks looks best.

`stubwidth(#)` specifies the width, in units of digit widths, to be allocated to the left stub of the table. The default is not a fixed number, but a number chosen by `table` according to what it thinks looks best.

`scsepxwidth(#)` specifies the separation between supercolumns in units of digit widths. The default is not a fixed number, but a number chosen by `table` to present the results best.

`center` specifies that results be centered in the table's cells. The default is to right-align results. For centering to work well, you typically need to specify a display format as well. `center format(%9.2f)` is popular.

`left` specifies that column labels be left-aligned. The default is to right-align column labels to distinguish them from supercolumn labels, which are left-aligned. If you specify `left`, both column and supercolumn labels are left-aligned.

`cw` specifies casewise deletion. If `cw` is not specified, all observations possible are used to calculate each of the specified statistics. `cw` is relevant only when you request a table containing statistics on multiple variables. For instance, `contents(mean mpg mean weight)` would produce a table reporting the means of variables `mpg` and `weight`. Consider an observation in which `mpg` is known but `weight` is missing. By default, that observation will be used in the calculation of the mean of `mpg`. If you specify `cw`, the observation will be excluded in the calculation of the means of both `mpg` and `weight`.

`row` specifies that a row be added to the table reflecting the total across the rows.

`col` specifies that a column be added to the table reflecting the total across columns.

`scol` specifies that a supercolumn be added to the table reflecting the total across supercolumns.

`concise` specifies that rows with all missing entries not be displayed.

`missing` specifies that missing statistics be shown in the table as periods (Stata's missing-value indicator). The default is that missing entries be left blank.

`replace` specifies that the data in memory be replaced with data containing 1 observation per cell (row, column, supercolumn, and superrow) and with variables containing the statistics designated in `contents()`.

This option is rarely specified. If you do not specify this option, the data in memory remain unchanged.

If you do specify this option, the first statistic will be named `table1`, the second `table2`, and so on. For instance, if `contents(mean mpg sd mpg)` was specified, the means of `mpg` would be in variable `table1` and the standard deviations in `table2`.

`name(string)` is relevant only if you specify `replace`. `name()` allows changing the default stub name that `replace` uses to name the new variables associated with the statistics. If you specify `name(stat)`, the first statistic will be placed in variable `stat1`, the second in `stat2`, and so on.

`format(%fmt)` specifies the display format for presenting numbers in the table's cells. `format(%9.0g)` is the default; `format(%9.2f)` and `format(%9.2fc)` are popular alternatives. The width of the format you specify does not matter, except that `%fmt` must be valid. The width of the cells is chosen by `table` to present the results best. The `cellwidth()` option allows you to override `table`'s choice.

## Limits

Up to four variables may be specified in the `by()`, so with the three row, column, and supercolumn variables, seven-way tables may be displayed.

Up to five statistics may be displayed in each cell of the table.

The sum of the number of rows, columns, supercolumns, and superrows is called the number of margins. A table may contain up to 3,000 margins. Thus a one-way table may contain 3,000 rows. A two-way table could contain 2,998 rows and two columns, 2,997 rows and three columns, ..., 1,500 rows and 1,500 columns, ..., two rows and 2,998 columns. A three-way table is similarly limited by the sum of the number of rows, columns, and supercolumns. A  $r \times c \times d$  table is feasible if  $r + c + d \leq 3,000$ . The limit is set in terms of the sum of the rows, columns, supercolumns, and superrows, and not, as you might expect, in terms of their product.

## Remarks

Remarks are presented under the following headings:

*One-way tables*

*Two-way tables*

*Three-way tables*

*Four-way and higher-dimensional tables*

## One-way tables

### ► Example 1

From the automobile dataset, here is a simple one-way table:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. table rep78, contents(mean mpg)
```

| Repair<br>Record<br>1978 | mean(mpg) |
|--------------------------|-----------|
| 1                        | 21        |
| 2                        | 19.125    |
| 3                        | 19.4333   |
| 4                        | 21.6667   |
| 5                        | 27.3636   |

We are not limited to including only one statistic:

```
. table rep78, c(n mpg mean mpg sd mpg median mpg)
```

| Repair<br>Record<br>1978 | N(mpg) | mean(mpg) | sd(mpg)  | med(mpg) |
|--------------------------|--------|-----------|----------|----------|
| 1                        | 2      | 21        | 4.24264  | 21       |
| 2                        | 8      | 19.125    | 3.758324 | 18       |
| 3                        | 30     | 19.4333   | 4.141325 | 19       |
| 4                        | 18     | 21.6667   | 4.93487  | 22.5     |
| 5                        | 11     | 27.3636   | 8.732385 | 30       |

We abbreviated `contents()` as `c()`. The `format()` option will allow us to better format the numbers in the table:

```
. table rep78, c(n mpg mean mpg sd mpg median mpg) format(%9.2f)
```

| Repair<br>Record<br>1978 |  | N(mpg) | mean(mpg) | sd(mpg) | med(mpg) |
|--------------------------|--|--------|-----------|---------|----------|
| 1                        |  | 2      | 21.00     | 4.24    | 21.00    |
| 2                        |  | 8      | 19.12     | 3.76    | 18.00    |
| 3                        |  | 30     | 19.43     | 4.14    | 19.00    |
| 4                        |  | 18     | 21.67     | 4.93    | 22.50    |
| 5                        |  | 11     | 27.36     | 8.73    | 30.00    |

The `center` option will center the results under the headings:

```
. table rep78, c(n mpg mean mpg sd mpg median mpg) format(%9.2f) center
```

| Repair<br>Record<br>1978 |  | N(mpg) | mean(mpg) | sd(mpg) | med(mpg) |
|--------------------------|--|--------|-----------|---------|----------|
| 1                        |  | 2      | 21.00     | 4.24    | 21.00    |
| 2                        |  | 8      | 19.12     | 3.76    | 18.00    |
| 3                        |  | 30     | 19.43     | 4.14    | 19.00    |
| 4                        |  | 18     | 21.67     | 4.93    | 22.50    |
| 5                        |  | 11     | 27.36     | 8.73    | 30.00    |



Two-way tables

Example 2

In example 1, when we typed `'table rep78, ...'`, we obtained a one-way table. If we were to type `'table rep78 foreign, ...'`, we would obtain a two-way table:

```
. table rep78 foreign, c(mean mpg)
```

| Repair<br>Record<br>1978 | Car type |         |
|--------------------------|----------|---------|
|                          | Domestic | Foreign |
| 1                        | 21       |         |
| 2                        | 19.125   |         |
| 3                        | 19       | 23.3333 |
| 4                        | 18.4444  | 24.8889 |
| 5                        | 32       | 26.3333 |

Note the missing cells. Certain combinations of repair record and car type do not exist in our dataset.

As with one-way tables, we can specify a display format for the cells and center the numbers within the cells if we wish.

```
. table rep78 foreign, c(mean mpg) format(%9.2f) center
```

| Repair<br>Record<br>1978 | Car type |         |
|--------------------------|----------|---------|
|                          | Domestic | Foreign |
| 1                        | 21.00    |         |
| 2                        | 19.12    |         |
| 3                        | 19.00    | 23.33   |
| 4                        | 18.44    | 24.89   |
| 5                        | 32.00    | 26.33   |

We can obtain row totals by specifying the `row` option and obtain column totals by specifying the `col` option. We specify both below:

```
. table rep78 foreign, c(mean mpg) format(%9.2f) center row col
```

| Repair<br>Record<br>1978 | Car type |         | Total |
|--------------------------|----------|---------|-------|
|                          | Domestic | Foreign |       |
| 1                        | 21.00    |         | 21.00 |
| 2                        | 19.12    |         | 19.12 |
| 3                        | 19.00    | 23.33   | 19.43 |
| 4                        | 18.44    | 24.89   | 21.67 |
| 5                        | 32.00    | 26.33   | 27.36 |
| Total                    | 19.54    | 25.29   | 21.29 |

`table` can display multiple statistics within cells, but once we move beyond one-way tables, the table becomes busy:

```
. table foreign rep78, c(mean mpg n mpg) format(%9.2f) center
```

| Car type | Repair Record 1978 |            |             |            |            |
|----------|--------------------|------------|-------------|------------|------------|
|          | 1                  | 2          | 3           | 4          | 5          |
| Domestic | 21.00<br>2         | 19.12<br>8 | 19.00<br>27 | 18.44<br>9 | 32.00<br>2 |
| Foreign  |                    |            | 23.33<br>3  | 24.89<br>9 | 26.33<br>9 |

This two-way table with two statistics per cell works well here. That was, in part, helped along by our interchanging the rows and columns. We turned the table around by typing `table foreign rep78` rather than `table rep78 foreign`.

(Continued on next page)

Another way to display two-way tables is to specify a row and superrow rather than a row and column. We do that below and display three statistics per cell:

```
. table foreign, by(rep78) c(mean mpg sd mpg n mpg) format(%9.2f) center
```

| Repair<br>Record<br>1978 and<br>Car type | mean(mpg)      | sd(mpg)      | N(mpg)  |
|------------------------------------------|----------------|--------------|---------|
| 1<br>Domestic<br>Foreign                 | 21.00          | 4.24         | 2       |
| 2<br>Domestic<br>Foreign                 | 19.12          | 3.76         | 8       |
| 3<br>Domestic<br>Foreign                 | 19.00<br>23.33 | 4.09<br>2.52 | 27<br>3 |
| 4<br>Domestic<br>Foreign                 | 18.44<br>24.89 | 4.59<br>2.71 | 9<br>9  |
| 5<br>Domestic<br>Foreign                 | 32.00<br>26.33 | 2.83<br>9.37 | 2<br>9  |



Three-way tables

Example 3

We have data on the prevalence of byssinosis, a form of pneumoconiosis to which workers exposed to cotton dust are susceptible. The dataset is on 5,419 workers in a large cotton mill. We know whether each worker smokes, his or her race, and the dustiness of the work area. The categorical variables are

- smokes      Smoker or nonsmoker in the last five years.
- race        White or other.
- workplace   1 (most dusty), 2 (less dusty), 3 (least dusty).

Moreover, this dataset includes a frequency-weight variable pop. Here is a three-way table showing the fraction of workers with byssinosis:

```
. use http://www.stata-press.com/data/r11/byssin
(Byssinosis incidence)
. table workplace smokes race [fw=pop], c(mean prob)
```

| Dustiness<br>of<br>workplace | Race and Smokes |              |             |              |
|------------------------------|-----------------|--------------|-------------|--------------|
|                              | other<br>no     | other<br>yes | white<br>no | white<br>yes |
| least                        | .0107527        | .0101523     | .0081549    | .0162774     |
| less                         | .02             | .0081633     | .0136612    | .0143149     |
| most                         | .0820896        | .1679105     | .0833333    | .2295082     |

This table would look better if we showed the fraction to four digits:

```
. table workplace smokes race [fw=pop], c(mean prob) format(%9.4f)
```

| Dustiness<br>of<br>workplace | Race and Smokes |        |        |        |
|------------------------------|-----------------|--------|--------|--------|
|                              | other           |        | white  |        |
|                              | no              | yes    | no     | yes    |
| least                        | 0.0108          | 0.0102 | 0.0082 | 0.0163 |
| less                         | 0.0200          | 0.0082 | 0.0137 | 0.0143 |
| most                         | 0.0821          | 0.1679 | 0.0833 | 0.2295 |

In this table, the rows are the dustiness of the workplace, the columns are whether the worker smokes, and the supercolumns are the worker's race.

Now we request that the table include the supercolumn totals by specifying the `sctotal` option, which we can abbreviate as `sc`:

```
. table workplace smokes race [fw=pop], c(mean prob) format(%9.4f) sc
```

| Dustiness<br>of<br>workplace | Race and Smokes |        |        |        | Total  |        |
|------------------------------|-----------------|--------|--------|--------|--------|--------|
|                              | other           |        | white  |        |        |        |
|                              | no              | yes    | no     | yes    | no     | yes    |
| least                        | 0.0108          | 0.0102 | 0.0082 | 0.0163 | 0.0090 | 0.0145 |
| less                         | 0.0200          | 0.0082 | 0.0137 | 0.0143 | 0.0159 | 0.0123 |
| most                         | 0.0821          | 0.1679 | 0.0833 | 0.2295 | 0.0826 | 0.1929 |

The supercolumn total is the total over race and is divided into its columns based on smokes. Here is the table with the column rather than the supercolumn totals:

```
. table workplace smokes race [fw=pop], c(mean prob) format(%9.4f) col
```

| Dustiness<br>of<br>workplace | Race and Smokes |        |        |        |        |        |
|------------------------------|-----------------|--------|--------|--------|--------|--------|
|                              | other           |        | white  |        |        |        |
|                              | no              | yes    | Total  | no     | yes    | Total  |
| least                        | 0.0108          | 0.0102 | 0.0104 | 0.0082 | 0.0163 | 0.0129 |
| less                         | 0.0200          | 0.0082 | 0.0135 | 0.0137 | 0.0143 | 0.0140 |
| most                         | 0.0821          | 0.1679 | 0.1393 | 0.0833 | 0.2295 | 0.1835 |

Here is the table with both column and supercolumn totals:

```
. table workplace smokes race [fw=pop], c(mean prob) format(%9.4f) sc col
```

| Dustin<br>ess of<br>workpl<br>ace | Race and Smokes |        |        |        |        |        |        |        |        |
|-----------------------------------|-----------------|--------|--------|--------|--------|--------|--------|--------|--------|
|                                   | other           |        |        | white  |        |        | Total  |        |        |
|                                   | no              | yes    | Total  | no     | yes    | Total  | no     | yes    | Total  |
| least                             | 0.0108          | 0.0102 | 0.0104 | 0.0082 | 0.0163 | 0.0129 | 0.0090 | 0.0145 | 0.0122 |
| less                              | 0.0200          | 0.0082 | 0.0135 | 0.0137 | 0.0143 | 0.0140 | 0.0159 | 0.0123 | 0.0138 |
| most                              | 0.0821          | 0.1679 | 0.1393 | 0.0833 | 0.2295 | 0.1835 | 0.0826 | 0.1929 | 0.1570 |

table is struggling to keep this table from becoming too wide—notice how it divided the words in the title in the top-left stub. Here, if the table had more columns, or, if we demanded more digits, table would be forced to segment the table and present it in pieces, which it would do:

```
. table workplace smokes race [fw=pop], c(mean prob) format(%9.6f) sc col
```

| Dustiness<br>of<br>workplace | Race and Smokes |          |          |          |          |          |
|------------------------------|-----------------|----------|----------|----------|----------|----------|
|                              | other           |          |          | white    |          |          |
|                              | no              | yes      | Total    | no       | yes      | Total    |
| least                        | 0.010753        | 0.010152 | 0.010417 | 0.008155 | 0.016277 | 0.012949 |
| less                         | 0.020000        | 0.008163 | 0.013483 | 0.013661 | 0.014315 | 0.014035 |
| most                         | 0.082090        | 0.167910 | 0.139303 | 0.083333 | 0.229508 | 0.183521 |

| Dustiness<br>of<br>workplace | Race and Smokes |          |          |
|------------------------------|-----------------|----------|----------|
|                              | Total           |          |          |
|                              | no              | yes      | Total    |
| least                        | 0.008990        | 0.014471 | 0.012174 |
| less                         | 0.015901        | 0.012262 | 0.013846 |
| most                         | 0.082569        | 0.192905 | 0.156951 |

Here three digits is probably enough, so here is the table including all the row, column, and supercolumn totals:

```
. table workplace smokes race [fw=pop], c(mean prob) format(%9.3f)
> sc col row
```

| Dustiness<br>of<br>workplace | Race and Smokes |       |       |       |       |       |       |       |       |
|------------------------------|-----------------|-------|-------|-------|-------|-------|-------|-------|-------|
|                              | other           |       |       | white |       |       | Total |       |       |
|                              | no              | yes   | Total | no    | yes   | Total | no    | yes   | Total |
| least                        | 0.011           | 0.010 | 0.010 | 0.008 | 0.016 | 0.013 | 0.009 | 0.014 | 0.012 |
| less                         | 0.020           | 0.008 | 0.013 | 0.014 | 0.014 | 0.014 | 0.016 | 0.012 | 0.014 |
| most                         | 0.082           | 0.168 | 0.139 | 0.083 | 0.230 | 0.184 | 0.083 | 0.193 | 0.157 |
| Total                        | 0.025           | 0.048 | 0.038 | 0.014 | 0.035 | 0.026 | 0.018 | 0.039 | 0.030 |

We can show multiple statistics:

```
. table workplace smokes race [fw=pop], c(mean prob n prob) format(%9.3f)
> sc col row
```

| Dustiness<br>of<br>workplace | Race and Smokes |                |                |                |                |                |                |                |                |
|------------------------------|-----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
|                              | other           |                |                | white          |                |                | Total          |                |                |
|                              | no              | yes            | Total          | no             | yes            | Total          | no             | yes            | Total          |
| least                        | 0.011<br>465    | 0.010<br>591   | 0.010<br>1,056 | 0.008<br>981   | 0.016<br>1,413 | 0.013<br>2,394 | 0.009<br>1,446 | 0.014<br>2,004 | 0.012<br>3,450 |
| less                         | 0.020<br>200    | 0.008<br>245   | 0.013<br>445   | 0.014<br>366   | 0.014<br>489   | 0.014<br>855   | 0.016<br>566   | 0.012<br>734   | 0.014<br>1,300 |
| most                         | 0.082<br>134    | 0.168<br>268   | 0.139<br>402   | 0.083<br>84    | 0.230<br>183   | 0.184<br>267   | 0.083<br>218   | 0.193<br>451   | 0.157<br>669   |
| Total                        | 0.025<br>799    | 0.048<br>1,104 | 0.038<br>1,903 | 0.014<br>1,431 | 0.035<br>2,085 | 0.026<br>3,516 | 0.018<br>2,230 | 0.039<br>3,189 | 0.030<br>5,419 |



## Four-way and higher-dimensional tables

### ► Example 4

Let's pretend that our byssinosis dataset also recorded each worker's sex (it does not, and we have made up this extra information). We obtain a four-way table just as we would a three-way table, but we specify the fourth variable as a superrow by including it in the `by()` option:

```
. use http://www.stata-press.com/data/r11/byssin1
(Byssinosis incidence)
. table workplace smokes race [fw=pop], by(sex) c(mean prob) format(%9.3f)
> sc col row
```

| Sex and<br>Dustiness<br>of<br>workplace | Race and Smokes |       |       |             |       |       |             |       |       |
|-----------------------------------------|-----------------|-------|-------|-------------|-------|-------|-------------|-------|-------|
|                                         | other<br>no     | yes   | Total | white<br>no | yes   | Total | Total<br>no | yes   | Total |
| Female                                  |                 |       |       |             |       |       |             |       |       |
| least                                   | 0.006           | 0.009 | 0.008 | 0.009       | 0.021 | 0.016 | 0.009       | 0.018 | 0.014 |
| less                                    | 0.020           | 0.008 | 0.010 | 0.015       | 0.015 | 0.015 | 0.016       | 0.012 | 0.014 |
| most                                    | 0.057           | 0.154 | 0.141 |             |       |       | 0.057       | 0.154 | 0.141 |
| Total                                   | 0.017           | 0.051 | 0.043 | 0.011       | 0.020 | 0.016 | 0.012       | 0.032 | 0.024 |
| Male                                    |                 |       |       |             |       |       |             |       |       |
| least                                   | 0.013           | 0.011 | 0.012 | 0.006       | 0.007 | 0.006 | 0.009       | 0.008 | 0.009 |
| less                                    | 0.020           | 0.000 | 0.019 | 0.000       | 0.013 | 0.011 | 0.016       | 0.013 | 0.014 |
| most                                    | 0.091           | 0.244 | 0.136 | 0.083       | 0.230 | 0.184 | 0.087       | 0.232 | 0.167 |
| Total                                   | 0.029           | 0.041 | 0.033 | 0.020       | 0.056 | 0.043 | 0.025       | 0.052 | 0.039 |

If our dataset also included work group and we wanted a five-way table, we could include both the sex and work-group variables in the `by()` option. You may include up to four variables in `by()`, and so produce up to 7-way tables.



## Methods and formulas

`table` is implemented as an ado-file. The contents of cells are calculated by `collapse` and are displayed by `tabdisp`; see [\[D\] collapse](#) and [\[P\] tabdisp](#).

## Also see

- [\[R\] summarize](#) — Summary statistics
- [\[R\] tabstat](#) — Display table of summary statistics
- [\[R\] tabulate oneway](#) — One-way tables of frequencies
- [\[R\] tabulate twoway](#) — Two-way tables of frequencies
- [\[D\] collapse](#) — Make dataset of summary statistics
- [\[P\] tabdisp](#) — Display tables

Syntax

```
tabstat varlist [if] [in] [weight] [, options]
```

| options                                                                  | description                                               |
|--------------------------------------------------------------------------|-----------------------------------------------------------|
| Main                                                                     |                                                           |
| by( <i>varname</i> )                                                     | group statistics by variable                              |
| statistics( <i>statname</i> [...])                                       | report specified statistics                               |
| Options                                                                  |                                                           |
| labelwidth(#)                                                            | width for by() variable labels; default is labelwidth(16) |
| varwidth(#)                                                              | variable width; default is varwidth(12)                   |
| columns(variables)                                                       | display variables in table columns; the default           |
| columns(statistics)                                                      | display statistics in table columns                       |
| format[ <i>(%fmt)</i> ]                                                  | display format for statistics; default format is %9.0g    |
| casewise                                                                 | perform casewise deletion of observations                 |
| nototal                                                                  | do not report overall statistics; use with by()           |
| missing                                                                  | report statistics for missing values of by() variable     |
| noseparator                                                              | do not use separator line between by() categories         |
| longstub                                                                 | make left table stub wider                                |
| save                                                                     | save summary statistics in r()                            |
| by is allowed; see <a href="#">[D] by</a> .                              |                                                           |
| aweight and fweight are allowed; see <a href="#">[U] 11.1.6 weight</a> . |                                                           |

Menu

Statistics > Summaries, tables, and tests > Tables > Table of summary statistics (tabstat)

Description

tabstat displays summary statistics for a series of numeric variables in one table, possibly broken down on (conditioned by) another variable.

Without the by() option, tabstat is a useful alternative to summarize because it allows you to specify the list of statistics to be displayed.

With the by() option, tabstat resembles tabulate used with its summarize() option in that both report statistics of varlist for the different values of varname. tabstat allows more flexibility in terms of the statistics presented and the format of the table.

tabstat is sensitive to the linesize; it widens the table if possible and wraps if necessary.

# Options

## Main

`by(varname)` specifies that the statistics be displayed separately for each unique value of *varname*; *varname* may be numeric or string. For instance, `tabstat height` would present the overall mean of height. `tabstat height, by(sex)` would present the mean height of males, and of females, and the overall mean height. Do not confuse the `by()` option with the `by` prefix; both may be specified.

`statistics(statname [...])` specifies the statistics to be displayed; the default is equivalent to specifying `statistics(mean)`. (`stats()` is a synonym for `statistics()`.) Multiple statistics may be specified and are separated by white space, such as `statistics(mean sd)`. Available statistics are

| <i>statname</i> | definition                               | <i>statname</i> | definition                           |
|-----------------|------------------------------------------|-----------------|--------------------------------------|
| <u>mean</u>     | mean                                     | p1              | 1st percentile                       |
| <u>count</u>    | count of nonmissing observations         | p5              | 5th percentile                       |
| <u>n</u>        | same as count                            | p10             | 10th percentile                      |
| <u>sum</u>      | sum                                      | p25             | 25th percentile                      |
| <u>max</u>      | maximum                                  | <u>median</u>   | median (same as p50)                 |
| <u>min</u>      | minimum                                  | p50             | 50th percentile (same as median)     |
| <u>range</u>    | range = max – min                        | p75             | 75th percentile                      |
| <u>sd</u>       | standard deviation                       | p90             | 90th percentile                      |
| <u>variance</u> | variance                                 | p95             | 95th percentile                      |
| <u>cv</u>       | coefficient of variation ( $sd/mean$ )   | p99             | 99th percentile                      |
| <u>semean</u>   | standard error of mean ( $sd/\sqrt{n}$ ) | iqr             | interquartile range = p75 – p25      |
| <u>skewness</u> | skewness                                 | q               | equivalent to specifying p25 p50 p75 |
| <u>kurtosis</u> | kurtosis                                 |                 |                                      |

## Options

`labelwidth(#)` specifies the maximum width to be used within the stub to display the labels of the `by()` variable. The default is `labelwidth(16)`.  $8 \leq \# \leq 32$ .

`varwidth(#)` specifies the maximum width to be used within the stub to display the names of the variables. The default is `varwidth(12)`. `varwidth()` is effective only with `columns(statistics)`. Setting `varwidth()` implies `longstub`.  $8 \leq \# \leq 16$ .

`columns(variables | statistics)` specifies whether to display variables or statistics in the columns of the table. `columns(variables)` is the default when more than one variable is specified.

`format` and `format(%fmt)` specify how the statistics are to be formatted. The default is to use a `%9.0g` format.

`format` specifies that each variable's statistics be formatted with the variable's display format; see [\[D\] format](#).

`format(%fmt)` specifies the format to be used for all statistics. The maximum width of the specified format should not exceed nine characters.

`casewise` specifies casewise deletion of observations. Statistics are to be computed for the sample that is not missing for any of the variables in *varlist*. The default is to use all the nonmissing values for each variable.

`nototal` is for use with `by()`; it specifies that the overall statistics not be reported.

`missing` specifies that missing values of the `by()` variable be treated just like any other value and that statistics should be displayed for them. The default is not to report the statistics for the `by()==missing` group. If the `by()` variable is a string variable, `by()=="` is considered to mean missing.

`noseparator` specifies that a separator line between the `by()` categories not be displayed.

`longstub` specifies that the left stub of the table be made wider so that it can include names of the statistics or variables in addition to the categories of `by(varname)`. The default is to describe the statistics or variables in a header. `longstub` is ignored if `by(varname)` is not specified.

`save` specifies that the summary statistics be returned in `r()`. The overall (unconditional) statistics are returned in matrix `r(StatTotal)` (rows are statistics, columns are variables). The conditional statistics are returned in the matrices `r(Stat1)`, `r(Stat2)`, ..., and the names of the corresponding variables are returned in the macros `r(name1)`, `r(name2)`, ...

## Remarks

This command is probably most easily understood by going through a series of examples.

### ► Example 1

We have data on the price, weight, mileage rating, and repair record of 22 foreign and 52 domestic 1978 automobiles. We want to summarize these variables for the different origins of the automobiles.

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. tabstat price weight mpg rep78, by(foreign)

Summary statistics: mean
 by categories of: foreign (Car type)
```

| foreign  | price    | weight   | mpg      | rep78    |
|----------|----------|----------|----------|----------|
| Domestic | 6072.423 | 3317.115 | 19.82692 | 3.020833 |
| Foreign  | 6384.682 | 2315.909 | 24.77273 | 4.285714 |
| Total    | 6165.257 | 3019.459 | 21.2973  | 3.405797 |

More summary statistics can be requested via the `statistics()` option. The group totals can be suppressed with the `nototal` option.

```
. tabstat price weight mpg rep78, by(foreign) stat(mean sd min max) nototal

Summary statistics: mean, sd, min, max
 by categories of: foreign (Car type)
```

| foreign  | price    | weight   | mpg      | rep78    |
|----------|----------|----------|----------|----------|
| Domestic | 6072.423 | 3317.115 | 19.82692 | 3.020833 |
|          | 3097.104 | 695.3637 | 4.743297 | .837666  |
|          | 3291     | 1800     | 12       | 1        |
|          | 15906    | 4840     | 34       | 5        |
| Foreign  | 6384.682 | 2315.909 | 24.77273 | 4.285714 |
|          | 2621.915 | 433.0035 | 6.611187 | .7171372 |
|          | 3748     | 1760     | 14       | 3        |
|          | 12990    | 3420     | 41       | 5        |

Although the header of the table describes the statistics running vertically in the “cells”, the table may become hard to read, especially with many variables or statistics. The `longstub` option specifies that a column be added describing the contents of the cells. The `format` option can be issued to specify that `tabstat` display the statistics by using the display format of the variables rather than the overall default `%9.0g`.

```
. tabstat price weight mpg rep78, by(foreign) stat(mean sd min max) long format
```

| foreign  | stats | price   | weight  | mpg     | rep78   |
|----------|-------|---------|---------|---------|---------|
| Domestic | mean  | 6,072.4 | 3,317.1 | 19.8269 | 3.02083 |
|          | sd    | 3,097.1 | 695.364 | 4.7433  | .837666 |
|          | min   | 3,291   | 1,800   | 12      | 1       |
|          | max   | 15,906  | 4,840   | 34      | 5       |
| Foreign  | mean  | 6,384.7 | 2,315.9 | 24.7727 | 4.28571 |
|          | sd    | 2,621.9 | 433.003 | 6.61119 | .717137 |
|          | min   | 3,748   | 1,760   | 14      | 3       |
|          | max   | 12,990  | 3,420   | 41      | 5       |
| Total    | mean  | 6,165.3 | 3,019.5 | 21.2973 | 3.4058  |
|          | sd    | 2,949.5 | 777.194 | 5.7855  | .989932 |
|          | min   | 3,291   | 1,760   | 12      | 1       |
|          | max   | 15,906  | 4,840   | 41      | 5       |

We can specify a layout of the table in which the statistics run horizontally and the variables run vertically by specifying the `col(statistics)` option.

```
. tabstat price weight mpg rep78, by(foreign) stat(min mean max) col(stat) long
```

| foreign  | variable | min  | mean     | max   |
|----------|----------|------|----------|-------|
| Domestic | price    | 3291 | 6072.423 | 15906 |
|          | weight   | 1800 | 3317.115 | 4840  |
|          | mpg      | 12   | 19.82692 | 34    |
|          | rep78    | 1    | 3.020833 | 5     |
| Foreign  | price    | 3748 | 6384.682 | 12990 |
|          | weight   | 1760 | 2315.909 | 3420  |
|          | mpg      | 14   | 24.77273 | 41    |
|          | rep78    | 3    | 4.285714 | 5     |
| Total    | price    | 3291 | 6165.257 | 15906 |
|          | weight   | 1760 | 3019.459 | 4840  |
|          | mpg      | 12   | 21.2973  | 41    |
|          | rep78    | 1    | 3.405797 | 5     |

Finally, `tabstat` can also be used to enhance `summarize` so we can specify the statistics to be displayed. For instance, we can display the number of observations, the mean, the coefficient of variation, and the 25%, 50%, and 75% quantiles for a list of variables.

```
. tabstat price weight mpg rep78, stat(n mean cv q) col(stat)
```

| variable | N  | mean     | cv       | p25  | p50    | p75  |
|----------|----|----------|----------|------|--------|------|
| price    | 74 | 6165.257 | .478406  | 4195 | 5006.5 | 6342 |
| weight   | 74 | 3019.459 | .2573949 | 2240 | 3190   | 3600 |
| mpg      | 74 | 21.2973  | .2716543 | 18   | 20     | 25   |
| rep78    | 69 | 3.405797 | .290661  | 3    | 3      | 4    |

Because we did not specify the `by()` option, these statistics were not displayed for the subgroups of the data formed by the categories of the `by()` variable.



## Methods and formulas

`tabstat` is implemented as an ado-file.

## Acknowledgments

The `tabstat` command was written by Jeroen Weesie and Vincent Buskens of Utrecht University in The Netherlands.

## Also see

[R] [summarize](#) — Summary statistics

[R] [table](#) — Tables of summary statistics

[R] [tabulate, summarize\(\)](#) — One- and two-way tables of summary statistics

[D] [collapse](#) — Make dataset of summary statistics

Syntax

One-way tables of frequencies

```
tabulate varname [if] [in] [weight] [, tabulate1_options]
```

One-way table for each variable—a convenience tool

```
tab1 varlist [if] [in] [weight] [, tab1_options]
```

| tabulate1_options           | description                                                                  |
|-----------------------------|------------------------------------------------------------------------------|
| Main                        |                                                                              |
| subpop( <i>varname</i> )    | exclude observations for which <i>varname</i> = 0                            |
| missing                     | treat missing values like other values                                       |
| nofreq                      | do not display frequencies                                                   |
| no label                    | display numeric codes rather than value labels                               |
| plot                        | produce a bar chart of the relative frequencies                              |
| sort                        | display the table in descending order of frequency                           |
| Advanced                    |                                                                              |
| generate( <i>stubname</i> ) | create indicator variables for <i>stubname</i>                               |
| matcell( <i>matname</i> )   | save frequencies in <i>matname</i> ; programmer's option                     |
| matrow( <i>matname</i> )    | save unique values of <i>varname</i> in <i>matname</i> ; programmer's option |

| tab1_options             | description                                        |
|--------------------------|----------------------------------------------------|
| Main                     |                                                    |
| subpop( <i>varname</i> ) | exclude observations for which <i>varname</i> = 0  |
| missing                  | treat missing values like other values             |
| nofreq                   | do not display frequencies                         |
| no label                 | display numeric codes rather than value labels     |
| plot                     | produce a bar chart of the relative frequencies    |
| sort                     | display the table in descending order of frequency |

by is allowed with tabulate and tab1; see [D] by.  
fweights, aweights, and iweights are allowed by tabulate. fweights are allowed by tab1. See [U] 11.1.6 weight.

## Menu

### **tabulate oneway**

Statistics > Summaries, tables, and tests > Tables > One-way tables

### **tabulate ..., generate()**

Data > Create or change data > Other variable-creation commands > Create indicator variables

### **tab1**

Statistics > Summaries, tables, and tests > Tables > Multiple one-way tables

## Description

**tabulate** produces one-way tables of frequency counts.

For information about two-way tables of frequency counts along with various measures of association, including the common Pearson  $\chi^2$ , the likelihood-ratio  $\chi^2$ , Cramér's  $V$ , Fisher's exact test, Goodman and Kruskal's gamma, and Kendall's  $\tau_b$ , see [\[R\] tabulate twoway](#).

**tab1** produces a one-way tabulation for each variable specified in *varlist*.

Also see [\[R\] table](#) and [\[R\] tabstat](#) if you want one-, two-, or  $n$ -way tables of frequencies and a wide variety of summary statistics. See [\[R\] tabulate, summarize\(\)](#) for a description of **tabulate** with the **summarize()** option; it produces tables (breakdowns) of means and standard deviations. **table** is better than **tabulate**, **summarize()**, but **tabulate**, **summarize()** is faster. See [\[ST\] epitab](#) for  $2 \times 2$  tables with statistics of interest to epidemiologists.

## Options

### Main

**subpop(*varname*)** excludes observations for which *varname* = 0 in tabulating frequencies. The mathematical results of **tabulate ...**, **subpop(myvar)** are the same as **tabulate ... if myvar !=0**, but the table may be presented differently. The identities of the rows and columns will be determined from all the data, including the *myvar* = 0 group, so there may be entries in the table with frequency 0.

Consider tabulating **answer**, a variable that takes on values 1, 2, and 3, but consider tabulating it just for the **male==1** subpopulation. Assume that **answer** is never 2 in this group. **tabulate answer if male==1** produces a table with two rows: one for answer 1 and one for answer 3. There will be no row for answer 2 because answer 2 was never observed. **tabulate answer, subpop(male)** produces a table with three rows. The row for answer 2 will be shown as having 0 frequency.

**missing** requests that missing values be treated like other values in calculations of counts, percentages, and other statistics.

**nofreq** suppresses the printing of the frequencies.

**no label** causes the numeric codes to be displayed rather than the value labels.

**plot** produces a bar chart of the relative frequencies in a one-way table. (Also see [\[R\] histogram](#).)

**sort** puts the table in descending order of frequency (and ascending order of the variable within equal values of frequency).



## Advanced

`generate(stubname)` creates a set of indicator variables (*stubname1*, *stubname2*, ...) reflecting the observed values of the tabulated variable. The `generate()` option may not be used with the `by` prefix.

`matcell(matname)` saves the reported frequencies in *matname*. This option is for use by programmers.

`matrow(matname)` saves the numeric values of the  $r \times 1$  row stub in *matname*. This option is for use by programmers. `matrow()` may not be specified if the row variable is a string.

## Limits

One-way tables may have a maximum of 12,000 rows (Stata/MP and Stata/SE), 3,000 rows (Stata/IC), or 500 rows (Small Stata).

## Remarks

Remarks are presented under the following headings:

*tabulate*  
*tab1*

For each value of a specified variable, `tabulate` reports the number of observations with that value. The number of times a value occurs is called its *frequency*.

## tabulate

### ► Example 1

We have data summarizing the speed limit and the accident rate per million vehicle miles along various Minnesota highways in 1973. The variable containing the speed limit is called `spdlimit`. If we summarize the variable, we obtain its mean and standard deviation:

```
. use http://www.stata-press.com/data/r11/hiway
(Minnesota Highway Data, 1973)
```

```
. summarize spdlimit
```

| Variable | Obs | Mean | Std. Dev. | Min | Max |
|----------|-----|------|-----------|-----|-----|
| spdlimit | 39  | 55   | 5.848977  | 40  | 70  |

The average speed limit is 55 miles per hour. We can learn more about this variable by tabulating it:

```
. tabulate spdlimit
```

| Speed limit | Freq. | Percent | Cum.   |
|-------------|-------|---------|--------|
| 40          | 1     | 2.56    | 2.56   |
| 45          | 3     | 7.69    | 10.26  |
| 50          | 7     | 17.95   | 28.21  |
| 55          | 15    | 38.46   | 66.67  |
| 60          | 11    | 28.21   | 94.87  |
| 65          | 1     | 2.56    | 97.44  |
| 70          | 1     | 2.56    | 100.00 |
| Total       | 39    | 100.00  |        |

We see that one highway has a speed limit of 40 miles per hour, three have speed limits of 45, 7 of 50, and so on. The column labeled `Percent` shows the percentage of highways in the dataset that have the indicated speed limit. For instance, 38.46% of highways in our dataset have a speed limit of 55 miles per hour. The final column shows the cumulative percentage. We see that 66.67% of highways in our dataset have a speed limit of 55 miles per hour or less.



➤ Example 2

The `plot` option places a sideways histogram alongside the table:

```
. tabulate spdlimit, plot
```

| Speed limit | Freq. |       |
|-------------|-------|-------|
| 40          | 1     | *     |
| 45          | 3     | ***   |
| 50          | 7     | ***** |
| 55          | 15    | ***** |
| 60          | 11    | ***** |
| 65          | 1     | *     |
| 70          | 1     | *     |
| Total       | 39    |       |

Of course, `graph` can produce better-looking histograms; see [\[R\] histogram](#).



➤ Example 3

`tabulate` labels tables using *variable* and *value labels* if they exist. To demonstrate how this works, let's add a new variable to our dataset that categorizes `spdlimit` into three categories. We will call this new variable `spdcat`:

```
. generate spdcat=recode(spdlimit,50,60,70)
```

The `recode()` function divides `spdlimit` into 50 miles per hour or below, 51–60, and above 60; see [\[D\] functions](#). We specified the breakpoints in the arguments `(spdlimit,50,60,70)`. The first argument is the variable to be recoded. The second argument is the first breakpoint, the third argument is the second breakpoint, and so on. We can specify as many breakpoints as we wish.

`recode()` used our arguments not only as the breakpoints but also to label the results. If `spdlimit` is less than or equal to 50, `spdcat` is set to 50; if `spdlimit` is between 51 and 60, `spdcat` is 60; otherwise, `spdcat` is arbitrarily set to 70. (See [\[U\] 25 Working with categorical data and factor variables](#).)

Because we just created the variable `spdcat`, it is not yet labeled. When we make a table using this variable, `tabulate` uses the variable's name to label it:

```
. tabulate spdcat
```

| spdcat | Freq. | Percent | Cum.   |
|--------|-------|---------|--------|
| 50     | 11    | 28.21   | 28.21  |
| 60     | 26    | 66.67   | 94.87  |
| 70     | 2     | 5.13    | 100.00 |
| Total  | 39    | 100.00  |        |

Even though the table is not well labeled, `recode()`'s coding scheme provides us with clues as to the table's meaning. The first line of the table corresponds to 50 miles per hour and below, the next to 51 through 60 miles per hour, and the last to above 60 miles per hour.

We can improve this table by labeling the values and variables:

```
. label define scat 50 "40 to 50" 60 "55 to 60" 70 "Above 60"
. label values spdcat scat
. label variable spdcat "Speed Limit Category"
```

We define a *value label* called `scat` that attaches labels to the numbers 50, 60, and 70 using the `label define` command; see [\[U\] 12.6.3 Value labels](#). We label the value 50 as '40 to 50', because we looked back at our original tabulation in the first example and saw that the speed limit was never less than 40. Similarly, we could have labeled the last category '65 to 70' because the speed limit is never greater than 70 miles per hour.

Next we requested that Stata label the values of the new variable `spdcat` using the value label `scat`. Finally, we labeled our variable `Speed Limit Category`. We are now ready to `tabulate` the result:

```
. tabulate spdcat
```

| Speed Limit<br>Category | Freq. | Percent | Cum.   |
|-------------------------|-------|---------|--------|
| 40 to 50                | 11    | 28.21   | 28.21  |
| 55 to 60                | 26    | 66.67   | 94.87  |
| Above 60                | 2     | 5.13    | 100.00 |
| Total                   | 39    | 100.00  |        |

◀

## ► Example 4

If we have missing values in our dataset, `tabulate` ignores them unless we explicitly indicate otherwise. We have no missing data in our example, so let's add some:

```
. replace spdcat=. in 39
(1 real change made, 1 to missing)
```

We changed the first observation on `spdcat` to *missing*. Let's now `tabulate` the result:

```
. tabulate spdcat
```

| Speed Limit<br>Category | Freq. | Percent | Cum.   |
|-------------------------|-------|---------|--------|
| 40 to 50                | 11    | 28.95   | 28.95  |
| 55 to 60                | 26    | 68.42   | 97.37  |
| Above 60                | 1     | 2.63    | 100.00 |
| Total                   | 38    | 100.00  |        |

Comparing this output with that in the previous example, we see that the total frequency count is now one less than it was—38 rather than 39. Also, the 'Above 60' category now has only one observation where it used to have two, so we evidently changed a road with a high speed limit.

We want `tabulate` to treat missing values just as it treats numbers, so we specify the `missing` option:

```
. tabulate spdcat, missing
```

| Speed Limit<br>Category | Freq. | Percent | Cum.   |
|-------------------------|-------|---------|--------|
| 40 to 50                | 11    | 28.21   | 28.21  |
| 55 to 60                | 26    | 66.67   | 94.87  |
| Above 60                | 1     | 2.56    | 97.44  |
| .                       | 1     | 2.56    | 100.00 |
| Total                   | 39    | 100.00  |        |

We now see our missing value—the last category, labeled ‘.’, shows a frequency count of 1. The table sum is once again 39.

Let’s put our dataset back as it was originally:

```
. replace spdcat=70 in 39
(1 real change made)
```



❏ Technical note

`tabulate` also can automatically create indicator variables from categorical variables. We will briefly review that capability here, but see [\[U\] 25 Working with categorical data and factor variables](#) for a complete description. Let’s begin by describing our highway dataset:

```
. describe
```

Contains data from <http://www.stata-press.com/data/r11/hiway.dta>

|       |                            |                              |
|-------|----------------------------|------------------------------|
| obs:  | 39                         | Minnesota Highway Data, 1973 |
| vars: | 2                          | 16 Nov 2008 12:39            |
| size: | 507 (99.9% of memory free) |                              |

| variable name | storage<br>type | display<br>format | value<br>label | variable label                             |
|---------------|-----------------|-------------------|----------------|--------------------------------------------|
| spdlimit      | byte            | %8.0g             |                | Speed limit                                |
| rate          | float           | %9.0g             | rcat           | Accident rate per million<br>vehicle miles |
| spdcat        | float           | %9.0g             | scat           | Speed Limit Category                       |

Sorted by:

Note: dataset has changed since last saved

Our dataset contains three variables. We will type `tabulate spdcat, generate(spd)`, `describe` our data, and then explain what happened.

```
. tabulate spdcat, generate(spd)
```

| Speed Limit<br>Category | Freq. | Percent | Cum.   |
|-------------------------|-------|---------|--------|
| 40 to 50                | 11    | 28.21   | 28.21  |
| 55 to 60                | 26    | 66.67   | 94.87  |
| Above 60                | 2     | 5.13    | 100.00 |
| Total                   | 39    | 100.00  |        |

```
. describe
Contains data from http://www.stata-press.com/data/r11/hiway.dta
 obs: 39 Minnesota Highway Data, 1973
 vars: 6 16 Nov 2006 12:39
size: 624 (99.9% of memory free)
```

| variable name | storage<br>type | display<br>format | value<br>label | variable label                             |
|---------------|-----------------|-------------------|----------------|--------------------------------------------|
| spdlimit      | byte            | %8.0g             |                | Speed limit                                |
| rate          | float           | %9.0g             | rcat           | Accident rate per million<br>vehicle miles |
| spdcat        | float           | %9.0g             | scat           | Speed Limit Category                       |
| spd1          | byte            | %8.0g             |                | spdcat==40 to 50                           |
| spd2          | byte            | %8.0g             |                | spdcat==55 to 60                           |
| spd3          | byte            | %8.0g             |                | spdcat==Above 60                           |

Sorted by:

Note: dataset has changed since last saved

When we typed `tabulate` with the `generate()` option, Stata responded by producing a one-way frequency table, so it appeared that the option did nothing. Yet when we `describe` our dataset, we find that we now have *six* variables instead of the original three. The new variables are named `spd1`, `spd2`, and `spd3`.

When we specify the `generate()` option, we are telling Stata to not only produce the table but also create a set of indicator variables that correspond to that table. Stata adds a numeric suffix to the name we specify in the parentheses. `spd1` refers to the first line of the table, `spd2` to the second line, and so on. Also Stata labels the variables so that we know what they mean. `spd1` is an indicator variable that is *true* (takes on the value 1) when `spdcat` is between 40 and 50; otherwise, it is zero. (There is an exception: if `spdcat` is missing, so are the `spd1`, `spd2`, and `spd3` variables. This did not happen in our dataset.)

We want to prove our claim. Because we have not yet introduced two-way tabulations, we will use the `summarize` statement:

```
. summarize spdlimit if spd1==1
```

| Variable | Obs | Mean     | Std. Dev. | Min | Max |
|----------|-----|----------|-----------|-----|-----|
| spdlimit | 11  | 47.72727 | 3.437758  | 40  | 50  |

```
. summarize spdlimit if spd2==1
```

| Variable | Obs | Mean     | Std. Dev. | Min | Max |
|----------|-----|----------|-----------|-----|-----|
| spdlimit | 26  | 57.11538 | 2.519157  | 55  | 60  |

```
. summarize spdlimit if spd3==1
```

| Variable | Obs | Mean | Std. Dev. | Min | Max |
|----------|-----|------|-----------|-----|-----|
| spdlimit | 2   | 67.5 | 3.535534  | 65  | 70  |

Notice the indicated minimum and maximum in each of the tables above. When we restrict the sample to `spd1`, `spdlimit` is between 40 and 50; when we restrict the sample to `spd2`, `spdlimit` is between 55 and 60; when we restrict the sample to `spd3`, `spdlimit` is between 65 and 70.

Thus `tabulate` provides an easy way to create indicator (sometimes called dummy) variables. For an overview of indicator and categorical variables, see [\[U\] 25 Working with categorical data and factor variables](#).



## tab1

`tab1` is a convenience tool. Typing

```
. tab1 myvar thisvar thatvar, plot
```

is equivalent to typing

```
. tabulate myvar, plot
. tabulate thisvar, plot
. tabulate thatvar, plot
```

## Saved results

`tabulate` and `tab1` save the following in `r()`:

Scalars

`r(N)`    number of observations

`r(r)`    number of rows

## Methods and formulas

`tab1` is implemented as an ado-file.

## Reference

Harrison, D. A. 2006. [Stata tip 34: Tabulation by listing](#). *Stata Journal* 6: 425–427.

## Also see

[R] [table](#) — Tables of summary statistics

[R] [tabstat](#) — Display table of summary statistics

[R] [tabulate twoway](#) — Two-way tables of frequencies

[R] [tabulate, summarize\(\)](#) — One- and two-way tables of summary statistics

[D] [collapse](#) — Make dataset of summary statistics

[ST] [epitab](#) — Tables for epidemiologists

[SVY] [svy: tabulate oneway](#) — One-way tables for survey data

[SVY] [svy: tabulate twoway](#) — Two-way tables for survey data

[XT] [xttab](#) — Tabulate xt data

[U] [12.6.3 Value labels](#)

[U] [25 Working with categorical data and factor variables](#)

Syntax

Two-way tables

```
tabulate varname1 varname2 [if] [in] [weight] [, options]
```

Two-way tables for all possible combinations—a convenience tool

```
tab2 varlist [if] [in] [weight] [, options]
```

Immediate form of two-way tabulations

```
tabi #11 #12 [...] \ #21 #22 [...] [\ ...] [, options]
```

| options | description |
|---------|-------------|
|---------|-------------|

---

|                    |                                                                    |
|--------------------|--------------------------------------------------------------------|
| Main               |                                                                    |
| <u>chi2</u>        | report Pearson’s $\chi^2$                                          |
| <u>exact</u> [(#)] | report Fisher’s exact test                                         |
| <u>gamma</u>       | report Goodman and Kruskal’s gamma                                 |
| <u>lrchi2</u>      | report likelihood-ratio $\chi^2$                                   |
| <u>taub</u>        | report Kendall’s $\tau_b$                                          |
| <u>V</u>           | report Cramér’s $V$                                                |
| <u>cchi2</u>       | report Pearson’s $\chi^2$ in each cell                             |
| <u>column</u>      | report relative frequency within its column of each cell           |
| <u>row</u>         | report relative frequency within its row of each cell              |
| <u>clrchi2</u>     | report likelihood-ratio $\chi^2$ in each cell                      |
| <u>cell</u>        | report the relative frequency of each cell                         |
| <u>expected</u>    | report expected frequency in each cell                             |
| <u>nofreq</u>      | do not display frequencies                                         |
| <u>missing</u>     | treat missing values like other values                             |
| <u>wrap</u>        | do not wrap wide tables                                            |
| [no]key            | report/suppress cell contents key                                  |
| <u>nolabel</u>     | display numeric codes rather than value labels                     |
| <u>nolog</u>       | do not display enumeration log for Fisher’s exact test             |
| * <u>firstonly</u> | show only tables that include the first variable in <i>varlist</i> |

|                  |                                                                                           |
|------------------|-------------------------------------------------------------------------------------------|
| Advanced         |                                                                                           |
| matcell(matname) | save frequencies in <i>matname</i> ; programmer’s option                                  |
| matrow(matname)  | save unique values of <i>varname</i> <sub>1</sub> in <i>matname</i> ; programmer’s option |
| matcol(matname)  | save unique values of <i>varname</i> <sub>2</sub> in <i>matname</i> ; programmer’s option |
| †replace         | replace current data with given cell frequencies                                          |
| †all             | equivalent to specifying <i>chi2 lrchi2 V gamma taub</i>                                  |

---

\***firstonly** is available only for **tab2**.

†**replace** is available only for **tab1**.

†**all** does not appear in the dialog box.

**by** is allowed with **tabulate** and **tab2**; see [\[D\] by](#).

**fweights**, **aweight**s, and **iweight**s are allowed by **tabulate**. **fweights** are allowed by **tab2**.

See [\[U\] 11.1.6 weight](#).

## Menu

### **tabulate**

Statistics > Summaries, tables, and tests > Tables > Two-way tables with measures of association

### **tab2**

Statistics > Summaries, tables, and tests > Tables > All possible two-way tabulations

### **tabi**

Statistics > Summaries, tables, and tests > Tables > Table calculator

## Description

**tabulate** produces two-way tables of frequency counts, along with various measures of association, including the common Pearson's  $\chi^2$ , the likelihood-ratio  $\chi^2$ , Cramér's  $V$ , Fisher's exact test, Goodman and Kruskal's gamma, and Kendall's  $\tau_b$ .

Line size is respected. That is, if you resize the Results window before running **tabulate**, the resulting two-way tabulation will take advantage of the available horizontal space. Stata for Unix(console) users can instead use the **set linesize** command to take advantage of this feature.

**tab2** produces all possible two-way tabulations of the variables specified in *varlist*.

**tabi** displays the  $r \times c$  table, using the values specified; rows are separated by '\'. If no options are specified, it is as if **exact** were specified for  $2 \times 2$  tables and **chi2** were specified otherwise. See [\[U\] 19 Immediate commands](#) for a general description of immediate commands. Specifics for **tabi** can be found toward the end of [\[R\] tabulate twoway](#).

See [\[R\] tabulate oneway](#) if you want one-way tables of frequencies. See [\[R\] table](#) and [\[R\] tabstat](#) if you want one-, two-, or  $n$ -way tables of frequencies and a wide variety of summary statistics. See [\[R\] tabulate, summarize\(\)](#) for a description of **tabulate** with the **summarize()** option; it produces tables (breakdowns) of means and standard deviations. **table** is better than **tabulate**, **summarize()**, but **tabulate, summarize()** is faster. See [\[ST\] epitab](#) for  $2 \times 2$  tables with statistics of interest to epidemiologists.

## Options

---

Main

**chi2** calculates and displays Pearson's  $\chi^2$  for the hypothesis that the rows and columns in a two-way table are independent. **chi2** may not be specified if **aweight**s or **iweight**s are specified.



**exact**[(#)] displays the significance calculated by Fisher's exact test and may be applied to  $r \times c$  as well as to  $2 \times 2$  tables. For  $2 \times 2$  tables, both one- and two-sided probabilities are displayed. For  $r \times c$  tables, one-sided probabilities are displayed. The optional positive integer # is a multiplier on the amount of memory that the command is permitted to consume. The default is 1. This option should not be necessary for reasonable  $r \times c$  tables. If the command terminates with error 910, try **exact**(2). The maximum row or column dimension allowed when computing Fisher's exact test is the maximum row or column dimension for **tabulate** (see **help limits**).

**gamma** displays Goodman and Kruskal's gamma, along with its asymptotic standard error. **gamma** is appropriate only when both variables are ordinal. **gamma** may not be specified if **aweights** or **weights** are specified.

**lrchi2** displays the likelihood-ratio  $\chi^2$  statistic. **lrchi2** may not be specified if **aweights** or **weights** are specified.

**taub** displays Kendall's  $\tau_b$  along with its asymptotic standard error. **taub** is appropriate only when both variables are ordinal. **taub** may not be specified if **aweights** or **weights** are specified.

**V** (note capitalization) displays Cramér's  $V$ . **V** may not be specified if **aweights** or **weights** are specified.

**cchi2** displays each cell's contribution to Pearson's chi-squared in a two-way table.

**column** displays the relative frequency of each cell within its column in a two-way table.

**row** displays the relative frequency of each cell within its row in a two-way table.

**clrch2** displays each cell's contribution to the likelihood-ratio chi-squared in a two-way table.

**cell** displays the relative frequency of each cell in a two-way table.

**expected** displays the expected frequency of each cell in a two-way table.

**nofreq** suppresses the printing of the frequencies.

**missing** requests that missing values be treated like other values in calculations of counts, percentages, and other statistics.

**wrap** requests that Stata take no action on wide, two-way tables to make them readable. Unless **wrap** is specified, wide tables are broken into pieces to enhance readability.

[**no**]**key** suppresses or forces the display of a key above two-way tables. The default is to display the key if more than one cell statistic is requested, and otherwise to omit it. **key** forces the display of the key. **nokey** suppresses its display.

**no label** causes the numeric codes to be displayed rather than the value labels.

**nolog** suppresses the display of the log for Fisher's exact test. Using Fisher's exact test requires counting all tables that have a probability exceeding that of the observed table given the observed row and column totals. The log counts down each stage of the network computations, starting from the number of columns and counting down to 1, displaying the number of nodes in the network at each stage. A log is not displayed for  $2 \times 2$  tables.

**firstonly**, available only with **tab2**, restricts the output to only those tables that include the first variable in **varlist**. Use this option to interact one variable with a set of others.

---

#### Advanced

**matcell**(*matname*) saves the reported frequencies in *matname*. This option is for use by programmers.

**matrow**(*matname*) saves the numeric values of the  $r \times 1$  row stub in *matname*. This option is for use by programmers. **matrow**() may not be specified if the row variable is a string.

`matcol(matname)` saves the numeric values of the  $1 \times c$  column stub in *matname*. This option is for use by programmers. `matcol()` may not be specified if the column variable is a string.

`replace` indicates that the immediate data specified as arguments to the `tabi` command be left as the current data in place of whatever data were there.

The following option is available with `tabulate` but is not shown in the dialog box:

`all` is equivalent to specifying `chi2 lrchi2 V gamma taub`. Note the omission of `exact`. When `all` is specified, `no` may be placed in front of the other options. `all noV` requests all measures of association but Cramér’s  $V$  (and Fisher’s exact). `all exact` requests all association measures, including Fisher’s exact test. `all` may not be specified if `aweights` or `iweights` are specified.

Limits

Two-way tables may have a maximum of 1,200 rows and 80 columns (Stata/MP and Stata/SE), 300 rows and 20 columns (Stata/IC), or 160 rows and 20 columns (Small Stata). If larger tables are needed, see [\[R\] table](#).

Remarks

Remarks are presented under the following headings:

- [tabulate](#)
- [Measures of association](#)
- [N-way tables](#)
- [Weighted data](#)
- [Tables with immediate data](#)
- [tab2](#)

For each value of a specified variable (or a set of values for a pair of variables), `tabulate` reports the number of observations with that value. The number of times a value occurs is called its *frequency*.

tabulate

➤ Example 1

`tabulate` will make two-way tables if we specify two variables following the word `tabulate`. In our highway dataset, we have a variable called `rate` that divides the accident rate into three categories: below 4, 4–7, and above 7 per million vehicle miles. Let’s make a table of the speed limit category and the accident-rate category:

```
. use http://www.stata-press.com/data/r11/hiway2
(Minnesota Highway Data, 1973)
. tabulate spdcat rate
```

| Speed<br>Limit<br>Category | Accident rate per million<br>vehicle miles |     |         | Total |
|----------------------------|--------------------------------------------|-----|---------|-------|
|                            | Below 4                                    | 4–7 | Above 7 |       |
| 40 to 50                   | 3                                          | 5   | 3       | 11    |
| 55 to 60                   | 19                                         | 6   | 1       | 26    |
| Above 60                   | 2                                          | 0   | 0       | 2     |
| Total                      | 24                                         | 11  | 4       | 39    |

The table indicates that three stretches of highway have an accident rate below 4 and a speed limit of 40 to 50 miles per hour. The table also shows the row and column sums (called the *marginals*). The number of highways with a speed limit of 40 to 50 miles per hour is 11, which is the same result we obtained in our previous one-way tabulations.

Stata can present this basic table in several ways—16, to be precise—and we will show just a few below. It might be easier to read the table if we included the row percentages. For instance, of 11 highways in the lowest speed limit category, three are also in the lowest accident-rate category. Three-elevenths amounts to some 27.3%. We can ask Stata to fill in this information for us by using the `row` option:

```
. tabulate spdcat rate, row
```

| Key                        |                                            |       |         |        |
|----------------------------|--------------------------------------------|-------|---------|--------|
| <i>frequency</i>           |                                            |       |         |        |
| <i>row percentage</i>      |                                            |       |         |        |
| Speed<br>Limit<br>Category | Accident rate per million<br>vehicle miles |       |         | Total  |
|                            | Below 4                                    | 4-7   | Above 7 |        |
| 40 to 50                   | 3                                          | 5     | 3       | 11     |
|                            | 27.27                                      | 45.45 | 27.27   | 100.00 |
| 55 to 60                   | 19                                         | 6     | 1       | 26     |
|                            | 73.08                                      | 23.08 | 3.85    | 100.00 |
| Above 60                   | 2                                          | 0     | 0       | 2      |
|                            | 100.00                                     | 0.00  | 0.00    | 100.00 |
| Total                      | 24                                         | 11    | 4       | 39     |
|                            | 61.54                                      | 28.21 | 10.26   | 100.00 |

The number listed below each frequency is the percentage of cases that each cell represents out of its row. That is easy to remember because we see 100% listed in the “Total” column. The bottom row is also informative. We see that 61.54% of all the highways in our dataset fall into the lowest accident-rate category, that 28.21% are in the middle category, and that 10.26% are in the highest.

`tabulate` can calculate column percentages and cell percentages, as well. It does so when we specify the `column` or `cell` options, respectively. We can even specify them together. Below is a table that includes everything:

(Continued on next page)

```
. tabulate spdcat rate, row column cell
```

| Key                      |
|--------------------------|
| <i>frequency</i>         |
| <i>row percentage</i>    |
| <i>column percentage</i> |
| <i>cell percentage</i>   |

| Speed<br>Limit<br>Category | Accident rate per million<br>vehicle miles |        |         | Total  |
|----------------------------|--------------------------------------------|--------|---------|--------|
|                            | Below 4                                    | 4-7    | Above 7 |        |
| 40 to 50                   | 3                                          | 5      | 3       | 11     |
|                            | 27.27                                      | 45.45  | 27.27   | 100.00 |
|                            | 12.50                                      | 45.45  | 75.00   | 28.21  |
|                            | 7.69                                       | 12.82  | 7.69    | 28.21  |
| 55 to 60                   | 19                                         | 6      | 1       | 26     |
|                            | 73.08                                      | 23.08  | 3.85    | 100.00 |
|                            | 79.17                                      | 54.55  | 25.00   | 66.67  |
|                            | 48.72                                      | 15.38  | 2.56    | 66.67  |
| Above 60                   | 2                                          | 0      | 0       | 2      |
|                            | 100.00                                     | 0.00   | 0.00    | 100.00 |
|                            | 8.33                                       | 0.00   | 0.00    | 5.13   |
|                            | 5.13                                       | 0.00   | 0.00    | 5.13   |
| Total                      | 24                                         | 11     | 4       | 39     |
|                            | 61.54                                      | 28.21  | 10.26   | 100.00 |
|                            | 100.00                                     | 100.00 | 100.00  | 100.00 |
|                            | 61.54                                      | 28.21  | 10.26   | 100.00 |

The number at the top of each cell is the frequency count. The second number is the row percentage—they sum to 100% going across the table. The third number is the column percentage—they sum to 100% going down the table. The bottom number is the cell percentage—they sum to 100% going down all the columns and across all the rows. For instance, highways with a speed limit above 60 miles per hour and in the lowest accident rate category account for 100% of highways with a speed limit above 60 miles per hour; 8.33% of highways in the lowest accident-rate category; and 5.13% of all our data.

A fourth option, `nofreq`, tells Stata not to print the frequency counts. To construct a table consisting of only row percentages, we type

```
. tabulate spdcat rate, row nofreq
```

| Speed<br>Limit<br>Category | Accident rate per million<br>vehicle miles |       |         | Total  |
|----------------------------|--------------------------------------------|-------|---------|--------|
|                            | Below 4                                    | 4-7   | Above 7 |        |
| 40 to 50                   | 27.27                                      | 45.45 | 27.27   | 100.00 |
| 55 to 60                   | 73.08                                      | 23.08 | 3.85    | 100.00 |
| Above 60                   | 100.00                                     | 0.00  | 0.00    | 100.00 |
| Total                      | 61.54                                      | 28.21 | 10.26   | 100.00 |

## Measures of association

### ► Example 2

`tabulate` will calculate the Pearson  $\chi^2$  test for the independence of the rows and columns if we specify the `chi2` option. Suppose that we have 1980 census data on 956 cities in the United States and wish to compare the age distribution across regions of the country. Assume that `agecat` is the median age in each city and that `region` denotes the region of the country in which the city is located.

```
. use http://www.stata-press.com/data/r11/citytemp2
(City Temperature Data)
```

```
. tabulate region agecat, chi2
```

| Census<br>Region | 19-29 | agecat<br>30-34 | 35+ | Total |
|------------------|-------|-----------------|-----|-------|
| NE               | 46    | 83              | 37  | 166   |
| N Cntrl          | 162   | 92              | 30  | 284   |
| South            | 139   | 68              | 43  | 250   |
| West             | 160   | 73              | 23  | 256   |
| Total            | 507   | 316             | 133 | 956   |

Pearson chi2(6) = 61.2877 Pr = 0.000

We obtain the standard two-way table and, at the bottom, a summary of the  $\chi^2$  test. Stata informs us that the  $\chi^2$  associated with this table has 6 degrees of freedom and is 61.29. The observed differences are significant.

The table is, perhaps, easier to understand if we suppress the frequencies and print just the row percentages:

```
. tabulate region agecat, row nofreq chi2
```

| Census<br>Region | 19-29 | agecat<br>30-34 | 35+   | Total  |
|------------------|-------|-----------------|-------|--------|
| NE               | 27.71 | 50.00           | 22.29 | 100.00 |
| N Cntrl          | 57.04 | 32.39           | 10.56 | 100.00 |
| South            | 55.60 | 27.20           | 17.20 | 100.00 |
| West             | 62.50 | 28.52           | 8.98  | 100.00 |
| Total            | 53.03 | 33.05           | 13.91 | 100.00 |

Pearson chi2(6) = 61.2877 Pr = 0.000

◀

### ► Example 3

We have data on dose level and outcome for a set of patients and wish to evaluate the association between the two variables. We can obtain all the association measures by specifying the `all` and `exact` options:

(Continued on next page)

```
. use http://www.stata-press.com/data/r11/dose
. tabulate dose function, all exact

Enumerating sample-space combinations:
stage 3: enumerations = 1
stage 2: enumerations = 9
stage 1: enumerations = 0
```

| Dosage | Function |        |    | Total |
|--------|----------|--------|----|-------|
|        | < 1 hr   | 1 to 4 | 4+ |       |
| 1/day  | 20       | 10     | 2  | 32    |
| 2/day  | 16       | 12     | 4  | 32    |
| 3/day  | 10       | 16     | 6  | 32    |
| Total  | 46       | 38     | 12 | 96    |

Pearson

chi2(4) = 6.7780

Pr = 0.148

likelihood-ratio

chi2(4) = 6.9844

Pr = 0.137

Cramer's V = 0.1879

gamma = 0.3689

ASE = 0.129

Kendall's tau-b = 0.2378

ASE = 0.086

Fisher's exact = 0.145

We find evidence of association but not enough to be truly convincing.

If we had not also specified the `exact` option, we would not have obtained Fisher's exact test. Stata can calculate this statistic both for  $2 \times 2$  tables and for  $r \times c$ . For  $2 \times 2$  tables, the calculation is almost instant. On more general tables, however, the calculation can take longer.

We carefully constructed our example so that `all` would be meaningful. Kendall's  $\tau_b$  and Goodman and Kruskal's gamma are relevant only when both dimensions of the table can be ordered, say, from low to high or from worst to best. The other statistics, however, are always applicable.

□ Technical note

Be careful when attempting to compute the  $p$ -value for Fisher's exact test because the number of tables that contribute to the  $p$ -value can be extremely large and a solution may not be feasible. The errors that are indicative of this situation are errors 910, exceeded memory limitations, and 1401, integer overflow due to large row-margin frequencies. If execution terminates because of memory limitations, use `exact(2)` to permit the algorithm to consume twice the memory, `exact(3)` for three times the memory, etc. The default memory usage should be sufficient for reasonable tables.

N-way tables

If you need more than two-way tables, your best alternative to is use `table`, not `tabulate`; see [\[R\] table](#).

The technical note below shows you how to use `tabulate` to create a sequence of two-way tables that together form, in effect, a three-way table, but using `table` is easy and produces prettier results:

```
. use http://www.stata-press.com/data/r11/birthcat
(City data)
. table birthcat region agecat, c(freq)
```

| birthcat | agecat and Census Region |         |       |      |       |         |       |      |
|----------|--------------------------|---------|-------|------|-------|---------|-------|------|
|          | 19-29                    |         |       |      | 30-34 |         |       |      |
|          | NE                       | N Cntrl | South | West | NE    | N Cntrl | South | West |
| 29-136   | 11                       | 23      | 11    | 11   | 34    | 27      | 10    | 8    |
| 137-195  | 31                       | 97      | 65    | 46   | 48    | 58      | 45    | 42   |
| 196-529  | 4                        | 38      | 59    | 91   | 1     | 3       | 12    | 21   |

| birthcat | agecat and Census Region |         |       |      |
|----------|--------------------------|---------|-------|------|
|          | 35+                      |         |       |      |
|          | NE                       | N Cntrl | South | West |
| 29-136   | 34                       | 26      | 27    | 18   |
| 137-195  | 3                        | 4       | 7     | 4    |
| 196-529  |                          |         | 4     |      |

## □ Technical note

We can make  $n$ -way tables by combining the [by varlist](#) prefix with `tabulate`. Continuing with the dataset of 956 cities, say that we want to make a table of age category by birth-rate category by region of the country. The birth-rate category variable is named `birthcat` in our dataset. To make separate tables for each age category, we would type

```
. by agecat, sort: tabulate birthcat region
```

```
-> agecat = 19-29
```

| birthcat | Census Region |         |       |      | Total |
|----------|---------------|---------|-------|------|-------|
|          | NE            | N Cntrl | South | West |       |
| 29-136   | 11            | 23      | 11    | 11   | 56    |
| 137-195  | 31            | 97      | 65    | 46   | 239   |
| 196-529  | 4             | 38      | 59    | 91   | 192   |
| Total    | 46            | 158     | 135   | 148  | 487   |

```
-> agecat = 30-34
```

| birthcat | Census Region |         |       |      | Total |
|----------|---------------|---------|-------|------|-------|
|          | NE            | N Cntrl | South | West |       |
| 29-136   | 34            | 27      | 10    | 8    | 79    |
| 137-195  | 48            | 58      | 45    | 42   | 193   |
| 196-529  | 1             | 3       | 12    | 21   | 37    |
| Total    | 83            | 88      | 67    | 71   | 309   |

```
-> agecat = 35+
```

| birthcat | Census Region |         |       |      | Total |
|----------|---------------|---------|-------|------|-------|
|          | NE            | N Cntrl | South | West |       |
| 29-136   | 34            | 26      | 27    | 18   | 105   |
| 137-195  | 3             | 4       | 7     | 4    | 18    |
| 196-529  | 0             | 0       | 4     | 0    | 4     |
| Total    | 37            | 30      | 38    | 22   | 127   |

Weighted data

Example 4

tabulate can process weighted as well as unweighted data. As with all Stata commands, we indicate the weight by specifying the `[weight]` modifier; see [\[U\] 11.1.6 weight](#).

Continuing with our dataset of 956 cities, we also have a variable called `pop`, the population of each city. We can make a table of region by age category, weighted by population, by typing

```
. tabulate region agecat [freq=pop]
```

| Census Region | agecat     |            |            | Total      |
|---------------|------------|------------|------------|------------|
|               | 19-29      | 30-34      | 35+        |            |
| NE            | 4,721,387  | 10,421,387 | 5,323,610  | 20,466,384 |
| N Cntrl       | 16,901,550 | 8,964,756  | 4,015,593  | 29,881,899 |
| South         | 13,894,254 | 7,686,531  | 4,141,863  | 25,722,648 |
| West          | 16,698,276 | 7,755,255  | 2,375,118  | 26,828,649 |
| Total         | 52,215,467 | 34,827,929 | 15,856,184 | 102899580  |

If we specify the `cell`, `column`, or `row` options, they will also be appropriately weighted. Below we repeat the table, suppressing the counts and substituting row percentages:

```
. tabulate region agecat [freq=pop], nofreq row
```

| Census Region | agecat |       |       | Total  |
|---------------|--------|-------|-------|--------|
|               | 19-29  | 30-34 | 35+   |        |
| NE            | 23.07  | 50.92 | 26.01 | 100.00 |
| N Cntrl       | 56.56  | 30.00 | 13.44 | 100.00 |
| South         | 54.02  | 29.88 | 16.10 | 100.00 |
| West          | 62.24  | 28.91 | 8.85  | 100.00 |
| Total         | 50.74  | 33.85 | 15.41 | 100.00 |



## Tables with immediate data

### ► Example 5

`tabi` ignores the dataset in memory and uses as the table the values that we specify on the command line:

```
. tabi 30 18 \ 38 14
```

| row                      | col |    | Total |
|--------------------------|-----|----|-------|
|                          | 1   | 2  |       |
| 1                        | 30  | 18 | 48    |
| 2                        | 38  | 14 | 52    |
| Total                    | 68  | 32 | 100   |
| Fisher's exact =         |     |    | 0.289 |
| 1-sided Fisher's exact = |     |    | 0.179 |

We may specify any of the options of `tabulate` and are not limited to  $2 \times 2$  tables:

```
. tabi 30 18 38 \ 13 7 22, chi2 exact
```

Enumerating sample-space combinations:

stage 3: enumerations = 1

stage 2: enumerations = 3

stage 1: enumerations = 0

| row   | col |    |    | Total |
|-------|-----|----|----|-------|
|       | 1   | 2  | 3  |       |
| 1     | 30  | 18 | 38 | 86    |
| 2     | 13  | 7  | 22 | 42    |
| Total | 43  | 25 | 60 | 128   |

Pearson  $\chi^2(2) = 0.7967$  Pr = 0.671

Fisher's exact = 0.707

```
. tabi 30 13 \ 18 7 \ 38 22, all exact col
```

| Key                      |
|--------------------------|
| <i>frequency</i>         |
| <i>column percentage</i> |

Enumerating sample-space combinations:

stage 3: enumerations = 1

stage 2: enumerations = 3

stage 1: enumerations = 0

| row   | col    |        | Total  |
|-------|--------|--------|--------|
|       | 1      | 2      |        |
| 1     | 30     | 13     | 43     |
|       | 34.88  | 30.95  | 33.59  |
| 2     | 18     | 7      | 25     |
|       | 20.93  | 16.67  | 19.53  |
| 3     | 38     | 22     | 60     |
|       | 44.19  | 52.38  | 46.88  |
| Total | 86     | 42     | 128    |
|       | 100.00 | 100.00 | 100.00 |

```
Pearson chi2(2) = 0.7967 Pr = 0.671
likelihood-ratio chi2(2) = 0.7985 Pr = 0.671
Cramer's V = 0.0789
gamma = 0.1204 ASE = 0.160
Kendall's tau-b = 0.0630 ASE = 0.084
Fisher's exact = 0.707
```

For  $2 \times 2$  tables, both one- and two-sided Fisher’s exact probabilities are displayed; this is true of both `tabulate` and `tabi`. See [Cumulative incidence data](#) and [Case-control data](#) in [ST] `epitab` for more discussion on the relationship between one- and two-sided probabilities.



❑ Technical note

`tabi`, as with all immediate commands, leaves any data in memory undisturbed. With the `replace` option, however, the data in memory are replaced by the data from the table:

```
. tabi 30 18 \ 38 14, replace
```

| row   | col |    | Total |
|-------|-----|----|-------|
|       | 1   | 2  |       |
| 1     | 30  | 18 | 48    |
| 2     | 38  | 14 | 52    |
| Total | 68  | 32 | 100   |

```
Fisher's exact = 0.289
1-sided Fisher's exact = 0.179
. list
```

|    | row | col | pop |
|----|-----|-----|-----|
| 1. | 1   | 1   | 30  |
| 2. | 1   | 2   | 18  |
| 3. | 2   | 1   | 38  |
| 4. | 2   | 2   | 14  |

With this dataset, you could re-create the above table by typing

```
. tabulate row col [freq=pop], exact
```

| row   | col |    | Total |
|-------|-----|----|-------|
|       | 1   | 2  |       |
| 1     | 30  | 18 | 48    |
| 2     | 38  | 14 | 52    |
| Total | 68  | 32 | 100   |

```
Fisher's exact = 0.289
1-sided Fisher's exact = 0.179
```



## tab2

tab2 is a convenience tool. Typing

```
. tab2 myvar thisvar thatvar, chi2
```

is equivalent to typing

```
. tabulate myvar thisvar, chi2
. tabulate myvar thatvar, chi2
. tabulate thisvar thatvar, chi2
```

## Saved results

tabulate, tab2, and tabi save the following in `r()`:

### Scalars

|                          |                                    |                          |                                           |
|--------------------------|------------------------------------|--------------------------|-------------------------------------------|
| <code>r(N)</code>        | number of observations             | <code>r(p_exact)</code>  | Fisher's exact $p$                        |
| <code>r(r)</code>        | number of rows                     | <code>r(chi2_lr)</code>  | likelihood-ratio $\chi^2$                 |
| <code>r(c)</code>        | number of columns                  | <code>r(p_lr)</code>     | significance of likelihood-ratio $\chi^2$ |
| <code>r(chi2)</code>     | Pearson's $\chi^2$                 | <code>r(CramersV)</code> | Cramér's $V$                              |
| <code>r(p)</code>        | significance of Pearson's $\chi^2$ | <code>r(ase_gam)</code>  | ASE of gamma                              |
| <code>r(gamma)</code>    | gamma                              | <code>r(ase_taub)</code> | ASE of $\tau_b$                           |
| <code>r(p1_exact)</code> | one-sided Fisher's exact $p$       | <code>r(taub)</code>     | $\tau_b$                                  |

`r(p1_exact)` is defined only for  $2 \times 2$  tables. Also, the `matrow()`, `matcol()`, and `matcell()` options allow you to obtain the row values, column values, and frequencies, respectively.

## Methods and formulas

tab2 and tabi are implemented as ado-files.

Let  $n_{ij}$ ,  $i = 1, \dots, I$  and  $j = 1, \dots, J$ , be the number of observations in the  $i$ th row and  $j$ th column. If the data are not weighted,  $n_{ij}$  is just a count. If the data are weighted,  $n_{ij}$  is the sum of the weights of all data corresponding to the  $(i, j)$  cell.

Define the row and column marginals as

$$n_{i\cdot} = \sum_{j=1}^J n_{ij} \qquad n_{\cdot j} = \sum_{i=1}^I n_{ij}$$

and let  $n = \sum_i \sum_j n_{ij}$  be the overall sum. Also, define the concordance and discordance as

$$A_{ij} = \sum_{k>i} \sum_{l>j} n_{kl} + \sum_{k<i} \sum_{l<j} n_{kl} \qquad D_{ij} = \sum_{k>i} \sum_{l<j} n_{kl} + \sum_{k<i} \sum_{l>j} n_{kl}$$

along with twice the number of concordances  $P = \sum_i \sum_j n_{ij} A_{ij}$  and twice the number of discordances  $Q = \sum_i \sum_j n_{ij} D_{ij}$ .

The Pearson  $\chi^2$  statistic with  $(I - 1)(J - 1)$  degrees of freedom (so called because it is based on [Pearson \(1900\)](#); see [Conover \[1999, 240\]](#) and [Fienberg \[1980, 9\]](#)) is defined as

$$X^2 = \sum_i \sum_j \frac{(n_{ij} - m_{ij})^2}{m_{ij}}$$

where  $m_{ij} = n_{i.}n_{.j}/n$ .

The likelihood-ratio  $\chi^2$  statistic with  $(I - 1)(J - 1)$  degrees of freedom ([Fienberg 1980, 40](#)) is defined as

$$G^2 = 2 \sum_i \sum_j n_{ij} \ln(n_{ij}/m_{ij})$$

Cramér's  $V$  ([Cramér 1946](#); also see [Agresti \[1984, 23–24\]](#)) is a measure of association designed so that the attainable upper bound is 1. For  $2 \times 2$  tables,  $-1 \leq V \leq 1$ , and otherwise,  $0 \leq V \leq 1$ .

$$V = \begin{cases} (n_{11}n_{22} - n_{12}n_{21})/(n_{1.}n_{2.}n_{.1}n_{.2})^{1/2} & \text{for } 2 \times 2 \\ \{(X^2/n)/\min(I - 1, J - 1)\}^{1/2} & \text{otherwise} \end{cases}$$

Gamma (Goodman and Kruskal [1954, 1959, 1963, 1972](#); also see [Agresti \[1984, 159–161\]](#)) ignores tied pairs and is based only on the number of concordant and discordant pairs of observations,  $-1 \leq \gamma \leq 1$ ,

$$\gamma = (P - Q)/(P + Q)$$

with asymptotic variance

$$16 \sum_i \sum_j n_{ij} (QA_{ij} - PD_{ij})^2 / (P + Q)^4$$

Kendall's  $\tau_b$  ([Kendall 1945](#); also see [Agresti 1984, 161–163](#)),  $-1 \leq \tau_b \leq 1$ , is similar to gamma, except that it uses a correction for ties,

$$\tau_b = (P - Q)/(w_r w_c)^{1/2}$$

with asymptotic variance

$$\frac{\sum_i \sum_j n_{ij} (2w_r w_c d_{ij} + \tau_b v_{ij})^2 - n^3 \tau_b^2 (w_r + w_c)^2}{(w_r w_c)^4}$$

where

$$\begin{aligned}w_r &= n^2 - \sum_i n_{i.}^2 \\w_c &= n^2 - \sum_j n_{.j}^2 \\d_{ij} &= A_{ij} - D_{ij} \\v_{ij} &= n_{i.}w_c + n_{.j}w_r\end{aligned}$$

Fisher's exact test (Fisher 1935; Finney 1948; see Zelterman and Louis [1992, 293–301] for the  $2 \times 2$  case) yields the probability of observing a table that gives at least as much evidence of association as the one actually observed under the assumption of no association. Holding row and column marginals fixed, the hypergeometric probability  $P$  of every possible table  $A$  is computed, and the

$$P = \sum_{T \in A} \Pr(T)$$

where  $A$  is the set of all tables with the same marginals as the observed table,  $T^*$ , such that  $\Pr(T) \leq \Pr(T^*)$ . For  $2 \times 2$  tables, the one-sided probability is calculated by further restricting  $A$  to tables in the same tail as  $T^*$ . The first algorithm extending this calculation to  $r \times c$  tables was Pagano and Halvorsen (1981); the one implemented here is the FEXACT algorithm by Mehta and Patel (1986). This is a search-tree clipping method originally published by Mehta and Patel (1983) with further refinements by Joe (1988) and Clarkson, Fan, and Joe (1993). Fisher's exact test is a permutation test. For more information on permutation tests, see Good (2005 and 2006).

## References

- Agresti, A. 1984. *Analysis of Ordinal Categorical Data*. New York: Wiley.
- Campbell, M. J., D. Machin, and S. J. Walters. 2007. *Medical Statistics: A Textbook for the Health Sciences*. 4th ed. Chichester, UK: Wiley.
- Clarkson, D. B., Y.-A. Fan, and H. Joe. 1993. A remark on Algorithm 643: FEXACT: An algorithm for performing Fisher's exact test in  $r \times c$  contingency tables. *ACM Transactions on Mathematical Software* 19: 484–488.
- Conover, W. J. 1999. *Practical Nonparametric Statistics*. 3rd ed. New York: Wiley.
- Cox, N. J. 1996. [sg57: An immediate command for two-way tables](#). *Stata Technical Bulletin* 33: 7–9. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 140–143. College Station, TX: Stata Press.
- . 1999. [sg113: Tabulation of modes](#). *Stata Technical Bulletin* 50: 26–27. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 180–181. College Station, TX: Stata Press.
- . 2003. [sg113\\_1: Software update: Tabulation of modes](#). *Stata Journal* 3: 211.
- Cramér, H. 1946. *Mathematical Methods of Statistics*. Princeton: Princeton University Press.
- Fienberg, S. E. 1980. *The Analysis of Cross-Classified Categorical Data*. 2nd ed. Cambridge, MA: MIT Press.
- Finney, D. J. 1948. The Fisher–Yates test of significance in  $2 \times 2$  contingency tables. *Biometrika* 35: 145–156.
- Fisher, R. A. 1935. The logic of inductive inference. *Journal of the Royal Statistical Society* 98: 39–82.
- Good, P. I. 2005. *Permutation, Parametric, and Bootstrap Tests of Hypotheses: A Practical Guide to Resampling Methods for Testing Hypotheses*. 3rd ed. New York: Springer.
- . 2006. *Resampling Methods: A Practical Guide to Data Analysis*. 3rd ed. Boston: Birkhäuser.
- Goodman, L. A., and W. H. Kruskal. 1954. Measures of association for cross classifications. *Journal of the American Statistical Association* 49: 732–764.
- . 1959. Measures of association for cross classifications II: Further discussion and references. *Journal of the American Statistical Association* 54: 123–163.

- . 1963. Measures of association for cross classifications III: Approximate sampling theory. *Journal of the American Statistical Association* 58: 310–364.
- . 1972. Measures of association for cross classifications IV: Simplification of asymptotic variances. *Journal of the American Statistical Association* 67: 415–421.
- Harrison, D. A. 2006. [Stata tip 34: Tabulation by listing](#). *Stata Journal* 6: 425–427.
- Jann, B. 2008. [Multinomial goodness-of-fit: Large-sample tests with survey design correction and exact tests for small samples](#). *Stata Journal* 8: 147–169.
- Joe, H. 1988. Extreme probabilities for contingency tables under row and column independence with application to Fisher's exact test. *Communications in Statistics, Theory and Methods* 17: 3677–3685.
- Judson, D. H. 1992. [sg12: Extended tabulate utilities](#). *Stata Technical Bulletin* 10: 22–23. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 140–141. College Station, TX: Stata Press.
- Kendall, M. G. 1945. The treatment of ties in rank problems. *Biometrika* 33: 239–251.
- Mehta, C. R., and N. R. Patel. 1983. A network algorithm for performing Fisher's exact test in  $r \times c$  contingency tables. *Journal of the American Statistical Association* 78: 427–434.
- . 1986. Algorithm 643 FEXACT: A FORTRAN subroutine for Fisher's exact test on unordered  $r \times c$  contingency tables. *ACM Transactions on Mathematical Software* 12: 154–161.
- Newson, R. 2002. [Parameters behind “nonparametric” statistics: Kendall's tau, Somers' D and median differences](#). *Stata Journal* 2: 45–64.
- Pagano, M., and K. T. Halvorsen. 1981. An algorithm for finding the exact significance levels of  $r \times c$  contingency tables. *Journal of the American Statistical Association* 76: 931–934.
- Pearson, K. 1900. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Philosophical Magazine, Series 5* 50: 157–175.
- Weesie, J. 2001. [dm91: Patterns of missing values](#). *Stata Technical Bulletin* 61: 5–7. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 49–51. College Station, TX: Stata Press.
- Wolfe, R. 1999. [sg118: Partitions of Pearson's  \$\chi^2\$  for analyzing two-way tables that have ordered columns](#). *Stata Technical Bulletin* 51: 37–40. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 203–207. College Station, TX: Stata Press.
- Zelterman, D., and T. A. Louis. 1992. Contingency tables in medical studies. In *Medical Uses of Statistics*, 2nd ed, ed. J. C. Bailar III and F. Mosteller, 293–310. Boston: Dekker.

## Also see

- [R] [table](#) — Tables of summary statistics
- [R] [tabstat](#) — Display table of summary statistics
- [R] [tabulate oneway](#) — One-way tables of frequencies
- [R] [tabulate, summarize\(\)](#) — One- and two-way tables of summary statistics
- [D] [collapse](#) — Make dataset of summary statistics
- [ST] [epitab](#) — Tables for epidemiologists
- [SVY] [svy: tabulate oneway](#) — One-way tables for survey data
- [SVY] [svy: tabulate twoway](#) — Two-way tables for survey data
- [XT] [xttab](#) — Tabulate xt data
- [U] [12.6.3 Value labels](#)
- [U] [25 Working with categorical data and factor variables](#)

# Title

tabulate, summarize() — One- and two-way tables of summary statistics

## Syntax

```
tabulate varname1 [varname2] [if] [in] [weight] [, options]
```

| options                          | description                                                                         |
|----------------------------------|-------------------------------------------------------------------------------------|
| Main                             |                                                                                     |
| summarize(varname <sub>3</sub> ) | report summary statistics for varname <sub>3</sub>                                  |
| [no] means                       | include or suppress means                                                           |
| [no] standard                    | include or suppress standard deviations                                             |
| [no] freq                        | include or suppress frequencies                                                     |
| [no] obs                         | include or suppress number of observations                                          |
| no label                         | show numeric codes, not labels                                                      |
| wrap                             | do not break wide tables                                                            |
| missing                          | treat missing values of varname <sub>1</sub> and varname <sub>2</sub> as categories |

by is allowed; see [D] by.  
aweight and fweight are allowed; see [U] 11.1.6 weight.

## Menu

Statistics > Summaries, tables, and tests > Tables > One/two-way table of summary statistics

## Description

tabulate, summarize() produces one- and two-way tables (breakdowns) of means and standard deviations. See [R] tabulate oneway and [R] tabulate twoway for one- and two-way frequency tables. See [R] table for a more flexible command that produces one-, two-, and *n*-way tables of frequencies and a wide variety of summary statistics. table is better, but tabulate, summarize() is faster. Also see [R] tabstat for yet another alternative.

## Options

Main

summarize(varname<sub>3</sub>) identifies the name of the variable for which summary statistics are to be reported. If you do not specify this option, a table of frequencies is produced; see [R] tabulate oneway and [R] tabulate twoway. The description here concerns tabulate when this option is specified.

[no]means includes or suppresses only the means from the table.

The `summarize()` table normally includes the mean, standard deviation, frequency, and, if the data are weighted, number of observations. Individual elements of the table may be included or suppressed by the `[no]means`, `[no]standard`, `[no]freq`, and `[no]obs` options. For example, typing

```
. tabulate category, summarize(myvar) means standard
```

produces a summary table by `category` containing only the means and standard deviations of `myvar`. You could also achieve the same result by typing

```
. tabulate category, summarize(myvar) nofreq
```

`[no]standard` includes or suppresses only the standard deviations from the table; see `[no]means` option above.

`[no]freq` includes or suppresses only the frequencies from the table; see `[no]means` option above.

`[no]obs` includes or suppresses only the reported number of observations from the table. If the data are not weighted, the number of observations is identical to the frequency, and by default only the frequency is reported. If the data are weighted, the frequency refers to the sum of the weights. See `[no]means` option above.

`no label` causes the numeric codes to be displayed rather than the label values.

`wrap` requests that no action be taken on wide tables to make them readable. Unless `wrap` is specified, wide tables are broken into pieces to enhance readability.

`missing` requests that missing values of `varname1` and `varname2` be treated as categories rather than as observations to be omitted from the analysis.

## Remarks

`tabulate` with the `summarize()` option produces one- and two-way tables of summary statistics. When combined with the `by` prefix, it can produce *n*-way tables as well.

Remarks are presented under the following headings:

- One-way tables*
- Two-way tables*

## One-way tables

### ► Example 1

We have data on 74 automobiles. Included in our dataset are the variables `foreign`, which marks domestic and foreign cars, and `mpg`, the car's mileage rating. Typing `tabulate foreign` displays a breakdown of the number of observations we have by the values of the `foreign` variable.

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. tabulate foreign
```

| Car type | Freq. | Percent | Cum.   |
|----------|-------|---------|--------|
| Domestic | 52    | 70.27   | 70.27  |
| Foreign  | 22    | 29.73   | 100.00 |
| Total    | 74    | 100.00  |        |



We discover that we have 52 domestic cars and 22 foreign cars in our dataset. If we add the `summarize(varname)` option, however, `tabulate` produces a table of summary statistics for `varname`:

```
. tabulate foreign, summarize(mpg)
```

| Car type | Summary of Mileage (mpg) |           | Freq. |
|----------|--------------------------|-----------|-------|
|          | Mean                     | Std. Dev. |       |
| Domestic | 19.826923                | 4.7432972 | 52    |
| Foreign  | 24.772727                | 6.6111869 | 22    |
| Total    | 21.297297                | 5.7855032 | 74    |

We also discover that the average gas mileage for domestic cars is about 20 mpg and the average foreign is almost 25 mpg. Overall, the average is 21 mpg in our dataset.



❏ Technical note

We might now wonder if the difference in gas mileage between foreign and domestic cars is statistically significant. We can use the `oneway` command to find out; see [\[R\] oneway](#). To obtain an analysis-of-variance table of `mpg` on `foreign`, we type

```
. oneway mpg foreign
```

| Source         | Analysis of Variance |    |            | F     | Prob > F |
|----------------|----------------------|----|------------|-------|----------|
|                | SS                   | df | MS         |       |          |
| Between groups | 378.153515           | 1  | 378.153515 | 13.18 | 0.0005   |
| Within groups  | 2065.30594           | 72 | 28.6848048 |       |          |
| Total          | 2443.45946           | 73 | 33.4720474 |       |          |

Bartlett's test for equal variances: `chi2(1) = 3.4818 Prob>chi2 = 0.062`

The *F* statistic is 13.18, and the difference between foreign and domestic cars' mileage ratings is significant at the 0.05% level.

There are several ways that we could have statistically compared mileage ratings—see, for instance, [\[R\] anova](#), [\[R\] oneway](#), [\[R\] regress](#), and [\[R\] ttest](#)—but `oneway` seemed the most convenient.



Two-way tables

➤ Example 2

`tabulate`, `summarize` can be used to obtain two-way as well as one-way breakdowns. For instance, we obtained summary statistics on `mpg` decomposed by `foreign` by typing `tabulate foreign, summarize(mpg)`. We can specify up to two variables before the comma:

(Continued on next page)

```
. generate wgtcat = autocode(weight,4,1760,4840)
. tabulate wgtcat foreign, summarize(mpg)

Means, Standard Deviations and Frequencies of Mileage (mpg)
```

| wgtcat | Car type  |           | Total     |
|--------|-----------|-----------|-----------|
|        | Domestic  | Foreign   |           |
| 2530   | 28.285714 | 27.0625   | 27.434783 |
|        | 3.0937725 | 5.9829619 | 5.2295149 |
|        | 7         | 16        | 23        |
| 3300   | 21.75     | 19.6      | 21.238095 |
|        | 2.4083189 | 3.4351128 | 2.7550819 |
|        | 16        | 5         | 21        |
| 4070   | 17.26087  | 14        | 17.125    |
|        | 1.8639497 | 0         | 1.9406969 |
|        | 23        | 1         | 24        |
| 4840   | 14.666667 | .         | 14.666667 |
|        | 3.32666   | .         | 3.32666   |
|        | 6         | 0         | 6         |
| Total  | 19.826923 | 24.772727 | 21.297297 |
|        | 4.7432972 | 6.6111869 | 5.7855032 |
|        | 52        | 22        | 74        |

In addition to the means, standard deviations, and frequencies for each weight–mileage cell, also reported are the summary statistics by weight, by mileage, and overall. For instance, the last row of the table reveals that the average mileage of domestic cars is 19.83 and that of foreign cars is 24.77—domestic cars yield poorer mileage than foreign cars. But we now see that domestic cars yield better gas mileage within weight class—the reason domestic cars yield poorer gas mileage is because they are, on average, heavier.



➤ Example 3

If we do not specify the statistics to be included in a table, `tabulate` reports the mean, standard deviation, and frequency. We can specify the statistics that we want to see using the `means`, `standard`, and `freq` options:

```
. tabulate wgtcat foreign, summarize(mpg) means

Means of Mileage (mpg)
```

| wgtcat | Car type  |           | Total     |
|--------|-----------|-----------|-----------|
|        | Domestic  | Foreign   |           |
| 2530   | 28.285714 | 27.0625   | 27.434783 |
| 3300   | 21.75     | 19.6      | 21.238095 |
| 4070   | 17.26087  | 14        | 17.125    |
| 4840   | 14.666667 | .         | 14.666667 |
| Total  | 19.826923 | 24.772727 | 21.297297 |

When we specify one or more of the `means`, `standard`, and `freq` options, only those statistics are displayed. Thus we could obtain a table containing just the means and standard deviations by typing `means standard` after the `summarize(mpg)` option. We can also suppress selected statistics by placing `no` in front of the option name. Another way of obtaining only the means and standard deviations is to add the `nofreq` option:

```
. tabulate wgtcat foreign, summarize(mpg) nofreq
 Means and Standard Deviations of Mileage (mpg)
```

| wgtcat | Car type  |           | Total     |
|--------|-----------|-----------|-----------|
|        | Domestic  | Foreign   |           |
| 2530   | 28.285714 | 27.0625   | 27.434783 |
|        | 3.0937725 | 5.9829619 | 5.2295149 |
| 3300   | 21.75     | 19.6      | 21.238095 |
|        | 2.4083189 | 3.4351128 | 2.7550819 |
| 4070   | 17.26087  | 14        | 17.125    |
|        | 1.8639497 | 0         | 1.9406969 |
| 4840   | 14.666667 | .         | 14.666667 |
|        | 3.32666   | .         | 3.32666   |
| Total  | 19.826923 | 24.772727 | 21.297297 |
|        | 4.7432972 | 6.6111869 | 5.7855032 |



Also see

- [R] [table](#) — Tables of summary statistics
- [R] [tabstat](#) — Display table of summary statistics
- [R] [tabulate oneway](#) — One-way tables of frequencies
- [R] [tabulate twoway](#) — Two-way tables of frequencies
- [D] [collapse](#) — Make dataset of summary statistics
- [SVY] [svy: tabulate oneway](#) — One-way tables for survey data
- [SVY] [svy: tabulate twoway](#) — Two-way tables for survey data
- [U] [12.6 Dataset, variable, and value labels](#)
- [U] [25 Working with categorical data and factor variables](#)

test — Test linear hypotheses after estimation

Syntax

Basic syntax

`test coeflist`(Syntax 1)

`test exp=exp[=...]`(Syntax 2)

`test [eqno] [: coeflist]`(Syntax 3)

`test [eqno=eqno[=...]] [: coeflist]`(Syntax 4)

`testparm varlist [, equal equation(eqno)]`

Full syntax

`test (spec) [(spec) ...] [, test_options]`

| <i>test_options</i>                              | description                                                                                                                                           |
|--------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| Options                                          |                                                                                                                                                       |
| <code><i>mtest</i>[(<i>opt</i>)]</code>          | test each condition separately                                                                                                                        |
| <code><i>coef</i></code>                         | report estimated constrained coefficients                                                                                                             |
| <code><i>accumulate</i></code>                   | test hypothesis jointly with previously tested hypotheses                                                                                             |
| <code><i>notest</i></code>                       | suppress the output                                                                                                                                   |
| <code><i>common</i></code>                       | test only variables common to all the equations                                                                                                       |
| <code><i>constant</i></code>                     | include the constant in coefficients to be tested                                                                                                     |
| <code><i>nosvyadjust</i></code>                  | carry out the Wald test as $W/k \sim F(k, d)$ ; for use with <code>svy</code> estimation commands                                                     |
| <code><i>minimum</i></code>                      | perform test with the constant, drop terms until the test becomes nonsingular, and test without the constant on the remaining terms; highly technical |
| <sup>†</sup> <code><i>matv1c(matname)</i></code> | save the variance–covariance matrix; programmer’s option                                                                                              |

<sup>†</sup>`matv1c(matname)` does not appear in the dialog box.  
*coeflist* and *varlist* may contain factor variables and time-series operators; see [U] 11.4.3 Factor variables and [U] 11.4.4 Time-series varlists.

- Syntax 1 tests that coefficients are 0.
- Syntax 2 tests that linear expressions are equal.
- Syntax 3 tests that coefficients in *eqno* are 0.
- Syntax 4 tests equality of coefficients between equations.

*spec* is one of

```
coeflist
exp=exp [=exp]
[eqno] [: coeflist]
[eqno1=eqno2 [=...]] [: coeflist]
```

*coeflist* is

```
coef [coef ...]
[eqno]coef [[eqno]coef...]
[eqno] _b[coef] [[eqno] _b[coef]...]
```

*exp* is a linear expression containing

```
coef
_b[coef]
[eqno]coef
[eqno] _b[coef]
```

*eqno* is

```
##
name
```

*coef* identifies a coefficient in the model. *coef* is typically a variable name, a level indicator, an interaction indicator, or an interaction involving continuous variables. Level indicators identify one level of a factor variable and interaction indicators identify one combination of levels of an interaction; see [U] [11.4.3 Factor variables](#). *coef* may contain time-series operators; see [U] [11.4.4 Time-series varlists](#).

Distinguish between `[]`, which are to be typed, and `[]`, which indicate optional arguments.

Although not shown in the syntax diagram, parentheses around *spec* are required only with multiple specifications. Also, the diagram does not show that `test` may be called without arguments to redisplay the results from the last `test`.

`anova` and `manova` allow the `test` syntax above plus more (see [R] [anova postestimation](#) for `test` after `anova`; see [MV] [manova postestimation](#) for `test` after `manova`).

## Menu

### test

Statistics > Postestimation > Tests > Test linear hypotheses

### testparm

Statistics > Postestimation > Tests > Test parameters

## Description

`test` performs Wald tests of simple and composite linear hypotheses about the parameters of the most recently fit model.

`test` supports `svy` estimators, carrying out an adjusted Wald test by default in such cases. `test` can be used with `svy` estimation results, see [SVY] [svy postestimation](#).

`testparm` provides a useful alternative to `test` that permits *varlist* rather than a list of coefficients (which is often nothing more than a list of variables), allowing the use of standard Stata notation, including ‘-’ and ‘\*’, which are given the *expression* interpretation by `test`.

`test` and `testparm` perform Wald tests. For likelihood-ratio tests, see [R] [lrtest](#). For Wald-type tests of nonlinear hypotheses, see [R] [testnl](#). To display estimates for one-dimensional linear or nonlinear expressions of coefficients, see [R] [lincom](#) and [R] [nlcom](#).

See [R] [anova postestimation](#) for additional `test` syntax allowed after `anova`.

See [MV] [manova postestimation](#) for additional `test` syntax allowed after `manova`.

## Options for testparm

`equal` tests that the variables appearing in *varlist*, which also appear in the previously fit model, are equal to each other rather than jointly equal to zero.

`equation(eqno)` is relevant only for multiple-equation models, such as `mvreg`, `mlogit`, and `heckman`.

It specifies the equation for which the all-zero or all-equal hypothesis is tested. `equation(#1)` specifies that the test be conducted regarding the first equation #1. `equation(price)` specifies that the test concern the equation named *price*.

## Options for test

### Options

`mtest[opt]` specifies that tests be performed for each condition separately. *opt* specifies the method for adjusting *p*-values for multiple testing. Valid values for *opt* are

|                         |                             |
|-------------------------|-----------------------------|
| <code>bonferroni</code> | Bonferroni’s method         |
| <code>holm</code>       | Holm’s method               |
| <code>sidak</code>      | Šidák’s method              |
| <code>noadjust</code>   | no adjustment is to be made |

Specifying `mtest` without an argument is equivalent to `mtest(noadjust)`.

`coef` specifies that the constrained coefficients be displayed.

`accumulate` allows a hypothesis to be tested jointly with the previously tested hypotheses.

`notest` suppresses the output. This option is useful when you are interested only in the joint test of several hypotheses, specified in a subsequent call of `test`, `accumulate`.

`common` specifies that when you use the `[eqno1=eqno2[=...]]` form of *spec*, the variables common to the equations *eqno*<sub>1</sub>, *eqno*<sub>2</sub>, etc., be tested. The default action is to complain if the equations have variables not in common.

`constant` specifies that `_cons` be included in the list of coefficients to be tested when using the `[eqno1=eqno2[=...]]` or `[eqno]` forms of *spec*. The default is not to include `_cons`.

`nosvyadjust` is for use with `svy` estimation commands. It specifies that the Wald test be carried out as  $W/k \sim F(k, d)$  rather than as  $(d - k + 1)W/(kd) \sim F(k, d - k + 1)$ , where *k* = the dimension of the test and *d* = the total number of sampled PSUs minus the total number of strata.

minimum is a highly technical option. It first performs the test with the constant added. If this test is singular, coefficients are dropped until the test becomes nonsingular. Then the test without the constant is performed with the remaining terms.

The following option is available with `test` but is not shown in the dialog box:

`matv1c(matname)`, a programmer's option, saves the variance–covariance matrix of the linear combinations involved in the suite of tests. For the test of the linear constraints  $Lb = c$ , `matname` contains  $LV'L'$ , where  $V$  is the estimated variance–covariance matrix of  $b$ .

## Remarks

Remarks are presented under the following headings:

[Introductory examples](#)

[Special syntaxes after multiple-equation estimation](#)

[Constrained coefficients](#)

[Multiple testing](#)

## Introductory examples

`test` performs  $F$  or  $\chi^2$  tests of linear restrictions applied to the most recently fitted model (e.g., `regress` or `svy: regress` in the linear regression case; `logit`, `stcox`, `svy: logit`, ... in the single-equation maximum-likelihood case; and `mlogit`, `mvreg`, `streg`, ... in the multiple-equation maximum-likelihood case). `test` may be used after *any* estimation command, although for maximum likelihood techniques, `test` produces a Wald test that depends only on the estimate of the covariance matrix—you may prefer to use the more computationally expensive likelihood-ratio test; see [\[U\] 20 Estimation and postestimation commands](#) and [\[R\] lrtest](#).

There are several variations on the syntax for `test`. The second syntax,

```
test exp=exp[=...]
```

is allowed after any form of estimation. After fitting a model of `depvar` on `x1`, `x2`, and `x3`, typing `test x1+x2=x3` tests the restriction that the coefficients on `x1` and `x2` sum to the coefficient on `x3`. The expressions can be arbitrarily complicated; for instance, typing `test x1+2*(x2+x3)=x2+3*x3` is the same as typing `test x1+x2=x3`.

As a convenient shorthand, `test` also allows you to specify equality for multiple expressions; for example, `test x1+x2 = x3+x4 = x5+x6` tests that the three specified pairwise sums of coefficients are equal.

`test` understands that when you type `x1`, you are referring to the coefficient on `x1`. You could also more explicitly type `test _b[x1]+_b[x2]=_b[x3]`; or you could `test _coef[x1]+_coef[x2]=_coef[x3]`, or `test [#1]x1+[#1]x2=[#1]x3`, or many other things because there is more than one way to refer to an estimated coefficient; see [\[U\] 13.5 Accessing coefficients and standard errors](#). The shorthand involves less typing. On the other hand, you must be more explicit after estimation of multiple-equation models because there may be more than one coefficient associated with an independent variable. You might type, for instance, `test [#2]x1+[#2]x2=[#2]x3` to test the constraint in equation 2 or, more readably, `test [ford]x1+[ford]x2=[ford]x3`, meaning that Stata will test the constraint on the equation corresponding to `ford`, which might be equation 2. `ford` would be an equation name after, say, `sureg`, or, after `mlogit`, `ford` would be one of the outcomes. For `mlogit`, you could also type `test [2]x1+[2]x2=[2]x3`—note the lack of the `#`—meaning not equation 2, but the equation corresponding to the numeric outcome 2. You can even test constraints across equations: `test [ford]x1+[ford]x2=[buick]x3`.

The syntax

`test coeflist`

is available after all estimation commands and is a convenient way to test that multiple coefficients are zero following estimation. A *coeflist* can simply be a list of variable names,

`test varname [varname ...]`

and it is most often specified that way. After you have fit a model of `depvar` on `x1`, `x2`, and `x3`, typing `test x1 x3` tests that the coefficients on `x1` and `x3` are jointly zero. After multiple-equation estimation, this would test that the coefficients on `x1` and `x3` are zero in all equations that contain them. You can also be more explicit and type, for instance, `test [ford]x1 [ford]x3` to test that the coefficients on `x1` and `x3` are zero in the equation for `ford`.

In the multiple-equation case, there are more alternatives. You could also test that the coefficients on `x1` and `x3` are zero in the equation for `ford` by typing `test [ford]: x1 x3`. You could test that all coefficients except the coefficient on the constant are zero in the equation for `ford` by typing `test [ford]`. You could test that the coefficients on `x1` and `x3` in the equation for `ford` are equal to the corresponding coefficients in the equation corresponding to `buick` by typing `test [ford=buick]: x1 x3`. You could test that all the corresponding coefficients except the constant in three equations are equal by typing `test [ford=buick=volvo]`.

`testparm` is much like the first syntax of `test`. Its usefulness will be demonstrated below.

The examples below use `regress`, but what is said applies equally after any single-equation estimation command (such as `logistic`). It also applies after multiple-equation estimation commands as long as references to coefficients are qualified with an equation name or number in square brackets placed before them. The convenient syntaxes for dealing with tests of many coefficients in multiple-equation models are demonstrated in [Special syntaxes after multiple-equation estimation](#) below.

## ► Example 1

We have 1980 census data on the 50 states recording the birth rate in each state (`brate`), the median age (`medage`), and the region of the country in which each state is located.

The `region` variable is 1 if the state is in the Northeast, 2 if the state is in the North Central, 3 if the state is in the South, and 4 if the state is in the West. We estimate the following regression:



```
. use http://www.stata-press.com/data/r11/census3
(1980 Census data by state)
```

```
. regress brate medage c.medage#c.medage i.region
```

| Source   | SS         | df | MS         |                        |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | 38803.4208 | 5  | 7760.68416 | Number of obs = 50     |  |  |
| Residual | 3393.39921 | 44 | 77.1227094 | F( 5, 44) = 100.63     |  |  |
|          |            |    |            | Prob > F = 0.0000      |  |  |
|          |            |    |            | R-squared = 0.9196     |  |  |
|          |            |    |            | Adj R-squared = 0.9104 |  |  |
| Total    | 42196.82   | 49 | 861.159592 | Root MSE = 8.782       |  |  |

| brate                 | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|-----------------------|-----------|-----------|-------|-------|----------------------|-----------|
| medage                | -109.0958 | 13.52452  | -8.07 | 0.000 | -136.3527            | -81.83892 |
| c.medage#<br>c.medage | 1.635209  | .2290536  | 7.14  | 0.000 | 1.173582             | 2.096836  |
| region                |           |           |       |       |                      |           |
| 2                     | 15.00283  | 4.252067  | 3.53  | 0.001 | 6.433353             | 23.57231  |
| 3                     | 7.366445  | 3.953335  | 1.86  | 0.069 | -.6009775            | 15.33387  |
| 4                     | 21.39679  | 4.650601  | 4.60  | 0.000 | 12.02412             | 30.76946  |
| _cons                 | 1947.611  | 199.8405  | 9.75  | 0.000 | 1544.859             | 2350.363  |

`test` can now be used to perform a variety of statistical tests. Specify the `coeflegend` option with your estimation command to see a legend of the coefficients and how to specify them; see [\[R\] estimation options](#). We can test the hypothesis that the coefficient on `3.region` is zero by typing

```
. test 3.region=0
(1) 3.region = 0
 F(1, 44) = 3.47
 Prob > F = 0.0691
```

The  $F$  statistic with 1 numerator and 44 denominator degrees of freedom is 3.47. The significance level of the test is 6.91%—we can reject the hypothesis at the 10% level but not at the 5% level.

This result from `test` is identical to one presented in the output from `regress`, which indicates that the  $t$  statistic on the `3.region` coefficient is 1.863 and that its significance level is 0.069. The  $t$  statistic presented in the output can be used to test the hypothesis that the corresponding coefficient is zero, although it states the test in slightly different terms. The  $F$  distribution with 1 numerator degree of freedom is, however, identical to the  $t^2$  distribution. We note that  $1.863^2 \approx 3.47$  and that the significance levels in each test agree, although one extra digit is presented by the `test` command.



## □ Technical note

After all estimation commands, including those that use the maximum likelihood method, the test that one variable is zero is identical to that reported by the command's output. The tests are performed in the same way—using the estimated covariance matrix—and are known as Wald tests. If the estimation command reports significance levels and confidence intervals using  $z$  rather than  $t$  statistics, `test` reports results using the  $\chi^2$  rather than the  $F$  statistic.



## ► Example 2

If that were all `test` could do, it would be useless. We can use `test`, however, to perform other tests. For instance, we can `test` the hypothesis that the coefficient on `2.region` is 21 by typing

```
. test 2.region=21
(1) 2.region = 21
 F(1, 44) = 1.99
 Prob > F = 0.1654
```

We find that we cannot reject that hypothesis, or at least we cannot reject it at any significance level below 16.5%.



## ► Example 3

The previous test is useful, but we could almost as easily perform it by hand using the results presented in the regression output if we were well read on our statistics. We could type

```
. display Ftail(1,44,((_coef[2.region]-21)/4.252068)^2)
.16544972
```

So, now let's test something a bit more difficult: whether the coefficient on `2.region` is the same as the coefficient on `4.region`:

```
. test 2.region=4.region
(1) 2.region - 4.region = 0
 F(1, 44) = 2.84
 Prob > F = 0.0989
```

We find that we cannot reject the equality hypothesis at the 5% level, but we can at the 10% level.



## ► Example 4

When we tested the equality of the `2.region` and `4.region` coefficients, Stata rearranged our algebra. When Stata displayed its interpretation of the specified test, it indicated that we were testing whether `2.region minus 4.region` is zero. The rearrangement is innocuous and, in fact, allows Stata to perform much more complicated algebra, for instance,

```
. test 2*(2.region-3*(3.region-4.region))=3.region+2.region+6*(4.region-3.region)
(1) 2.region - 3.region = 0
 F(1, 44) = 5.06
 Prob > F = 0.0295
```

Although we requested what appeared to be a lengthy hypothesis, once Stata simplified the algebra, it realized that all we wanted to do was test whether the coefficient on `2.region` is the same as the coefficient on `3.region`.



## □ Technical note

Stata's ability to simplify and test complex hypotheses is limited to *linear* hypotheses. If you attempt to test a nonlinear hypothesis, you will be told that it is not possible:

```
. test 2.region/3.region=2.region+3.region
not possible with test
r(131);
```

To test a nonlinear hypothesis, see [\[R\] `testnl`](#).



## ► Example 5

The real power of `test` is demonstrated when we test *joint* hypotheses. Perhaps we wish to test whether the region variables, taken as a whole, are significant by testing whether the coefficients on `2.region`, `3.region`, and `4.region` are simultaneously zero. `test` allows us to specify multiple conditions to be tested, each embedded within parentheses.

```
. test (2.region=0) (3.region=0) (4.region=0)
(1) 2.region = 0
(2) 3.region = 0
(3) 4.region = 0
 F(3, 44) = 8.85
 Prob > F = 0.0001
```

`test` displays the set of conditions and reports an  $F$  statistic of 8.85. `test` also reports the degrees of freedom of the test to be 3, the “dimension” of the hypothesis, and the residual degrees of freedom, 44. The significance level of the test is close to 0, so we can strongly reject the hypothesis of no difference between the regions.

An alternative method to specify simultaneous hypotheses uses the convenient shorthand of conditions with multiple equality operators.

```
. test 2.region=3.region=4.region=0
(1) 2.region - 3.region = 0
(2) 2.region - 4.region = 0
(3) 2.region = 0
 F(3, 44) = 8.85
 Prob > F = 0.0001
```

◀

## □ Technical note

Another method to test simultaneous hypotheses is to specify a `test` for each constraint and `accumulate` it with the previous constraints:

```
. test 2.region=0
(1) 2.region = 0
 F(1, 44) = 12.45
 Prob > F = 0.0010

. test 3.region=0, accumulate
(1) 2.region = 0
(2) 3.region = 0
 F(2, 44) = 6.42
 Prob > F = 0.0036

. test 4.region=0, accumulate
(1) 2.region = 0
(2) 3.region = 0
(3) 4.region = 0
 F(3, 44) = 8.85
 Prob > F = 0.0001
```

We tested the hypothesis that the coefficient on `2.region` was zero by typing `test 2.region=0`. We then tested whether the coefficient on `3.region` was also zero by typing `test 3.region=0, accumulate`. The `accumulate` option told Stata that this was not the start of a new test but a continuation of a previous one. Stata responded by showing us the two equations and reporting an  $F$  statistic of 6.42. The significance level associated with those two coefficients being zero is 0.36%.

When we added the last constraint `test 4.region=0, accumulate`, we discovered that the three region variables are significant. If all we wanted was the overall significance and we did not want to bother seeing the interim results, we could have used the `notest` option:

```
. test 2.region=0, notest
(1) 2.region = 0
. test 3.region=0, accumulate notest
(1) 2.region = 0
(2) 3.region = 0
. test 4.region=0, accumulate
(1) 2.region = 0
(2) 3.region = 0
(3) 4.region = 0
 F(3, 44) = 8.85
 Prob > F = 0.0001
```

□

## ► Example 6

Because tests that coefficients are zero are so common in applied statistics, the `test` command has a more convenient syntax to accommodate this case:

```
. test 2.region 3.region 4.region
(1) 2.region = 0
(2) 3.region = 0
(3) 4.region = 0
 F(3, 44) = 8.85
 Prob > F = 0.0001
```

◀

## ► Example 7

We will now show how to use `testparm`. In its first syntax, `test` accepts a list of variable names but not a *varlist*.

```
. test i(2/4).region
i not found
r(111);
```

In the varlist, `i(2/4).region` means all the level variables from `2.region` through `4.region`, yet we received an error. `test` does not actually understand varlists, but `testparm` does. In fact, it understands only varlists.

```
. testparm i(2/4).region
(1) 2.region = 0
(2) 3.region = 0
(3) 4.region = 0
 F(3, 44) = 8.85
 Prob > F = 0.0001
```

That `testparm` accepts varlists has other advantages that do not involve factor variables. Suppose that we have a dataset that has dummy variables `reg2`, `reg3`, and `reg4`, rather than the categorical variable `region`.

```
. use http://www.stata-press.com/data/r11/census4
(birth rate, median age)

. regress brate medage c.medage#c.medage reg2 reg3 reg4
(output omitted)

. test reg2-reg4
- not found
r(111);
```

In a *varlist*, *reg2-reg4* means variables *reg2* and *reg4* and all the variables between, yet we received an error. *test* is confused because the *-* has two meanings: it means subtraction in an expression and “through” in a *varlist*. Similarly, *\** means “any set of characters” in a *varlist* and multiplication in an expression. *testparm* avoids this confusion—it allows only a *varlist*.

```
. testparm reg2-reg4
(1) reg2 = 0
(2) reg3 = 0
(3) reg4 = 0
 F(3, 44) = 8.85
 Prob > F = 0.0001
```

*testparm* has another advantage. We have five variables in our dataset that start with the characters *reg*: *region*, *reg1*, *reg2*, *reg3*, and *reg4*. *reg\** thus means those five variables:

```
. describe reg*
 variable name storage display value
 type format label variable label

region int %8.0g region Census Region
reg1 byte %9.0g
reg2 byte %9.0g
reg3 byte %9.0g
reg4 byte %9.0g
region==NE
region==N Cntrl
region==South
region==West
```

We cannot type *test reg\** because, in an expression, *\** means multiplication, but here is what would happen if we attempted to test all the variables that begin with *reg*:

```
. test region reg1 reg2 reg3 reg4
region not found
r(111);
```

The variable *region* was not included in our model, so it was not found. However, with *testparm*,

```
. testparm reg*
(1) reg2 = 0
(2) reg3 = 0
(3) reg4 = 0
 F(3, 44) = 8.85
 Prob > F = 0.0001
```

That is, *testparm* took *reg\** to mean all variables that start with *reg* that were in our model.



## □ Technical note

Actually, *reg\** means what it always does—all variables in our dataset that begin with *reg*—in this case, *region reg1 reg2 reg3 reg4*. *testparm* just ignores any variables you specify that are not in the model.



## ► Example 8

We just used `test` (`testparm`, actually, but it does not matter) to test the hypothesis that `reg2`, `reg3`, and `reg4` are jointly zero. We can review the results of our last test by typing `test` without arguments:

```
. test
(1) reg2 = 0
(2) reg3 = 0
(3) reg4 = 0
 F(3, 44) = 8.85
 Prob > F = 0.0001
```

◀

## □ Technical note

`test` does not care how we build joint hypotheses; we may freely mix different forms of syntax. (We can even start with `testparm`, but we cannot use it thereafter because it does not have an `accumulate` option.)

Say that we type `test reg2 reg3 reg4` to test that the coefficients on our region dummies are jointly zero. We could then add a fourth constraint, say, that `medage = 100`, by typing `test medage=100, accumulate`. Or, if we had introduced the `medage` constraint first (our first `test` command had been `test medage=100`), we could then add the region dummy test by typing `test reg2 reg3 reg4, accumulate` or `test (reg2=0) (reg3=0) (reg4=0), accumulate`.

Remember that all previous tests are cleared when we do not specify the `accumulate` option. No matter what tests we performed in the past, if we type `test medage c.medage#c.medage`, omitting the `accumulate` option, we would test that `medage` and `c.medage#c.medage` are jointly zero.

□

## ► Example 9

Let's return to our `census3.dta` dataset and test the hypothesis that all the included regions have the *same* coefficient—that the Northeast is significantly different from the rest of the nation:

```
. use http://www.stata-press.com/data/r11/census3
(1980 Census data by state)
. regress brate medage c.medage#c.medage i.region
(output omitted)
. test 2.region=3.region=4.region
(1) 2.region - 3.region = 0
(2) 2.region - 4.region = 0
 F(2, 44) = 8.23
 Prob > F = 0.0009
```

We find that they are not all the same. The syntax `2.region=3.region=4.region` with multiple `=` operators is just a convenient shorthand for typing that the first expression equals the second expression and that the first expression equals the third expression,

```
. test (2.region=3.region) (2.region=4.region)
```

We performed the test for equality of the three regions by imposing two constraints: region 2 has the same coefficient as region 3, and region 2 has the same coefficient as region 4. Alternatively, we could have tested that the coefficients on regions 2 and 3 are the same and that the coefficients on regions 3 and 4 are the same. We would obtain the same results in either case.

To test for equality of the three regions, we might, likely by mistake, type equality constraints for *all* pairs of regions:

```
. test (2.region=3.region) (2.region=4.region) (3.region=4.region)
(1) 2.region - 3.region = 0
(2) 2.region - 4.region = 0
(3) 3.region - 4.region = 0
 Constraint 3 dropped
 F(2, 44) = 8.23
 Prob > F = 0.0009
```

Equality of regions 2 and 3 and of regions 2 and 4, however, implies equality of regions 3 and 4. `test` recognized that the last constraint is implied by the other constraints and hence dropped it.



## □ Technical note

Generally, Stata uses `=` for assignment, as in `gen newvar = exp`, and `==` as the operator for testing equality in expressions. For your convenience, `test` allows both `=` and `==` to be used.



## ▷ Example 10

The test for the equality of the regions is also possible with the `testparm` command. When we include the `equal` option, `testparm` tests that the coefficients of all the variables specified are equal:

```
. testparm i(2/4).region, equal
(1) - 2.region + 3.region = 0
(2) - 2.region + 4.region = 0
 F(2, 44) = 8.23
 Prob > F = 0.0009
```

We can also obtain the equality test by accumulating single equality tests.

```
. test 2.region=3.region, notest
(1) 2.region - 3.region = 0
. test 2.region=4.region, accum
(1) 2.region - 3.region = 0
(2) 2.region - 4.region = 0
 F(2, 44) = 8.23
 Prob > F = 0.0009
```



## □ Technical note

If we specify a set of inconsistent constraints, `test` will tell us by dropping the constraint or constraints that led to the inconsistency. For instance, let's `test` that the coefficients on region 2 and region 4 are the same, add the test that the coefficient on region 2 is 20, and finally add the test that the coefficient on region 4 is 21:

```
. test (2.region=4.region) (2.region=20) (4.region=21)
(1) 2.region - 4.region = 0
(2) 2.region = 20
(3) 4.region = 21
 Constraint 2 dropped
 F(2, 44) = 1.82
 Prob > F = 0.1737
```

`test` informed us that it was dropping constraint 2. All three equations cannot be simultaneously true, so `test` drops whatever it takes to get back to something that makes sense.



### Special syntaxes after multiple-equation estimation

Everything said above about tests after single-equation estimation applies to tests after multiple-equation estimation, as long as you remember to specify the equation name. To demonstrate, let's estimate a seemingly unrelated regression by using `sureg`; see [\[R\] sureg](#).

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. sureg (price foreign mpg displ) (weight foreign length)
Seemingly unrelated regression
```

| Equation | Obs | Parms | RMSE     | "R-sq" | chi2   | P      |
|----------|-----|-------|----------|--------|--------|--------|
| price    | 74  | 3     | 2165.321 | 0.4537 | 49.64  | 0.0000 |
| weight   | 74  | 2     | 245.2916 | 0.8990 | 661.84 | 0.0000 |

|              | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|--------------|-----------|-----------|-------|-------|----------------------|-----------|
| price        |           |           |       |       |                      |           |
| foreign      | 3058.25   | 685.7357  | 4.46  | 0.000 | 1714.233             | 4402.267  |
| mpg          | -104.9591 | 58.47209  | -1.80 | 0.073 | -219.5623            | 9.644042  |
| displacement | 18.18098  | 4.286372  | 4.24  | 0.000 | 9.779842             | 26.58211  |
| _cons        | 3904.336  | 1966.521  | 1.99  | 0.047 | 50.0263              | 7758.645  |
| weight       |           |           |       |       |                      |           |
| foreign      | -147.3481 | 75.44314  | -1.95 | 0.051 | -295.2139            | .517755   |
| length       | 30.94905  | 1.539895  | 20.10 | 0.000 | 27.93091             | 33.96718  |
| _cons        | -2753.064 | 303.9336  | -9.06 | 0.000 | -3348.763            | -2157.365 |

To test the significance of `foreign` in the `price` equation, we could type

```
. test [price]foreign
(1) [price]foreign = 0
 chi2(1) = 19.89
 Prob > chi2 = 0.0000
```

which is the same result reported by `sureg`:  $4.460^2 \approx 19.89$ . To test `foreign` in both equations, we could type

```
. test [price]foreign [weight]foreign
(1) [price]foreign = 0
(2) [weight]foreign = 0
 chi2(2) = 31.61
 Prob > chi2 = 0.0000
```

or

```
. test foreign
(1) [price]foreign = 0
(2) [weight]foreign = 0
 chi2(2) = 31.61
 Prob > chi2 = 0.0000
```

This last syntax—typing the variable name by itself—tests the coefficients in all equations in which they appear. The variable `length` appears in only the `weight` equation, so typing

```
. test length
(1) [weight]length = 0
 chi2(1) = 403.94
 Prob > chi2 = 0.0000
```



yields the same result as typing `test [weight]length`. We may also specify a linear expression rather than a list of coefficients:

```
. test mpg=displ
(1) [price]mpg - [price]displ = 0
 chi2(1) = 4.85
 Prob > chi2 = 0.0277
```

or

```
. test [price]mpg = [price]displ
(1) [price]mpg - [price]displ = 0
 chi2(1) = 4.85
 Prob > chi2 = 0.0277
```

A variation on this syntax can be used to test cross-equation constraints:

```
. test [price]foreign = [weight]foreign
(1) [price]foreign - [weight]foreign = 0
 chi2(1) = 23.07
 Prob > chi2 = 0.0000
```

Typing an equation name in square brackets by itself tests all the coefficients except the intercept in that equation:

```
. test [price]
(1) [price]foreign = 0
(2) [price]mpg = 0
(3) [price]displacement = 0
 chi2(3) = 49.64
 Prob > chi2 = 0.0000
```

Typing an equation name in square brackets, a colon, and a list of variable names tests those variables in the specified equation:

```
. test [price]: foreign displ
(1) [price]foreign = 0
(2) [price]displacement = 0
 chi2(2) = 25.19
 Prob > chi2 = 0.0000
```

`test [eqname1=eqname2]` tests that all the coefficients in the two equations are equal. We cannot use that syntax here because there are different variables in the model:

```
. test [price=weight]
variables differ between equations
(to test equality of coefficients in common, specify option -common-)
r(111);
```

The common option specifies a test of the equality coefficients common to the equations `price` and `weight`,

```
. test [price=weight], common
(1) [price]foreign - [weight]foreign = 0
 chi2(1) = 23.07
 Prob > chi2 = 0.0000
```

By default, `test` does not include the constant, the coefficient of the constant variable `_cons`, in the test. The `cons` option specifies that the constant be included.

```
. test [price=weight], common cons
(1) [price]foreign - [weight]foreign = 0
(2) [price]_cons - [weight]_cons = 0
 chi2(2) = 51.23
 Prob > chi2 = 0.0000
```

We can also use a modification of this syntax with the model if we also type a colon and the names of the variables we want to test:

```
. test [price=weight]: foreign
(1) [price]foreign - [weight]foreign = 0
 chi2(1) = 23.07
 Prob > chi2 = 0.0000
```

We have only one variable in common between the two equations, but if there had been more, we could have listed them.

Finally, a simultaneous test of multiple constraints may be specified just as after single-equation estimation.

```
. test ([price]: foreign) ([weight]: foreign)
(1) [price]foreign = 0
(2) [weight]foreign = 0
 chi2(2) = 31.61
 Prob > chi2 = 0.0000
```

`test` can also test for equality of coefficients across more than two equations. For instance, `test [eq1=eq2=eq3]` specifies a test that the coefficients in the three equations `eq1`, `eq2`, and `eq3` are equal. This requires that the same variables be included in the three equations. If some variables are entered only in some of the equations, you can type `test [eq1=eq2=eq3], common` to test that the coefficients of the variables common to all three equations are equal. Alternatively, you can explicitly list the variables for which equality of coefficients across the equations is to be tested. For instance, `test [eq1=eq2=eq3]: time money` tests that the coefficients of the variables `time` and `money` do not differ between the equations.

## □ Technical note

`test [eq1=eq2=eq3], common` tests the equality of the coefficients common to all equations, but it does *not* test the equality of all common coefficients. Consider the case where

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>eq1</code> | contains the variables <code>var1 var2 var3</code> |
| <code>eq2</code> | contains the variables <code>var1 var2 var4</code> |
| <code>eq3</code> | contains the variables <code>var1 var3 var4</code> |

Obviously, only `var1` is common to all three equations. Thus `test [eq1=eq2=eq3], common` tests that the coefficients of `var1` do not vary across the equations, so it is equivalent to `test [eq1=eq2=eq3]: var1`. To perform a test of the coefficients of variables common to two equations, you could explicitly list the constraints to be tested,

```
. test ([eq1=eq2=eq3]:var1) ([eq1=eq2]:var2) ([eq1=eq3]:var3) ([eq2=eq3]:var4)
```

or use `test` with the `accumulate` option, and maybe also with the `notest` option, to form the appropriate joint hypothesis:

```
. test [eq1=eq2], common notest
. test [eq1=eq3], common accumulate notest
. test [eq2=eq3], common accumulate
```

□

## Constrained coefficients

If the test indicates that the data do not allow you to conclude that the constraints are not satisfied, you may want to inspect the constrained coefficients. The `coef` option specified that the constrained results, estimated by GLS, are shown.

```
. test [price=weight], common coef
(1) [price]foreign - [weight]foreign = 0
 chi2(1) = 23.07
 Prob > chi2 = 0.0000
```

Constrained coefficients

|              | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|--------------|-----------|-----------|-------|-------|----------------------|-----------|
| price        |           |           |       |       |                      |           |
| foreign      | -216.4015 | 74.06083  | -2.92 | 0.003 | -361.558             | -71.2449  |
| mpg          | -121.5717 | 58.36972  | -2.08 | 0.037 | -235.9742            | -7.169116 |
| displacement | 7.632566  | 3.681114  | 2.07  | 0.038 | .4177148             | 14.84742  |
| _cons        | 7312.856  | 1834.034  | 3.99  | 0.000 | 3718.215             | 10907.5   |
| weight       |           |           |       |       |                      |           |
| foreign      | -216.4015 | 74.06083  | -2.92 | 0.003 | -361.558             | -71.2449  |
| length       | 30.34875  | 1.534815  | 19.77 | 0.000 | 27.34057             | 33.35693  |
| _cons        | -2619.719 | 302.6632  | -8.66 | 0.000 | -3212.928            | -2026.51  |

The constrained coefficient of `foreign` is  $-216.40$  with standard error  $74.06$  in equations `price` and `weight`. The other coefficients and their standard errors are affected by imposing the equality constraint of the two coefficients of `foreign` because the unconstrained estimates of these two coefficients were correlated with the estimates of the other coefficients.

## □ Technical note

The two-step constrained coefficients  $b_c$  displayed by `test`, `coef` are asymptotically equivalent to the one-stage constrained estimates that are computed by specifying the constraints during estimation using the `constraint()` option of estimation commands (Gourieroux and Monfort 1995, chap. 10). Generally, one-step constrained estimates have better small-sample properties. For inspection and interpretation, however, two-step constrained estimates are a convenient alternative. Moreover, some estimation commands (e.g., `stcox`, many `xt` estimators) do not have a `constraint()` option.

□

## Multiple testing

When performing the test of a joint hypothesis, you might want to inspect the underlying 1-degree-of-freedom hypotheses. Which constraint “is to blame”? `test` displays the univariate as well as the simultaneous test if the `mtest` option is specified. For example,

```
. test [price=weight], common cons mtest
(1) [price]foreign - [weight]foreign = 0
(2) [price]_cons - [weight]_cons = 0
```

|     | chi2  | df | p        |
|-----|-------|----|----------|
| (1) | 23.07 | 1  | 0.0000 # |
| (2) | 11.17 | 1  | 0.0008 # |
| all | 51.23 | 2  | 0.0000   |

# unadjusted p-values

Both coefficients seem to contribute to the highly significant result. The 1-degree-of-freedom test shown here is identical to those if `test` had been invoked to test just this simple hypotheses. There is, of course, a real risk in inspecting these simple hypotheses. Especially in high-dimensional hypotheses, you may easily find one hypothesis that happens to be significant. Multiple testing procedures are designed to provide some safeguard against this risk. *p*-values of the univariate hypotheses are modified so that the probability of falsely rejecting one of the null hypotheses is bounded. `test` provides the methods based on Bonferroni, Šidák, and Holm.

```
. test [price=weight], common cons mtest(b)
(1) [price]foreign - [weight]foreign = 0
(2) [price]_cons - [weight]_cons = 0
```

|     | chi2  | df | p        |
|-----|-------|----|----------|
| (1) | 23.07 | 1  | 0.0000 # |
| (2) | 11.17 | 1  | 0.0017 # |
| all | 51.23 | 2  | 0.0000   |

# Bonferroni adjusted p-values

## Saved results

`test` and `testparm` save the following in `r()`:

Scalars

|                           |                                         |                      |                                            |
|---------------------------|-----------------------------------------|----------------------|--------------------------------------------|
| <code>r(p)</code>         | two-sided <i>p</i> -value               | <code>r(chi2)</code> | $\chi^2$                                   |
| <code>r(F)</code>         | <i>F</i> statistic                      | <code>r(ss)</code>   | sum of squares (test)                      |
| <code>r(df)</code>        | test constraints degrees of freedom     | <code>r(rss)</code>  | residual sum of squares                    |
| <code>r(df_r)</code>      | residual degrees of freedom             | <code>r(drop)</code> | 1 if constraints were dropped, 0 otherwise |
| <code>r(dropped_i)</code> | index of <i>i</i> th constraint dropped |                      |                                            |

Macros

|                         |                                           |
|-------------------------|-------------------------------------------|
| <code>r(mtmetho)</code> | method of adjustment for multiple testing |
|-------------------------|-------------------------------------------|

Matrices

|                       |                       |
|-----------------------|-----------------------|
| <code>r(mtest)</code> | multiple test results |
|-----------------------|-----------------------|

`r(ss)` and `r(rss)` are defined only when `test` is used for testing effects after `anova`.

## Methods and formulas

`test` and `testparm` are implemented as ado-files.

`test` and `testparm` perform Wald tests. Let the estimated coefficient vector be  $\mathbf{b}$  and the estimated variance–covariance matrix be  $\mathbf{V}$ . Let  $\mathbf{Rb} = \mathbf{r}$  denote the set of  $q$  linear hypotheses to be tested jointly.

The Wald test statistic is (Judge et al. 1985, 20–28)

$$W = (\mathbf{Rb} - \mathbf{r})'(\mathbf{RVR}')^{-1}(\mathbf{Rb} - \mathbf{r})$$

If the estimation command reports its significance levels using  $Z$  statistics, a chi-squared distribution with  $q$  degrees of freedom,

$$W \sim \chi_q^2$$

is used for computation of the significance level of the hypothesis test.

If the estimation command reports its significance levels using  $t$  statistics with  $d$  degrees of freedom, an  $F$  statistic,

$$F = \frac{1}{q}W$$

is computed, and an  $F$  distribution with  $q$  numerator degrees of freedom and  $d$  denominator degrees of freedom computes the significance level of the hypothesis test.

The two-step constrained estimates  $b_c$  displayed by `test` with the `coef` option are the GLS estimates of the unconstrained estimates  $b$  subject to the specified constraints  $\mathbf{Rb} = \mathbf{c}$  (Gourieroux and Monfort 1995, chap. 10),

$$\mathbf{b}_c = \mathbf{b} - \mathbf{R}'(\mathbf{RVR}')^{-1}\mathbf{R}(\mathbf{Rb} - \mathbf{r})$$

with variance–covariance matrix

$$\mathbf{V}_c = \mathbf{V} - \mathbf{VR}'(\mathbf{RVR}')^{-1}\mathbf{RV}$$

If `test` displays a Wald test for joint (simultaneous) hypotheses, it can also display all 1-degree-of-freedom tests, with  $p$ -values adjusted for multiple testing. Let  $p_1, p_2, \dots, p_k$  be the unadjusted  $p$ -values of these 1-degree-of-freedom tests. The Bonferroni-adjusted  $p$ -values are defined as  $p_i^b = \min(1, kp_i)$ . The Šidák-adjusted  $p$ -values are  $p_i^s = 1 - (1 - p_i)^k$ . Holm's method for adjusting  $p$ -values is defined as  $p_i^h = \min(1, k_i p_i)$ , where  $k_i$  is the number of  $p$ -values at least as large as  $p_i$ . Note that  $p_i^h < p_i^b$ , reflecting that Holm's method is strictly less conservative than the widely used Bonferroni method.

If `test` is used after a `svy` command, it carries out an adjusted Wald test—this adjustment should not be confused with the adjustment for multiple testing. Both adjustments may actually be combined. Specifically, the survey adjustment uses an approximate  $F$  statistic  $(d - k + 1)W/(kd)$ , where  $W$  is the Wald test statistic,  $k$  is the dimension of the hypothesis test, and  $d$  is the total number of sampled PSUs minus the total number of strata. Under the null hypothesis,  $(d - k + 1)F/(kd) \sim F(k, d - k + 1)$ , where  $F(k, d - k + 1)$  is an  $F$  distribution with  $k$  numerator degrees of freedom and  $d - k + 1$  denominator degrees of freedom. If `svynoadjust` is specified, the  $p$ -value is computed using  $W/k \sim F(k, d)$ .

See Korn and Graubard (1990) for a detailed description of the Bonferroni adjustment technique and for a discussion of the relative merits of it and of the adjusted and unadjusted Wald tests.

## Acknowledgment

The `svy` adjustment code was adopted from another command developed in collaboration with John L. Eltinge, Bureau of Labor Statistics.

## References

- Beale, E. M. L. 1960. Confidence regions in non-linear estimation. *Journal of the Royal Statistical Society, Series B* 22: 41–88.
- Eltinge, J. L., and W. M. Sribney. 1996. [svy5: Estimates of linear combinations and hypothesis tests for survey data](#). *Stata Technical Bulletin* 31: 31–42. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 246–259. College Station, TX: Stata Press.
- Gourieroux, C., and A. Monfort. 1995. *Statistics and Econometric Models, Vol 1: General Concepts, Estimation, Prediction, and Algorithms*. Trans. Q. Vuong. Cambridge: Cambridge University Press.
- Holm, S. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics* 6: 65–70.
- Judge, G. G., W. E. Griffiths, R. C. Hill, H. Lütkepohl, and T.-C. Lee. 1985. *The Theory and Practice of Econometrics*. 2nd ed. New York: Wiley.
- Korn, E. L., and B. I. Graubard. 1990. Simultaneous testing of regression coefficients with complex survey data: Use of Bonferroni *t* statistics. *American Statistician* 44: 270–276.
- Weesie, J. 1999. [sg100: Two-stage linear constrained estimation](#). *Stata Technical Bulletin* 47: 24–30. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 217–225. College Station, TX: Stata Press.

## Also see

- [R] [anova](#) — Analysis of variance and covariance
- [R] [anova postestimation](#) — Postestimation tools for anova
- [R] [lincom](#) — Linear combinations of estimators
- [R] [lrtest](#) — Likelihood-ratio test after estimation
- [R] [nestreg](#) — Nested model statistics
- [R] [nlcom](#) — Nonlinear combinations of estimators
- [R] [testnl](#) — Test nonlinear hypotheses after estimation
- [U] [13.5 Accessing coefficients and standard errors](#)
- [U] [20 Estimation and postestimation commands](#)

Syntax

```
testnl exp=exp [=exp...] [, options]

testnl (exp=exp [=exp...]) [(exp=exp [=exp...]) ...] [, options]
```

| options                      | description                                                                                      |
|------------------------------|--------------------------------------------------------------------------------------------------|
| <u>m</u> test [ <i>opt</i> ] | test each condition separately                                                                   |
| <u>n</u> osv <u>y</u> adjust | carry out the Wald test as $W/k \sim F(k,d)$ ; for use with <code>svy</code> estimation commands |
| <u>i</u> terate( <i>#</i> )  | use maximum <i>#</i> of iterations to find the optimal step size                                 |

The second syntax means that if more than one constraint is specified, each must be surrounded by parentheses.

Menu

Statistics > Postestimation > Tests > Test nonlinear hypotheses

Description

`testnl` tests (linear or nonlinear) hypotheses about the estimated parameters from the most recently fitted model.

`testnl` produces Wald-type tests of smooth nonlinear (or linear) hypotheses about the estimated parameters from the most recently fitted model. The *p*-values are based on the “delta method”, an approximation appropriate in large samples.

`testnl` can be used with `svy` estimation results; see [\[SVY\] svy postestimation](#).

The format (*exp*<sub>1</sub>=*exp*<sub>2</sub>=*exp*<sub>3</sub>...) for a simultaneous-equality hypothesis is just a convenient shorthand for a list (*exp*<sub>1</sub>=*exp*<sub>2</sub>) (*exp*<sub>1</sub>=*exp*<sub>3</sub>), etc.

`testnl` may also be used to test linear hypotheses. `test` is faster if you want to test only linear hypotheses. `testnl` is the only option for testing linear and nonlinear hypotheses simultaneously.

Options

`mtest` [*opt*] specifies that tests be performed for each condition separately. *opt* specifies the method for adjusting *p*-values for multiple testing. Valid values for *opt* are

|                    |                             |
|--------------------|-----------------------------|
| <u>b</u> onferroni | Bonferroni’s method         |
| <u>h</u> olm       | Holm’s method               |
| <u>s</u> idak      | Šidák’s method              |
| <u>n</u> oadjust   | no adjustment is to be made |

Specifying `mtest` without an argument is equivalent to specifying `mtest(noadjust)`.

`nosvyadjust` is for use with `svy` estimation commands. It specifies that the Wald test be carried out as  $W/k \sim F(k, d)$  rather than as  $(d - k + 1)W/(kd) \sim F(k, d - k + 1)$ , where  $k$  = the dimension of the test and  $d$  = the total number of sampled PSUs minus the total number of strata.

`iterate(#)` specifies the maximum number of iterations used to find the optimal step size in the calculation of numerical derivatives of the test expressions. By default, the maximum number of iterations is 100, but convergence is usually achieved after only a few iterations. You should rarely have to use this option.

## Remarks

Remarks are presented under the following headings:

- [Introduction](#)
- [Using `testnl` to perform linear tests](#)
- [Specifying constraints](#)
- [Dropped constraints](#)
- [Output](#)
- [Multiple constraints](#)
- [Manipulability](#)

## Introduction

### ► Example 1

We have just estimated the parameters of an earnings model on cross-sectional time-series data using one of Stata’s more sophisticated estimators:

```
. use http://www.stata-press.com/data/r11/earnings
(NLS Women 14-24 in 1968)

. xtgee ln_w grade age c.age#c.age, corr(exchangeable) nolog

GEE population-averaged model
Group variable: idcode Number of obs = 1326
Link: identity Number of groups = 269
Family: Gaussian Obs per group: min = 1
Correlation: exchangeable avg = 4.9
 max = 9
Scale parameter: .0976738 Wald chi2(3) = 327.33
 Prob > chi2 = 0.0000
```

| ln_wage     | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|-------------|-----------|-----------|-------|-------|----------------------|-----------|
| grade       | .0749686  | .0066111  | 11.34 | 0.000 | .062011              | .0879261  |
| age         | .1080806  | .0235861  | 4.58  | 0.000 | .0618526             | .1543086  |
| c.age#c.age | -.0016253 | .0004739  | -3.43 | 0.001 | -.0025541            | -.0006966 |
| _cons       | -.8788933 | .2830899  | -3.10 | 0.002 | -1.433739            | -.3240473 |

An implication of this model is that peak earnings occur at age  $-\_b[\text{age}]/(2*\_b[\text{c.age}\#\text{c.age}])$ , which here is equal to 33.2. Say that we have a theory that peak earnings should occur at age  $16 + 1/\_b[\text{grade}]$ .

```
. testnl -_b[age]/(2*_b[c.age#c.age]) = 16 + 1/_b[grade]
(1) -_b[age]/(2*_b[c.age#c.age]) = 16 + 1/_b[grade]
 chi2(1) = 1.71
 Prob > chi2 = 0.1914
```



These data do not reject our theory.



## Using testnl to perform linear tests

`testnl` may be used to test linear constraints, but `test` is faster; see [R] [test](#). You could type

```
. testnl _b[x4] = _b[x1]
```

but it would take less computer time if you typed

```
. test _b[x4] = _b[x1]
```

## Specifying constraints

The constraints to be tested can be formulated in many different ways. You could type

```
. testnl _b[mpg]*_b[weight] = 1
```

or

```
. testnl _b[mpg] = 1/_b[weight]
```

or you could express the constraint any other way you wished. (To say that `testnl` allows constraints to be specified in different ways does not mean that the test itself does not depend on the formulation. This point is briefly discussed later.) In formulating the constraints, you must, however, exercise one caution: users of `test` often refer to the coefficient on a variable by specifying the variable name. For example,

```
. test mpg = 0
```

More formally, they should type

```
. test _b[mpg] = 0
```

but `test` allows the `_b[]` surrounding the variable name to be omitted. `testnl` does not allow this shorthand. Typing

```
. testnl mpg=0
```

specifies the constraint that the value of variable `mpg` in the first observation is zero. If you make this mistake, sometimes `testnl` will catch it:

```
. testnl mpg=0
equation (1) contains reference to X rather than _b[X]
r(198);
```

In other cases, `testnl` may not catch the mistake; then the constraint will be dropped because it does not make sense:

```
. testnl mpg=0
Constraint (1) dropped
```

(There are reasons other than this for constraints being dropped.) The worst case, however, is

```
. testnl _b[weight]*mpg = 1
```

when what you mean is not that `_b[weight]` equals the reciprocal of the value of `mpg` in the first observation, but rather that

```
. testnl _b[weight]*_b[mpg] = 1
```

Sometimes this mistake will be caught by the “contains reference to X rather than `_b[X]`” error, and sometimes it will not. Be careful.

`testnl`, like `test`, can be used after any Stata estimation command, including the survey estimators. When you use it after a multiple-equation command, such as `mlogit` or `heckman`, you refer to coefficients by using Stata’s standard syntax: `[eqname]_b[varname]`.

Stata’s single-equation estimation output looks like this:

|        | Coef  | ... |                              |
|--------|-------|-----|------------------------------|
| weight | 12.27 | ... | <- coefficient is _b[weight] |
| mpg    | 3.21  | ... |                              |

Stata’s multiple-equation output looks like this:

|      | Coef   | ...   |                                    |
|------|--------|-------|------------------------------------|
| cat1 |        | ...   | <- coefficient is [cat1]_b[weight] |
|      | weight | 12.27 |                                    |
|      | mpg    | 3.21  |                                    |
| 8    |        | ...   | <- coefficient is [8]_b[weight]    |
|      | weight | 5.83  |                                    |
|      | mpg    | 7.43  |                                    |

Dropped constraints

`testnl` automatically drops constraints when

- They are nonbinding, e.g., `_b[mpg]=_b[mpg]`. More subtle cases include

```
_b[mpg]*_b[weight] = 4
_b[weight] = 2
_b[mpg] = 2
```

In this example, the third constraint is nonbinding because it is implied by the first two.

- They are contradictory, e.g., `_b[mpg]=2` and `_b[mpg]=3`. More subtle cases include

```
_b[mpg]*_b[weight] = 4
_b[weight] = 2
_b[mpg] = 3
```

The third constraint contradicts the first two.

## Output

testnl reports the constraints being tested followed by an  $F$  or a  $\chi^2$  test:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. regress price mpg weight c.weight#c.weight foreign
(output omitted)

. testnl (39*_b[mpg]^2 = _b[foreign]) (_b[mpg]/_b[weight] = 4)
(1) 39*_b[mpg]^2 = _b[foreign]
(2) _b[mpg]/_b[weight] = 4
 F(2, 69) = 0.08
 Prob > F = 0.9195

. logit foreign price weight mpg
(output omitted)

. testnl (45*_b[mpg]^2 = _b[price]) (_b[mpg]/_b[weight] = 4)
(1) 45*_b[mpg]^2 = _b[price]
(2) _b[mpg]/_b[weight] = 4
 chi2(2) = 2.44
 Prob > chi2 = 0.2946
```

## Multiple constraints

### ► Example 2

We illustrate the simultaneous test of a series of constraints using simulated data on labor-market promotion in a given year. We fit a probit model with separate effects for education, experience, and experience-squared for men and women.

```
. use http://www.stata-press.com/data/r11/promotion
. probit promo male male#c.(yedu yexp yexp2), nolog

Probit regression Number of obs = 775
 LR chi2(7) = 424.42
 Prob > chi2 = 0.0000
Log likelihood = -245.42768 Pseudo R2 = 0.4637
```

| promo        | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|--------------|-----------|-----------|-------|-------|----------------------|-----------|
| male         | .6489974  | .203739   | 3.19  | 0.001 | .2496763             | 1.048318  |
| male#c.yedu  |           |           |       |       |                      |           |
| 0            | .9730237  | .1056136  | 9.21  | 0.000 | .7660248             | 1.180023  |
| 1            | 1.390517  | .1527288  | 9.10  | 0.000 | 1.091174             | 1.68986   |
| male#c.yexp  |           |           |       |       |                      |           |
| 0            | .4559544  | .0901169  | 5.06  | 0.000 | .2793285             | .6325803  |
| 1            | 1.422539  | .1544255  | 9.21  | 0.000 | 1.11987              | 1.725207  |
| male#c.yexp2 |           |           |       |       |                      |           |
| 0            | -.1027149 | .0573059  | -1.79 | 0.073 | -.2150325            | .0096026  |
| 1            | -.3749457 | .1160113  | -3.23 | 0.001 | -.6023236            | -.1475677 |
| _cons        | .9872018  | .1148215  | 8.60  | 0.000 | .7621559             | 1.212248  |

Note: 1 failure and 2 successes completely determined.

The effects of human capital seem to differ between men and women. A formal test confirms this.

```
. test (yedu#0.male = yedu#1.male) (yexp#0.male = yexp#1.male)
> (yexp2#0.male = yexp2#1.male)
(1) [promo]0b.male#c.yedu - [promo]1.male#c.yedu = 0
(2) [promo]0b.male#c.yexp - [promo]1.male#c.yexp = 0
(3) [promo]0b.male#c.yexp2 - [promo]1.male#c.yexp2 = 0

 chi2(3) = 35.43
 Prob > chi2 = 0.0000
```

How do we interpret this gender difference? It has repeatedly been stressed (see, for example, [Long \[1997, 47–50\]](#); [Allison \[1999\]](#)) that comparison of groups in binary-response models, and similarly in other latent-variable models, is hampered by an identification problem: with  $\beta$  the regression coefficients for the latent variable and  $\sigma$  the standard deviation of the latent residual, only the  $\beta/\sigma$  are identified. In fact, in terms of the latent regression, the probit coefficients should be interpreted as  $\beta/\sigma$ , not as the  $\beta$ . If we cannot claim convincingly that the residual standard deviation  $\sigma$  does not vary between the sexes, equality of the regression coefficients  $\beta$  implies that the coefficients of the probit model for men and women are *proportional* but not necessarily equal. This is a nonlinear hypothesis in terms of the probit coefficients, not a linear one.

```
. testnl _b[yedu#1.male]/_b[yedu#0.male] = _b[yexp#1.male]/_b[yexp#0.male]
> = _b[yexp2#1.male]/_b[yexp2#0.male]
(1) _b[yedu#1.male]/_b[yedu#0.male] = _b[yexp#1.male]/_b[yexp#0.male]
(2) _b[yedu#1.male]/_b[yedu#0.male] = _b[yexp2#1.male]/_b[yexp2#0.male]

 chi2(2) = 9.21
 Prob > chi2 = 0.0100
```

We conclude that we find fairly strong evidence against the proportionality of the coefficients, and hence we have to conclude that success in the labor market is produced in different ways by men and women. (But remember, these were simulated data.)

◀

## ► Example 3

The syntax for specifying the equality of multiple expressions is just a convenient shorthand for specifying a series of constraints, namely, that the first expression equals the second expression, the first expression also equals the third expression, etc. The Wald test performed and the output of `testnl` are the same whether we use the shorthand or we specify the series of constraints. The lengthy specification as a series of constraints can be simplified using the continuation symbols `///`.

```
. testnl (_b[yedu#1.male]/_b[yedu#0.male] = ///
 _b[yexp#1.male]/_b[yexp#0.male]) ///
 (_b[yedu#1.male]/_b[yedu#0.male] = ///
 _b[yexp2#1.male]/_b[yexp2#0.male])
(1) _b[yedu#1.male]/_b[yedu#0.male] = _b[yexp#1.male]/_b[yexp#0.male]
(2) _b[yedu#1.male]/_b[yedu#0.male] = _b[yexp2#1.male]/_b[yexp2#0.male]

 chi2(2) = 9.21
 Prob > chi2 = 0.0100
```

Having established differences between men and women, we would like to do multiple testing between the ratios. Because we did not specify hypotheses in advance, we prefer to adjust the  $p$ -values of tests using, here, Bonferroni's method.

```
. testnl _b[yedu#1.male]/_b[yedu#0.male] = ///
 _b[yexp#1.male]/_b[yexp#0.male] = ///
 _b[yexp2#1.male]/_b[yexp2#0.male], mtest(b)
(1) _b[yedu#1.male]/_b[yedu#0.male] = _b[yexp#1.male]/_b[yexp#0.male]
(2) _b[yedu#1.male]/_b[yedu#0.male] = _b[yexp2#1.male]/_b[yexp2#0.male]
```

|     | chi2 | df | p        |
|-----|------|----|----------|
| (1) | 6.89 | 1  | 0.0173 # |
| (2) | 0.93 | 1  | 0.6713 # |
| all | 9.21 | 2  | 0.0100   |

# Bonferroni adjusted p-values

◀

## Manipulability

Although `testnl` allows you to specify constraints in different ways that are mathematically equivalent, as noted above, this does not mean that the tests are the same. This difference is known as the manipulability of the Wald test for nonlinear hypotheses; also see [R] [boxcox](#). The test might even be significant for one formulation but not significant for another formulation that is mathematically equivalent. Trying out different specifications to find a formulation with the desired  $p$ -value is totally inappropriate, though it may actually be fun to try. There is no variance under representation because the nonlinear Wald test is actually a standard Wald test for a linearization of the constraint, which depends on the particular specification. We note that the likelihood-ratio test is not manipulable in this sense.

From a statistical point of view, it is best to choose a specification of the constraints that is as linear as possible. Doing so usually improves the accuracy of the approximation of the null-distribution of the test by a  $\chi^2$  or an  $F$  distribution. The example above used the nonlinear Wald test to test whether the coefficients of human capital variables for men were proportional to those of women. A specification of proportionality of coefficients in terms of ratios of coefficients is fairly nonlinear if the coefficients in the denominator are close to 0. A more linear version of the test results from a bilinear formulation. Thus instead of

```
. testnl _b[yedu#1.male]/_b[yedu#0.male] = _b[yexp#1.male]/_b[yexp#0.male]
(1) _b[yedu#1.male]/_b[yedu#0.male] = _b[yexp#1.male]/_b[yexp#0.male]
 chi2(1) = 6.89
 Prob > chi2 = 0.0087
```

perhaps

```
. testnl _b[yedu#1.male]*_b[yexp#0.male] = _b[yedu#0.male]*_b[yexp#1.male]
(1) _b[yedu#1.male]*_b[yexp#0.male] = _b[yedu#0.male]*_b[yexp#1.male]
 chi2(1) = 13.95
 Prob > chi2 = 0.0002
```

is better, and in fact it has been suggested that the latter version of the test is more reliable. This assertion is confirmed by performing simulations and is in line with theoretical results of Philips and Park (1988). There is strong evidence against the proportionality of human capital effects between men and women, implying for this example that differences in the residual variances between the sexes can be ruled out as the explanation of the sex differences in the analysis of labor market participation.

## Saved results

`testnl` saves the following in `r()`:

### Scalars

|                      |                             |
|----------------------|-----------------------------|
| <code>r(df)</code>   | degrees of freedom          |
| <code>r(df_r)</code> | residual degrees of freedom |
| <code>r(chi2)</code> | $\chi^2$                    |
| <code>r(p)</code>    | significance                |
| <code>r(F)</code>    | $F$ statistic               |

### Matrices

|                   |                                                                                                               |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <code>r(G)</code> | derivatives of $R(\mathbf{b})$ with respect to $\mathbf{b}$ ; see <a href="#">Methods and formulas</a> below. |
| <code>r(R)</code> | $R(\mathbf{b}) - \mathbf{q}$ ; see <a href="#">Methods and formulas</a> below.                                |

## Methods and formulas

`testnl` is implemented as an ado-file.

After fitting a model, define  $\mathbf{b}$  as the resulting  $1 \times k$  parameter vector and  $\mathbf{V}$  as the  $k \times k$  covariance matrix. The (linear or nonlinear) hypothesis is given by  $R(\mathbf{b}) = \mathbf{q}$ , where  $R$  is a function returning a  $j \times 1$  vector. The Wald test formula is ([Greene 2008](#), 501)

$$W = \left\{ R(\mathbf{b}) - \mathbf{q} \right\}' \left( \mathbf{G} \mathbf{V} \mathbf{G}' \right)^{-1} \left\{ R(\mathbf{b}) - \mathbf{q} \right\}$$

where  $\mathbf{G}$  is the derivative matrix of  $R(\mathbf{b})$  with respect to  $\mathbf{b}$ .  $W$  is distributed as  $\chi^2$  if  $\mathbf{V}$  is an asymptotic covariance matrix.  $F = W/j$  is distributed as  $F$  for linear regression.

The adjustment methods for multiple testing are described in [\[R\] test](#). The adjustment for survey design effects is described in [\[SVY\] svy postestimation](#).

## References

- Allison, P. D. 1999. Comparing logit and probit coefficients across groups. *Sociological Methods and Research* 28: 186–208.
- Gould, W. W. 1996. [crc43: Wald test of nonlinear hypotheses after model estimation](#). *Stata Technical Bulletin* 29: 2–4. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 15–18. College Station, TX: Stata Press.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.
- Phillips, P. C. B., and J. Y. Park. 1988. On the formulation of Wald tests of nonlinear restrictions. *Econometrica* 56: 1065–1083.

## Also see

- [\[R\] lincom](#) — Linear combinations of estimators
- [\[R\] lrtest](#) — Likelihood-ratio test after estimation
- [\[R\] nlcom](#) — Nonlinear combinations of estimators
- [\[R\] test](#) — Test linear hypotheses after estimation
- [\[U\] 13.5 Accessing coefficients and standard errors](#)
- [\[U\] 20 Estimation and postestimation commands](#)

Syntax

tetrachoric *varlist* [*if*] [*in*] [*weight*] [, *options*]

| <i>options</i>                      | description                                                                                         |
|-------------------------------------|-----------------------------------------------------------------------------------------------------|
| Main                                |                                                                                                     |
| <code>stats(<i>statlist</i>)</code> | list of statistics; select up to 4 statistics; default is <code>stats(rho)</code>                   |
| <code>edwards</code>                | use the noniterative Edwards and Edwards estimator; default is the maximum likelihood estimator     |
| <code>print(#)</code>               | significance level for displaying coefficients                                                      |
| <code>star(#)</code>                | significance level for displaying with a star                                                       |
| <code>bonferroni</code>             | use Bonferroni-adjusted significance level                                                          |
| <code>sidak</code>                  | use Šidák-adjusted significance level                                                               |
| <code>pw</code>                     | calculate all the pairwise correlation coefficients by using all available data (pairwise deletion) |
| <code>zeroadjust</code>             | adjust frequencies when one cell has a zero count                                                   |
| <code>matrix</code>                 | display output in matrix form                                                                       |
| <code>notable</code>                | suppress display of correlations                                                                    |
| <code>posdef</code>                 | modify correlation matrix to be positive semidefinite                                               |
|                                     |                                                                                                     |
| <i>statlist</i>                     | description                                                                                         |
| <code>rho</code>                    | tetrachoric correlation coefficient                                                                 |
| <code>se</code>                     | standard error of rho                                                                               |
| <code>obs</code>                    | number of observations                                                                              |
| <code>p</code>                      | exact two-sided significance level                                                                  |

`by` is allowed; see [\[D\]](#) `by`.  
`fweights` are allowed; see [\[U\]](#) [11.1.6 weight](#).

Menu

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Tetrachoric correlations

Description

`tetrachoric` computes estimates of the tetrachoric correlation coefficients of the binary variables in *varlist*. All these variables should be 0, 1, or missing values.

Tetrachoric correlations assume a latent bivariate-normal distribution  $(X_1, X_2)$  for each pair of variables  $(v_1, v_2)$ , with a threshold model for the manifest variables,  $v_i = 1$  if and only if  $X_i > 0$ . The means and variances of the latent variables are not identified, but the correlation,  $r$ , of  $X_1$  and  $X_2$  can be estimated from the joint distribution of  $v_1$  and  $v_2$  and is called the tetrachoric correlation coefficients.

**tetrachoric** computes pairwise estimates of the tetrachoric correlations by the (iterative) maximum likelihood estimator obtained from bivariate probit without explanatory variables (see [R] **biprobbit**) by using the [Edwards and Edwards \(1984\)](#) noniterative estimator as the initial value.

The pairwise correlation matrix is returned as **r(Rho)** and can be used to perform a factor analysis or a principal component analysis of binary variables by using the **factormat** or **pcamat** commands; see [MV] **factor** and [MV] **pca**.

## Options

### Main

**stats(*statlist*)** specifies the statistics to be displayed in the matrix of output. **stats(rho)** is the default. Up to four statistics may be specified. **stats(rho se p obs)** would display the tetrachoric correlation coefficient, its standard error, the significance level, and the number of observations. If *varlist* contains only two variables, all statistics are shown in tabular form. **stats()**, **print()**, and **star()** have no effect unless the **matrix** option is also specified.

**edwards** specifies that the noniterative Edwards and Edwards estimator be used. The default is the maximum likelihood estimator. If you analyze many binary variables, you may want to use the fast noniterative estimator proposed by [Edwards and Edwards \(1984\)](#). However, if you have skewed variables, the approximation does not perform well.

**print(#)** specifies the maximum significance level of correlation coefficients to be printed. Correlation coefficients with larger significance levels are left blank in the matrix. Typing **tetrachoric ... , print(.10)** would list only those correlation coefficients that are significant at the 10% level or lower.

**star(#)** specifies the maximum significance level of correlation coefficients to be marked with a star. Typing **tetrachoric ... , star(.05)** would “star” all correlation coefficients significant at the 5% level or lower.

**bonferroni** makes the Bonferroni adjustment to calculated significance levels. This option affects printed significance levels and the **print()** and **star()** options. Thus **tetrachoric ... , print(.05) bonferroni** prints coefficients with Bonferroni-adjusted significance levels of 0.05 or less.

**sidak** makes the Šidák adjustment to calculated significance levels. This option affects printed significance levels and the **print()** and **star()** options. Thus **tetrachoric ... , print(.05) sidak** prints coefficients with Šidák-adjusted significance levels of 0.05 or less.

**pw** specifies that the tetrachoric correlation be calculated by using all available data. By default, **tetrachoric** uses casewise deletion, where observations are ignored if any of the specified variables in *varlist* are missing.

**zeroadjust** specifies that when one of the cells has a zero count, a frequency adjustment be applied in such a way as to increase the zero to one-half and maintain row and column totals.

**matrix** forces **tetrachoric** to display the statistics as a matrix, even if *varlist* contains only two variables. **matrix** is implied if more than two variables are specified.

**notable** suppresses the output.

**posdef** modifies the correlation matrix so that it is positive semidefinite, i.e., a proper correlation matrix. The modified result is the correlation matrix associated with the least-squares approximation of the tetrachoric correlation matrix by a positive-semidefinite matrix. If the correlation matrix is modified, the standard errors and significance levels are not displayed and are returned in **r()**.



## Remarks

Remarks are presented under the following headings:

*Association in 2-by-2 tables*

*Factor analysis of dichotomous variables*

*Tetrachoric correlations with simulated data*

## Association in 2-by-2 tables

Although a wide variety of measures of association in cross tabulations have been proposed, such measures are essentially equivalent (monotonically related) in the special case of  $2 \times 2$  tables—there is only 1 degree of freedom for nonindependence. Still, some measures have more desirable properties than others. Here we compare two measures: the standard Pearson correlation coefficient and the tetrachoric correlation coefficient. Given asymmetric row or column margins, Pearson correlations are limited to a range smaller than  $-1$  to  $1$ , although tetrachoric correlations can still span the range from  $-1$  to  $1$ . To illustrate, consider the following set of tables for two binary variables,  $X$  and  $Z$ :

|         | $Z = 0$  | $Z = 1$  |    |
|---------|----------|----------|----|
| $X = 0$ | $20 - a$ | $10 + a$ | 30 |
| $X = 1$ | $a$      | $10 - a$ | 10 |
|         | 20       | 20       | 40 |

For  $a$  equal to 0, 1, 2, 5, 8, 9, and 10, the Pearson and tetrachoric correlations for the above table are

| $a$         | 0     | 1     | 2     | 5 | 8      | 9      | 10     |
|-------------|-------|-------|-------|---|--------|--------|--------|
| Pearson     | 0.577 | 0.462 | 0.346 | 0 | -0.346 | -0.462 | -0.577 |
| Tetrachoric | 1.000 | 0.792 | 0.607 | 0 | -0.607 | -0.792 | -1.000 |

The restricted range for the Pearson correlation is especially unfortunate when you try to analyze the association between binary variables by using models developed for continuous data, such as factor analysis and principal component analysis.

The tetrachoric correlation of two variables  $(Y_1, Y_2)$  can be thought of as the Pearson correlation of two latent bivariate-normal distributed variables  $(Y_1^*, Y_2^*)$  with threshold measurement models  $Y_i = (Y_i^* > c_i)$  for unknown cutpoints  $c_i$ . Or equivalently,  $Y_i = (Y_i^{**} > 0)$  where the latent bivariate-normal  $(Y_1^{**}, Y_2^{**})$  are shifted versions of  $(Y_1^*, Y_2^*)$  so that the cutpoints are zero. Obviously, you must judge whether assuming underlying latent variables is meaningful for the data. If this assumption is justified, tetrachoric correlations have two advantages. First, you have an intuitive understanding of the size of correlations that are substantively interesting in your field of research, and this intuition is based on correlations that range from  $-1$  to  $1$ . Second, because the tetrachoric correlation for binary variables estimates the Pearson correlation of the latent continuous variables (assumed multivariate-normal distributed), you can use the tetrachoric correlations to analyze multivariate relationships between the dichotomous variables. When doing so, remember that you must interpret the model in terms of the underlying continuous variables.

➤ Example 1

To illustrate tetrachoric correlations, we examine three binary variables from the `familyvalues` dataset (described in example 2).

```
. use http://www.stata-press.com/data/r11/familyvalues
(Attitudes on gender, relationships and family)
```

```
. tabulate RS075 RS076
```

| fam att:<br>women in<br>charge bad | fam att: trad<br>division of labor |       | Total |
|------------------------------------|------------------------------------|-------|-------|
|                                    | 0                                  | 1     |       |
| 0                                  | 1,564                              | 979   | 2,543 |
| 1                                  | 119                                | 632   | 751   |
| Total                              | 1,683                              | 1,611 | 3,294 |

```
. correlate RS074 RS075 RS076
(obs=3291)
```

|       | RS074  | RS075  | RS076  |
|-------|--------|--------|--------|
| RS074 | 1.0000 |        |        |
| RS075 | 0.0396 | 1.0000 |        |
| RS076 | 0.1595 | 0.3830 | 1.0000 |

```
. tetrachoric RS074 RS075 RS076
(obs=3291)
```

|       | RS074  | RS075  | RS076  |
|-------|--------|--------|--------|
| RS074 | 1.0000 |        |        |
| RS075 | 0.0689 | 1.0000 |        |
| RS076 | 0.2480 | 0.6427 | 1.0000 |

As usual, the tetrachoric correlation coefficients are larger (in absolute value) and more dispersed than the Pearson correlations.



Factor analysis of dichotomous variables

➤ Example 2

Factor analysis is a popular model for measuring latent continuous traits. The standard estimators are appropriate only for continuous unimodal data. Because of the skewness implied by Bernoulli-distributed variables (especially when the probability is distributed unevenly), a factor analysis of a Pearson correlation matrix can be rather misleading when used in this context. A factor analysis of a matrix of tetrachoric correlations is more appropriate under these conditions (Uebersax 2000). We illustrate this with data on gender, relationship, and family attitudes of spouses using the Households in The Netherlands survey 1995 (Weesie et al. 1995). For attitude variables, it seems reasonable to assume that agreement or disagreement is just a coarse measurement of more nuanced underlying attitudes.

To demonstrate, we examine a few of the variables from the familyvalues dataset.

```
. use http://www.stata-press.com/data/r11/familyvalues
(Attitudes on gender, relationships and family)
. describe RS056-RS063
```

| variable name | storage<br>type | display<br>format | value<br>label | variable label                  |
|---------------|-----------------|-------------------|----------------|---------------------------------|
| RS056         | byte            | %9.0g             |                | fam att: should be together     |
| RS057         | byte            | %9.0g             |                | fam att: should fight for relat |
| RS058         | byte            | %9.0g             |                | fam att: should avoid conflict  |
| RS059         | byte            | %9.0g             |                | fam att: woman better nurturer  |
| RS060         | byte            | %9.0g             |                | fam att: both spouses money goo |
| RS061         | byte            | %9.0g             |                | fam att: woman techn school goo |
| RS062         | byte            | %9.0g             |                | fam att: man natural breadwinne |
| RS063         | byte            | %9.0g             |                | fam att: common leisure good    |

```
. summarize RS056-RS063
```

| Variable | Obs  | Mean     | Std. Dev. | Min | Max |
|----------|------|----------|-----------|-----|-----|
| RS056    | 3298 | .5630685 | .4960816  | 0   | 1   |
| RS057    | 3296 | .5400485 | .4984692  | 0   | 1   |
| RS058    | 3283 | .6387451 | .4804374  | 0   | 1   |
| RS059    | 3308 | .654474  | .4756114  | 0   | 1   |
| RS060    | 3302 | .3906723 | .487975   | 0   | 1   |
| RS061    | 3293 | .7102946 | .4536945  | 0   | 1   |
| RS062    | 3307 | .5857272 | .4926705  | 0   | 1   |
| RS063    | 3298 | .5379018 | .498637   | 0   | 1   |

```
. correlate RS056-RS063
(obs=3221)
```

|       | RS056   | RS057   | RS058   | RS059   | RS060   | RS061   | RS062  |
|-------|---------|---------|---------|---------|---------|---------|--------|
| RS056 | 1.0000  |         |         |         |         |         |        |
| RS057 | 0.1350  | 1.0000  |         |         |         |         |        |
| RS058 | 0.2377  | 0.0258  | 1.0000  |         |         |         |        |
| RS059 | 0.1816  | 0.0097  | 0.2550  | 1.0000  |         |         |        |
| RS060 | -0.1020 | -0.0538 | -0.0424 | 0.0126  | 1.0000  |         |        |
| RS061 | -0.1137 | 0.0610  | -0.1375 | -0.2076 | 0.0706  | 1.0000  |        |
| RS062 | 0.2014  | 0.0285  | 0.2273  | 0.4098  | -0.0793 | -0.2873 | 1.0000 |
| RS063 | 0.2057  | 0.1460  | 0.1049  | 0.0911  | 0.0179  | -0.0233 | 0.0975 |
|       | RS063   |         |         |         |         |         |        |
| RS063 | 1.0000  |         |         |         |         |         |        |

(Continued on next page)

Skewness in these data is relatively modest. For comparison, here are the tetrachoric correlations:

```
. tetrachoric RS056-RS063
(obs=3221)
```

|       | RS056   | RS057   | RS058   | RS059   | RS060   | RS061   | RS062  |
|-------|---------|---------|---------|---------|---------|---------|--------|
| RS056 | 1.0000  |         |         |         |         |         |        |
| RS057 | 0.2114  | 1.0000  |         |         |         |         |        |
| RS058 | 0.3716  | 0.0416  | 1.0000  |         |         |         |        |
| RS059 | 0.2887  | 0.0158  | 0.4007  | 1.0000  |         |         |        |
| RS060 | -0.1620 | -0.0856 | -0.0688 | 0.0208  | 1.0000  |         |        |
| RS061 | -0.1905 | 0.1011  | -0.2382 | -0.3664 | 0.1200  | 1.0000  |        |
| RS062 | 0.3135  | 0.0452  | 0.3563  | 0.6109  | -0.1267 | -0.4845 | 1.0000 |
| RS063 | 0.3187  | 0.2278  | 0.1677  | 0.1467  | 0.0286  | -0.0388 | 0.1538 |
|       | RS063   |         |         |         |         |         |        |
| RS063 | 1.0000  |         |         |         |         |         |        |

Again we see that the tetrachoric correlations are generally larger in absolute value than the Pearson correlations. The bivariate probit and Edwards and Edwards estimators (the `edwards` option) implemented in `tetrachoric` may return a correlation matrix that is not positive semidefinite—a mathematical property of any real correlation matrix. Positive definiteness is required by commands for analyses of correlation matrices, such as `factormat` and `pcamat`; see [\[MV\] factor](#) and [\[MV\] pca](#). The `posdef` option of `tetrachoric` tests for positive definiteness and projects the estimated correlation matrix to a positive-semidefinite matrix if needed.

```
. tetrachoric RS056-RS063, notable posdef
. matrix C = r(corr)
```

This time, we suppressed the display of the correlations with the `notable` option and requested that the correlation matrix be positive semidefinite with the `posdef` option. Had the correlation matrix not been positive definite, `tetrachoric` would have displayed a warning message and then adjusted the matrix to be positive semidefinite. We placed the resulting tetrachoric correlation matrix into a matrix, `C`, so that we can perform a factor analysis upon it.

`tetrachoric` with the `posdef` option asserted that `C` was positive definite because no warning message was displayed. We can verify this by using a familiar characterization of symmetric positive-definite matrices: all eigenvalues are real and positive.

```
. matrix symeigen eigenvectors eigenvalues = C
. matrix list eigenvalues
eigenvalues[1,8]
 e1 e2 e3 e4 e5 e6 e7
r1 2.5974789 1.3544664 1.0532476 .77980391 .73462018 .57984565 .54754512
 e8
r1 .35299228
```

We can proceed with a factor analysis on the matrix `C`. We use `factormat` and select iterated principal factors as the estimation method; see [\[MV\] factor](#).

```
. factorformat C, n(3221) ipf factor(2)
(obs=3221)
```

|                                    |                    |      |
|------------------------------------|--------------------|------|
| Factor analysis/correlation        | Number of obs =    | 3221 |
| Method: iterated principal factors | Retained factors = | 2    |
| Rotation: (unrotated)              | Number of params = | 15   |

| Factor  | Eigenvalue | Difference | Proportion | Cumulative |
|---------|------------|------------|------------|------------|
| Factor1 | 2.06855    | 1.40178    | 0.7562     | 0.7562     |
| Factor2 | 0.66677    | 0.47180    | 0.2438     | 1.0000     |
| Factor3 | 0.19497    | 0.06432    | 0.0713     | 1.0713     |
| Factor4 | 0.13065    | 0.10967    | 0.0478     | 1.1191     |
| Factor5 | 0.02098    | 0.10085    | 0.0077     | 1.1267     |
| Factor6 | -0.07987   | 0.01037    | -0.0292    | 1.0975     |
| Factor7 | -0.09024   | 0.08626    | -0.0330    | 1.0645     |
| Factor8 | -0.17650   | .          | -0.0645    | 1.0000     |

LR test: independent vs. saturated:  $\chi^2(28) = 4620.01$  Prob> $\chi^2 = 0.0000$

Factor loadings (pattern matrix) and unique variances

| Variable | Factor1 | Factor2 | Uniqueness |
|----------|---------|---------|------------|
| RS056    | 0.5528  | 0.4120  | 0.5247     |
| RS057    | 0.1124  | 0.4214  | 0.8098     |
| RS058    | 0.5333  | 0.0718  | 0.7105     |
| RS059    | 0.6961  | -0.1704 | 0.4865     |
| RS060    | -0.1339 | -0.0596 | 0.9785     |
| RS061    | -0.5126 | 0.2851  | 0.6560     |
| RS062    | 0.7855  | -0.2165 | 0.3361     |
| RS063    | 0.2895  | 0.3919  | 0.7626     |

4

## ► Example 3

We noted in example 2 that the matrix of estimates of the tetrachoric correlation coefficients need not be positive definite. Here is an example:

```
. use http://www.stata-press.com/data/r11/familyvalues
(Attitudes on gender, relationships and family)
```

```
. tetrachoric RS056-RS063 in 1/20, posdef
```

```
(obs=18)
```

matrix with tetrachoric correlations is not positive semidefinite;  
it has 2 negative eigenvalues

```
maxdiff(corr,adj-corr) = 0.2346
```

```
(adj-corr: tetrachoric correlations adjusted to be positive semidefinite)
```

| adj-corr | RS056   | RS057   | RS058   | RS059   | RS060   | RS061   | RS062  |
|----------|---------|---------|---------|---------|---------|---------|--------|
| RS056    | 1.0000  |         |         |         |         |         |        |
| RS057    | 0.5284  | 1.0000  |         |         |         |         |        |
| RS058    | 0.3012  | 0.2548  | 1.0000  |         |         |         |        |
| RS059    | 0.3251  | 0.2791  | 0.0550  | 1.0000  |         |         |        |
| RS060    | -0.5197 | -0.4222 | -0.7163 | 0.0552  | 1.0000  |         |        |
| RS061    | 0.3448  | 0.4815  | -0.0958 | -0.1857 | -0.0980 | 1.0000  |        |
| RS062    | 0.1066  | -0.0375 | 0.0072  | 0.3909  | -0.2333 | -0.7654 | 1.0000 |
| RS063    | 0.3830  | 0.4939  | 0.4336  | 0.0075  | -0.8937 | -0.0337 | 0.4934 |
| adj-corr | RS063   |         |         |         |         |         |        |
| RS063    | 1.0000  |         |         |         |         |         |        |

```
. mata
_____ mata (type end to exit) _____
: C2 = st_matrix("r(corr)")
: eigenvecs = .
: eigenvals = .
: syemeigensystem(C2, eigenvecs, eigenvals)
: eigenvals
 1 2 3 4
1 | 3.156592567 2.065279398 1.324911199 .7554904485
 |_____
 5 6 7 8
1 | .4845368741 .2131895139 2.02944e-16 -1.11650e-16
 |_____
: end

```

The estimated tetrachoric correlation matrix is rank-2 deficient. With this C2 matrix, we can only use models of correlation that allow for singular cases.



Tetrachoric correlations with simulated data

Example 4

We use `drawnorm` (see [\[D\] drawnorm](#)) to generate a sample of 1,000 observations from a bivariate normal distribution with means  $-1$  and  $1$ , unit variances, and correlation  $0.4$ .

```
. clear
. set seed 11000
. matrix m = (1, -1)
. matrix V = (1, 0.4 \ 0.4, 1)
. drawnorm c1 c2, n(1000) means(m) cov(V)
(obs 1000)
```

Now consider the measurement model assumed by the tetrachoric correlations. We observe only whether `c1` and `c2` are greater than zero,

```
. generate d1 = (c1 > 0)
. generate d2 = (c2 > 0)
. tabulate d1 d2
```

| d1    | d2  |     | Total |
|-------|-----|-----|-------|
|       | 0   | 1   |       |
| 0     | 144 | 6   | 150   |
| 1     | 699 | 151 | 850   |
| Total | 843 | 157 | 1,000 |

We want to estimate the correlation of `c1` and `c2` from the binary variables `d1` and `d2`. Pearson's correlation of the binary variables `d1` and `d2` is  $0.135$ —a seriously biased estimate of the underlying correlation  $\rho = 0.4$ .

```
. correlate d1 d2
(obs=1000)
```

|    | d1     | d2     |
|----|--------|--------|
| d1 | 1.0000 |        |
| d2 | 0.1351 | 1.0000 |

The tetrachoric correlation coefficient of d1 and d2 estimates the Pearson correlation of the latent continuous variables, c1 and c2.

```
. tetrachoric d1 d2
 Number of obs = 1000
 Tetrachoric rho = 0.4012
 Std error = 0.0775
Test of Ho: d1 and d2 are independent
2-sided exact P = 0.0000
```

The estimate of the tetrachoric correlation of d1 and d2, 0.4012, is much closer to the underlying correlation, 0.4, between c1 and c2.



## Saved results

`tetrachoric` saves the following in `r()`:

### Scalars

`r(rho)` tetrachoric correlation coefficient between variables 1 and 2  
`r(N)` number of observations  
`r(nneg)` number of negative eigenvalues (posdef only)  
`r(se_rho)` standard error of `r(rho)`  
`r(p)` exact two-sided significance level

### Macros

`r(method)` estimator used

### Matrices

`r(Rho)` tetrachoric correlation matrix  
`r(Se_Rho)` standard errors of `r(Rho)`  
`r(corr)` synonym for `r(Rho)`  
`r(Nobs)` number of observations used in computing correlation  
`r(P)` exact two-sided significance level matrix

## Methods and formulas

`tetrachoric` is implemented as an ado-file.

`tetrachoric` provides two estimators for the tetrachoric correlation  $\rho$  of two binary variables with the frequencies  $n_{ij}$ ,  $i, j = 0, 1$ . `tetrachoric` defaults to the slower (iterative) maximum likelihood estimator obtained from bivariate probit without explanatory variables (see [R] [biprobit](#)) by using the Edwards and Edwards noniterative estimator as the initial value. A fast (noniterative) estimator is also available by specifying the `edwards` option (Edwards and Edwards 1984; Digby 1983)

$$\hat{\rho} = \frac{\alpha - 1}{\alpha + 1}$$

where

$$\alpha = \left( \frac{n_{00}n_{11}}{n_{01}n_{10}} \right)^{\pi/4} \quad (\pi = 3.14\dots)$$

if all  $n_{ij} > 0$ . If  $n_{00} = 0$  or  $n_{11} = 0$ ,  $\hat{\rho} = -1$ ; if  $n_{01} = 0$  or  $n_{10} = 0$ ,  $\hat{\rho} = 1$ .

The asymptotic variance of the Edwards and Edwards estimator of the tetrachoric correlation is easily obtained by the delta method,

$$\text{avar}(\hat{\rho}) = \left( \frac{\pi\alpha}{2(1+\alpha)^2} \right)^2 \left( \frac{1}{n_{00}} + \frac{1}{n_{01}} + \frac{1}{n_{10}} + \frac{1}{n_{11}} \right)$$

provided all  $n_{ij} > 0$ , otherwise it is left undefined (missing). The Edwards and Edwards estimator is fast, but may be inaccurate if the margins are very skewed.

`tetrachoric` reports exact  $p$ -values for statistical independence, computed by the `exact` option of `[R] tabulate twoway`.

## References

- Brown, M. B. 1977. Algorithm AS 116: The tetrachoric correlation and its asymptotic standard error. *Applied Statistics* 26: 343–351.
- Brown, M. B., and J. K. Benedetti. 1977. On the mean and variance of the tetrachoric correlation coefficient. *Psychometrika* 42: 347–355.
- Digby, P. G. N. 1983. Approximating the tetrachoric correlation coefficient. *Biometrics* 39: 753–757.
- Edwards, J. H., and A. W. F. Edwards. 1984. Approximating the tetrachoric correlation coefficient. *Biometrics* 40: 563.
- Golub, G. H., and C. F. Van Loan. 1996. *Matrix Computations*. 3rd ed. Baltimore: Johns Hopkins University Press.
- Uebersax, J. S. 2000. Estimating a latent trait model by factor analysis of tetrachoric correlations.  
<http://ourworld.compuserve.com/homepages/jsuebersax/irt.htm>.
- Weesie, J., M. Kalmijn, W. Bernasco, and D. Giesen. 1995. *Households in The Netherlands 1995*. Utrecht, Netherlands: Datafile, ISCORE, University of Utrecht.

## Also see

- `[MV] factor` — Factor analysis
- `[MV] pca` — Principal component analysis
- `[R] tabulate twoway` — Two-way tables of frequencies
- `[R] biprobit` — Bivariate probit regression
- `[R] correlate` — Correlations (covariances) of variables or coefficients
- `[R] spearman` — Spearman’s and Kendall’s correlations



Syntax

`tobit` *devar* [*indepvars*] [*if*] [*in*] [*weight*], `ll` [(#)] `ul` [(#)] [*options*]

| <i>options</i>                         | description                                                                                                                                              |
|----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                  |                                                                                                                                                          |
| <code>noconstant</code>                | suppress constant term                                                                                                                                   |
| <code>*ll</code> [(#)]                 | left-censoring limit                                                                                                                                     |
| <code>*ul</code> [(#)]                 | right-censoring limit                                                                                                                                    |
| <code>offset</code> ( <i>varname</i> ) | include <i>varname</i> in model with coefficient constrained to 1                                                                                        |
| SE/Robust                              |                                                                                                                                                          |
| <code>vce</code> ( <i>vcetype</i> )    | <i>vcetype</i> may be <code>oim</code> , <code>robust</code> , <code>cluster</code> <i>clustvar</i> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                              |                                                                                                                                                          |
| <code>level</code> (#)                 | set confidence level; default is <code>level(95)</code>                                                                                                  |
| <code>display_options</code>           | control spacing and display of omitted variables and base and empty cells                                                                                |
| Maximization                           |                                                                                                                                                          |
| <code>maximize_options</code>          | control the maximization process; seldom used                                                                                                            |
| <sup>†</sup> <code>coeflegend</code>   | display coefficients' legend instead of coefficient table                                                                                                |

\*You must specify at least one of `ll`[(#)] or `ul`[(#)].

<sup>†</sup>`coeflegend` does not appear in the dialog box.

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

`bootstrap`, `by`, `jackknife`, `nestreg`, `rolling`, `statsby`, `stepwise`, and `svy` are allowed; see [U] 11.1.10 Prefix commands.

Weights are not allowed with the `bootstrap` prefix.

`aweight`s are not allowed with the `jackknife` prefix.

`vce()` and weights are not allowed with the `svy` prefix.

`aweight`s, `fweight`s, `pweight`s, and `iweight`s are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Linear models and related > Censored regression > Tobit regression

Description

`tobit` fits a model of *devar* on *indepvars* where the censoring values are fixed.

## Options

### Model

`noconstant`; see [R] [estimation options](#).

`ll(#)` and `ul(#)` indicate the lower and upper limits for censoring, respectively. You may specify one or both. Observations with *depvar*  $\leq$  `ll()` are left-censored; observations with *depvar*  $\geq$  `ul()` are right-censored; and remaining observations are not censored. You do not have to specify the censoring values at all. It is enough to type `ll`, `ul`, or both. When you do not specify a censoring value, `tobit` assumes that the lower limit is the minimum observed in the data (if `ll` is specified) and the upper limit is the maximum (if `ul` is specified).

`offset(varname)`; see [R] [estimation options](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

### Reporting

`level(#)`; see [R] [estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

### Maximization

`maximize_options`: `iterate(#)`, `[no]log`, `trace`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`; see [R] [maximize](#). These options are seldom used.

Unlike most maximum likelihood commands, `tobit` defaults to `nolog`—it suppresses the iteration log.

The following option is available with `tobit` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

Tobit estimation was originally developed by Tobin (1958). A consumer durable was purchased if a consumer's desire was high enough, where desire was measured by the dollar amount spent by the purchaser. If no purchase was made, the measure of desire was censored at zero.

### ► Example 1

We will demonstrate `tobit` with an artificial example, which in the process will allow us to emphasize the assumptions underlying the estimation. We have a dataset containing the mileage ratings and weights of 74 cars. There are no censored variables in this dataset, but we are going to create one. Before that, however, the relationship between mileage and weight in our complete data is

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. generate wgt = weight/1000

. regress mpg wgt
```

| Source   | SS         | df | MS         | Number of obs = 74     |  |  |
|----------|------------|----|------------|------------------------|--|--|
| Model    | 1591.99024 | 1  | 1591.99024 | F( 1, 72) = 134.62     |  |  |
| Residual | 851.469221 | 72 | 11.8259614 | Prob > F = 0.0000      |  |  |
|          |            |    |            | R-squared = 0.6515     |  |  |
|          |            |    |            | Adj R-squared = 0.6467 |  |  |
| Total    | 2443.45946 | 73 | 33.4720474 | Root MSE = 3.4389      |  |  |

| mpg   | Coef.     | Std. Err. | t      | P> t  | [95% Conf. Interval] |           |
|-------|-----------|-----------|--------|-------|----------------------|-----------|
| wgt   | -6.008687 | .5178782  | -11.60 | 0.000 | -7.041058            | -4.976316 |
| _cons | 39.44028  | 1.614003  | 24.44  | 0.000 | 36.22283             | 42.65774  |

(We divided `weight` by 1,000 simply to make discussing the resulting coefficients easier. We find that each additional 1,000 pounds of weight reduces mileage by 6 mpg.)

`mpg` in our data ranges from 12 to 41. Let us now pretend that our data were censored in the sense that we could not observe a mileage rating below 17 mpg. If the true `mpg` is 17 or less, all we know is that the `mpg` is less than or equal to 17:

```
. replace mpg=17 if mpg<=17
(14 real changes made)
```

```
. tobit mpg wgt, ll
```

```
Tobit regression
```

```
Number of obs = 74
LR chi2(1) = 72.85
Prob > chi2 = 0.0000
Pseudo R2 = 0.1815
```

```
Log likelihood = -164.25438
```

| mpg    | Coef.    | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|--------|----------|-----------|-------|-------|----------------------|-----------|
| wgt    | -6.87305 | .7002559  | -9.82 | 0.000 | -8.268658            | -5.477442 |
| _cons  | 41.49856 | 2.05838   | 20.16 | 0.000 | 37.39621             | 45.6009   |
| /sigma | 3.845701 | .3663309  |       |       | 3.115605             | 4.575797  |

```
Obs. summary: 18 left-censored observations at mpg<=17
 56 uncensored observations
 0 right-censored observations
```

The `replace` before estimation was not really necessary—we remapped all the mileage ratings below 17 to 17 merely to reassure you that `tobit` was not somehow using uncensored data. We typed `ll` after `tobit` to inform `tobit` that the data were left-censored. `tobit` found the minimum of `mpg` in our data and assumed that was the censoring point. We could also have dispensed with `replace` and typed `ll(17)`, informing `tobit` that all values of the dependent variable 17 and below are really censored at 17. In either case, at the bottom of the table, we are informed that there are, as a result, 18 left-censored observations.

On these data, our estimate is now a reduction of 6.9 mpg per 1,000 extra pounds of weight as opposed to 6.0. The parameter reported as `/sigma` is the estimated standard error of the regression; the resulting 3.8 is comparable with the estimated root mean squared error reported by `regress` of 3.4.

□ Technical note

You would never want to throw away information by purposefully censoring variables. The `regress` estimates are in every way preferable to those of `tobit`. Our example is designed solely to illustrate the relationship between `tobit` and `regress`. If you have uncensored data, use `regress`. If your data are censored, you have no choice but to use `tobit`.

▷ Example 2

`tobit` can also fit models that are censored from above. This time, let’s assume that we do not observe the actual mileage rating of cars yielding 24 mpg or better—we know only that it is at least 24. (Also assume that we have undone the change to mpg we made in the previous example.)

```
. use http://www.stata-press.com/data/r11/auto, clear
(1978 Automobile Data)
. generate wgt = weight/1000
. regress mpg wgt
(output omitted)
. tobit mpg wgt, ul(24)

Tobit regression Number of obs = 74
 LR chi2(1) = 90.72
 Prob > chi2 = 0.0000
Log likelihood = -129.8279 Pseudo R2 = 0.2589
```

| mpg    | Coef.     | Std. Err. | t      | P> t  | [95% Conf. Interval] |           |
|--------|-----------|-----------|--------|-------|----------------------|-----------|
| wgt    | -5.080645 | .43493    | -11.68 | 0.000 | -5.947459            | -4.213831 |
| _cons  | 36.08037  | 1.432056  | 25.19  | 0.000 | 33.22628             | 38.93445  |
| /sigma | 2.385357  | .2444604  |        |       | 1.898148             | 2.872566  |

Obs. summary:                    0 left-censored observations  
                              51 uncensored observations  
                              23 right-censored observations at mpg>=24

▷ Example 3

`tobit` can also fit models that are censored from both sides (the so-called two-limit tobit):

```
. tobit mpg wgt, ll(17) ul(24)

Tobit regression Number of obs = 74
 LR chi2(1) = 77.60
 Prob > chi2 = 0.0000
Log likelihood = -104.25976 Pseudo R2 = 0.2712
```

| mpg    | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|--------|-----------|-----------|-------|-------|----------------------|-----------|
| wgt    | -5.764448 | .7245417  | -7.96 | 0.000 | -7.208457            | -4.320438 |
| _cons  | 38.07469  | 2.255917  | 16.88 | 0.000 | 33.57865             | 42.57072  |
| /sigma | 2.886337  | .3952143  |       |       | 2.098676             | 3.673998  |

Obs. summary:                    18 left-censored observations at mpg<=17  
                              33 uncensored observations  
                              23 right-censored observations at mpg>=24

## Saved results

`tobit` saves the following in `e()`:

### Scalars

|                           |                                              |
|---------------------------|----------------------------------------------|
| <code>e(N)</code>         | number of observations                       |
| <code>e(N_unc)</code>     | number of uncensored observations            |
| <code>e(N_lc)</code>      | number of left-censored observations         |
| <code>e(N_rc)</code>      | number of right-censored observations        |
| <code>e(llopt)</code>     | contents of <code>ll()</code> , if specified |
| <code>e(ulopt)</code>     | contents of <code>ul()</code> , if specified |
| <code>e(k_aux)</code>     | number of auxiliary parameters               |
| <code>e(df_m)</code>      | model degrees of freedom                     |
| <code>e(df_r)</code>      | residual degrees of freedom                  |
| <code>e(r2_p)</code>      | pseudo- <i>R</i> -squared                    |
| <code>e(chi2)</code>      | $\chi^2$                                     |
| <code>e(ll)</code>        | log likelihood                               |
| <code>e(ll_0)</code>      | log likelihood, constant-only model          |
| <code>e(N_clust)</code>   | number of clusters                           |
| <code>e(F)</code>         | <i>F</i> statistic                           |
| <code>e(p)</code>         | significance                                 |
| <code>e(rank)</code>      | rank of <code>e(V)</code>                    |
| <code>e(converged)</code> | 1 if converged, 0 otherwise                  |

### Macros

|                            |                                                                |
|----------------------------|----------------------------------------------------------------|
| <code>e(cmd)</code>        | <code>tobit</code>                                             |
| <code>e(cmdline)</code>    | command as typed                                               |
| <code>e(depvar)</code>     | name of dependent variable                                     |
| <code>e(wtype)</code>      | weight type                                                    |
| <code>e(wexp)</code>       | weight expression                                              |
| <code>e(title)</code>      | title in estimation output                                     |
| <code>e(clustvar)</code>   | name of cluster variable                                       |
| <code>e(offset)</code>     | offset                                                         |
| <code>e(chi2type)</code>   | LR; type of model $\chi^2$ test                                |
| <code>e(vce)</code>        | <i>vcetype</i> specified in <code>vce()</code>                 |
| <code>e(vcetype)</code>    | title used to label Std. Err.                                  |
| <code>e(crittype)</code>   | optimization criterion                                         |
| <code>e(properties)</code> | <code>b V</code>                                               |
| <code>e(predict)</code>    | program used to implement <code>predict</code>                 |
| <code>e(footnote)</code>   | program used to implement the footnote display                 |
| <code>e(asbalanced)</code> | factor variables <code>fvset</code> as <code>asbalanced</code> |
| <code>e(asobserved)</code> | factor variables <code>fvset</code> as <code>asobserved</code> |

### Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

### Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

James Tobin (1918–2002) was an American economist who after education and research at Harvard moved to Yale, where he was on the faculty from 1950 to 1988. He made many outstanding contributions to economics and was awarded the Nobel Prize in 1981 “for his analysis of financial markets and their relations to expenditure decisions, employment, production and prices”. He appeared thinly disguised as a character in Herman Wouk’s novel *The Caine Mutiny* (1951) who thwarts the ambition of Willie Keith to be first in his class at midshipman school: “A mandarin-like midshipman named Tobit, with a domed forehead, measured quiet speech, and a mind like a sponge, was ahead of the field by a spacious percentage.”

## Methods and formulas

`tobit` is implemented as an ado-file.

See [Methods and formulas](#) in [R] [intreg](#).

See [Tobin \(1958\)](#) for the original derivation of the tobit model. An introductory description of the tobit model can be found in, for instance, [Wooldridge \(2009, 587–595\)](#), [Davidson and MacKinnon \(2004, 484–486\)](#), [Long \(1997, 196–210\)](#), and [Maddala and Lahiri \(2006, 333–336\)](#). [Cameron and Trivedi \(2009, chap. 16\)](#) discuss the tobit model using Stata examples.

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

`tobit` also supports estimation with survey data. For details on VCEs with survey data, see [\[SVY\] variance estimation](#).

## References

- Amemiya, T. 1973. Regression analysis when the dependent variable is truncated normal. *Econometrica* 41: 997–1016.
- . 1984. Tobit models: A survey. *Journal of Econometrics* 24: 3–61.
- Cameron, A. C., and P. K. Trivedi. 2009. *Microeconometrics Using Stata*. College Station, TX: Stata Press.
- Cong, R. 2000. `sg144: Marginal effects of the tobit model`. *Stata Technical Bulletin* 56: 27–34. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 189–197. College Station, TX: Stata Press.
- Davidson, R., and J. G. MacKinnon. 2004. *Econometric Theory and Methods*. New York: Oxford University Press.
- Drukker, D. M. 2002. Bootstrapping a conditional moments test for normality after tobit estimation. *Stata Journal* 2: 125–139.
- Goldberger, A. S. 1983. Abnormal selection bias. In *Studies in Econometrics, Time Series, and Multivariate Statistics*, ed. S. Karlin, T. Amemiya, and L. A. Goodman, 67–84. New York: Academic Press.
- Hurd, M. 1979. Estimation in truncated samples when there is heteroscedasticity. *Journal of Econometrics* 11: 247–258.
- Kendall, M. G., and A. Stuart. 1973. *The Advanced Theory of Statistics, Vol. 2: Inference and Relationship*. New York: Hafner.
- Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.
- Maddala, G. S., and K. Lahiri. 2006. *Introduction to Econometrics*. 4th ed. New York: Wiley.
- McDonald, J. F., and R. A. Moffitt. 1980. The use of tobit analysis. *Review of Economics and Statistics* 62: 318–321.
- Shiller, R. J. 1999. The ET interview: Professor James Tobin. *Econometric Theory* 15: 867–900.
- Stewart, M. B. 1983. On least squares estimation when the dependent variable is grouped. *Review of Economic Studies* 50: 737–753.
- Tobin, J. 1958. Estimation of relationships for limited dependent variables. *Econometrica* 26: 24–36.
- Wooldridge, J. M. 2009. *Introductory Econometrics: A Modern Approach*. 4th ed. Cincinnati, OH: South-Western.

## Also see

- [R] **tobit postestimation** — Postestimation tools for tobit
- [R] **intreg** — Interval regression
- [R] **heckman** — Heckman selection model
- [R] **ivtobit** — Tobit model with continuous endogenous regressors
- [R] **regress** — Linear regression
- [R] **truncreg** — Truncated regression
- [XT] **xtintreg** — Random-effects interval-data regression models
- [XT] **xttobit** — Random-effects tobit models
- [U] **20 Estimation and postestimation commands**

Description

The following postestimation commands are available for `tobit`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>hausman</code>             | Hausman's specification test                                                                        |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>            | link test for model specification                                                                   |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic nooffset]

predict [type] { stub* | newvarreg newvarsigma } [if] [in] , scores
```

| statistic               | description                                 |
|-------------------------|---------------------------------------------|
| <code>xb</code>         | linear prediction; the default              |
| <code>stdp</code>       | standard error of the linear prediction     |
| <code>stdf</code>       | standard error of the forecast              |
| <code>pr(a,b)</code>    | $\Pr(a < y_j < b)$                          |
| <code>e(a,b)</code>     | $E(y_j   a < y_j < b)$                      |
| <code>ystar(a,b)</code> | $E(y_j^*), y_j^* = \max\{a, \min(y_j, b)\}$ |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

`stdf` is not allowed with `svy` estimation results.



where  $a$  and  $b$  may be numbers or variables;  $a$  missing ( $a \geq .$ ) means  $-\infty$ , and  $b$  missing ( $b \geq .$ ) means  $+\infty$ ; see [U] [12.2.1 Missing values](#).

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

**xb**, the default, calculates the linear prediction.

**stdp** calculates the standard error of the prediction, which can be thought of as the standard error of the predicted expected value or mean for the observation's covariate pattern. The standard error of the prediction is also referred to as the standard error of the fitted value.

**stdf** calculates the standard error of the forecast, which is the standard error of the point prediction for 1 observation. It is commonly referred to as the standard error of the future or forecast value. By construction, the standard errors produced by **stdf** are always larger than those produced by **stdp**; see [Methods and formulas](#) in [R] [regress](#).

**pr( $a,b$ )** calculates  $\Pr(a < \mathbf{x}_j\mathbf{b} + u_j < b)$ , the probability that  $y_j|\mathbf{x}_j$  would be observed in the interval  $(a, b)$ .

$a$  and  $b$  may be specified as numbers or variable names;  $lb$  and  $ub$  are variable names;

**pr(20,30)** calculates  $\Pr(20 < \mathbf{x}_j\mathbf{b} + u_j < 30)$ ;

**pr( $lb,ub$ )** calculates  $\Pr(lb < \mathbf{x}_j\mathbf{b} + u_j < ub)$ ; and

**pr(20, $ub$ )** calculates  $\Pr(20 < \mathbf{x}_j\mathbf{b} + u_j < ub)$ .

$a$  missing ( $a \geq .$ ) means  $-\infty$ ; **pr(.,30)** calculates  $\Pr(-\infty < \mathbf{x}_j\mathbf{b} + u_j < 30)$ ;

**pr( $lb,30$ )** calculates  $\Pr(-\infty < \mathbf{x}_j\mathbf{b} + u_j < 30)$  in observations for which  $lb \geq .$  and calculates  $\Pr(lb < \mathbf{x}_j\mathbf{b} + u_j < 30)$  elsewhere.

$b$  missing ( $b \geq .$ ) means  $+\infty$ ; **pr(20,.)** calculates  $\Pr(+\infty > \mathbf{x}_j\mathbf{b} + u_j > 20)$ ;

**pr(20, $ub$ )** calculates  $\Pr(+\infty > \mathbf{x}_j\mathbf{b} + u_j > 20)$  in observations for which  $ub \geq .$  and calculates  $\Pr(20 < \mathbf{x}_j\mathbf{b} + u_j < ub)$  elsewhere.

**e( $a,b$ )** calculates  $E(\mathbf{x}_j\mathbf{b} + u_j \mid a < \mathbf{x}_j\mathbf{b} + u_j < b)$ , the expected value of  $y_j|\mathbf{x}_j$  conditional on  $y_j|\mathbf{x}_j$  being in the interval  $(a, b)$ , meaning that  $y_j|\mathbf{x}_j$  is censored.

$a$  and  $b$  are specified as they are for **pr()**.

**ystar( $a,b$ )** calculates  $E(y_j^*)$ , where  $y_j^* = a$  if  $\mathbf{x}_j\mathbf{b} + u_j \leq a$ ,  $y_j^* = b$  if  $\mathbf{x}_j\mathbf{b} + u_j \geq b$ , and  $y_j^* = \mathbf{x}_j\mathbf{b} + u_j$  otherwise, meaning that  $y_j^*$  is truncated.  $a$  and  $b$  are specified as they are for **pr()**.

**nooffset** is relevant only if you specified **offset(varname)**. It modifies the calculations made by **predict** so that they ignore the offset variable; the linear prediction is treated as  $\mathbf{x}_j\mathbf{b}$  rather than as  $\mathbf{x}_j\mathbf{b} + \text{offset}_j$ .

**scores** calculates equation-level score variables.

The first new variable will contain  $\partial \ln L / \partial (\mathbf{x}_j\boldsymbol{\beta})$ .

The second new variable will contain  $\partial \ln L / \partial \sigma$ .

## Remarks

Following [Cong \(2000\)](#), write the tobit model as

$$y_i^* = \begin{cases} y_i, & \text{if } a < y_i < b \\ a, & \text{if } y_i \leq a \\ b, & \text{if } y_i \geq b \end{cases}$$

$y_i$  is a latent variable; instead, we observe  $y_i^*$ , which is bounded between  $a$  and  $b$  if  $y_i$  is outside those bounds.

There are four types of marginal effects that may be of interest in the tobit model, depending on the application:

1. The  $\beta$  coefficients themselves measure how the unobserved variable  $y_i$  changes with respect to changes in the regressors.
2. The marginal effects of the truncated expected value  $E[y_i * |a < y_i^* < b]$  measure the changes in  $y_i$  with respect to changes in the regressors among the subpopulation for which  $y_i$  is not at a boundary.
3. The marginal effects of the censored expected value  $E[y_i^*]$  describe how the observed variable  $y_i^*$  changes with respect to the regressors.
4. The marginal effects of  $\Pr[a < y_i^* < b]$  describe how the probability of being uncensored changes with respect to the regressors.

In the next example, we show how to obtain each of these.

### ► Example 1

In [example 3](#) of [\[R\] tobit](#), we fit a two-limit tobit model of `mpg` on `wgt`. The marginal effects of  $y_i$  are simply the  $\beta$ 's reported by `tobit`:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. generate wgt = weight/1000

. tobit mpg wgt, ll(17) ul(24)
```

Tobit regression

Number of obs = 74

LR chi2(1) = 77.60

Prob > chi2 = 0.0000

Pseudo R2 = 0.2712

Log likelihood = -104.25976

| mpg    | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|--------|-----------|-----------|-------|-------|----------------------|-----------|
| wgt    | -5.764448 | .7245417  | -7.96 | 0.000 | -7.208457            | -4.320438 |
| _cons  | 38.07469  | 2.255917  | 16.88 | 0.000 | 33.57865             | 42.57072  |
| /sigma | 2.886337  | .3952143  |       |       | 2.098676             | 3.673998  |

```
Obs. summary: 18 left-censored observations at mpg<=17
 33 uncensored observations
 23 right-censored observations at mpg>=24
```

If `mpg` were not censored at 17 and 24, then a 1,000-pound increase in a car's weight (which is a 1-unit increase in `wgt`) would lower fuel economy by 5.8 mpg.

To get the marginal effect on the censored expected value conditional on weight being at each of four values (1,000; 2,000; 3,000; and 4,000 pounds), we type

```
. margins, dydx(wgt) predict(e(17,24)) at(wgt=(1 2 3 4 5))
Conditional marginal effects Number of obs = 74
Model VCE : OIM
Expression : E(mpg|17<mpg<24), predict(e(17,24))
dy/dx w.r.t. : wgt
1._at : wgt = 1
2._at : wgt = 2
3._at : wgt = 3
4._at : wgt = 4
5._at : wgt = 5
```

|     |     | Delta-method |           | z      | P> z  | [95% Conf. Interval] |           |
|-----|-----|--------------|-----------|--------|-------|----------------------|-----------|
|     |     | dy/dx        | Std. Err. |        |       |                      |           |
| wgt | _at |              |           |        |       |                      |           |
|     | 1   | -.4286109    | .067281   | -6.37  | 0.000 | -.5604793            | -.2967425 |
|     | 2   | -1.166572    | .0827549  | -14.10 | 0.000 | -1.328768            | -1.004375 |
|     | 3   | -2.308842    | .4273727  | -5.40  | 0.000 | -3.146477            | -1.471207 |
|     | 4   | -1.288896    | .0889259  | -14.49 | 0.000 | -1.463188            | -1.114604 |
|     | 5   | -.4685559    | .0740565  | -6.33  | 0.000 | -.613704             | -.3234079 |

For cars whose fuel economy lies between 17 and 24 mpg, a 1,000-pound increase in the car's weight lowers fuel economy most for a 3,000 pound car, 2.3 mpg, and least for a 5,000 pound car, 0.5 mpg.

To get the marginal effect on the truncated expected value at the same levels of wgt, we type

```
. margins, dydx(wgt) predict(ystar(17,24)) at(wgt=(1 2 3 4 5))
Conditional marginal effects Number of obs = 74
Model VCE : OIM
Expression : E(mpg*|17<mpg<24), predict(ystar(17,24))
dy/dx w.r.t. : wgt
1._at : wgt = 1
2._at : wgt = 2
3._at : wgt = 3
4._at : wgt = 4
5._at : wgt = 5
```

|     |     | Delta-method |           | z     | P> z  | [95% Conf. Interval] |           |
|-----|-----|--------------|-----------|-------|-------|----------------------|-----------|
|     |     | dy/dx        | Std. Err. |       |       |                      |           |
| wgt | _at |              |           |       |       |                      |           |
|     | 1   | -.0117778    | .0204876  | -0.57 | 0.565 | -.0519328            | .0283771  |
|     | 2   | -1.0861      | .311273   | -3.49 | 0.000 | -1.696184            | -.4760162 |
|     | 3   | -4.45315     | .4772541  | -9.33 | 0.000 | -5.388551            | -3.51775  |
|     | 4   | -1.412822    | .3289702  | -4.29 | 0.000 | -2.057591            | -.768052  |
|     | 5   | -.0424155    | .0390014  | -1.09 | 0.277 | -.1188568            | .0340258  |

The decrease in observed fuel economy (which is bounded between 17 and 24) for a 1,000 pound increase in weight is about 4.5 mpg for a 3,000 pound car. For a 5,000 pound car, the decrease is almost negligible because of the binding truncation at 17 mpg.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## References

- Cong, R. 2000. [sg144: Marginal effects of the tobit model](#). *Stata Technical Bulletin* 56: 27–34. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 189–197. College Station, TX: Stata Press.
- McDonald, J. F., and R. A. Moffitt. 1980. The use of tobit analysis. *Review of Economics and Statistics* 62: 318–321.

## Also see

- [R] [tobit](#) — Tobit regression
- [U] [20 Estimation and postestimation commands](#)

Syntax

total *varlist* [*if*] [*in*] [*weight*] [, *options*]

| <i>options</i>                                                        | description                                                                                                     |
|-----------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| if/in/over<br>over( <i>varlist</i> [, <u>no</u> label])               | group over subpopulations defined by <i>varlist</i> ; optionally, suppress group labels                         |
| SE/Cluster<br>vce( <i>vcetype</i> )                                   | <i>vcetype</i> may be <u>analytic</u> , <u>cluster</u> <i>clustvar</i> , <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting<br><u>level</u> (#)<br><u>no</u> header<br><u>no</u> legend | set confidence level; default is level(95)<br>suppress table header<br>suppress table legend                    |
| <sup>†</sup> <u>coef</u> legend                                       | display coefficients' legend instead of coefficient table                                                       |

<sup>†</sup>coeflegend does not appear in the dialog box.  
bootstrap, jackknife, rolling, statsby, and svy are allowed; see [U] 11.1.10 Prefix commands.  
Weights are not allowed with the bootstrap prefix.  
vce() and weights are not allowed with the svy prefix.  
fweights, pweights, and iweights are allowed; see [U] 11.1.6 weight.  
See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Totals

Description

total produces estimates of totals, along with standard errors.

Options

if/in/over

over(*varlist* [, nolabel]) specifies that estimates be computed for multiple subpopulations, which are identified by the different values of the variables in *varlist*.

When this option is supplied with one variable name, such as `over(varname)`, the value labels of *varname* are used to identify the subpopulations. If *varname* does not have labeled values (or there are unlabeled values), the values themselves are used, provided that they are nonnegative integers. Noninteger values, negative values, and labels that are not valid Stata names are substituted with a default identifier.

When `over()` is supplied with multiple variable names, each subpopulation is assigned a unique default identifier.

`no label` specifies that value labels attached to the variables identifying the subpopulations be ignored.

SE/Cluster

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

`vce(analytic)`, the default, uses the analytically derived variance estimator associated with the sample total.

Reporting

`level(#)`; see [R] [estimation options](#).

`noheader` prevents the table header from being displayed. This option implies `nolegend`.

`nolegend` prevents the table legend identifying the subpopulations from being displayed.

The following option is available with `total` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

### ➤ Example 1

Suppose that we collected data on incidence of heart attacks. The variable `heartatk` indicates whether a person ever had a heart attack (1 means yes; 0 means no). We can then estimate the total number of persons who have had heart attacks for each `sex` in the population represented by the data we collected.

```
. use http://www.stata-press.com/data/r11/total
. total heartatk [pw=swgt], over(sex)
Total estimation Number of obs = 4946
 Male: sex = Male
 Female: sex = Female
```

| Over     | Total  | Std. Err. | [95% Conf. Interval] |          |
|----------|--------|-----------|----------------------|----------|
| heartatk |        |           |                      |          |
| Male     | 944559 | 104372.3  | 739943               | 1149175  |
| Female   | 581590 | 82855.59  | 419156.3             | 744023.7 |

## Saved results

`total` saves the following in `e()`:

|                         |                                               |                               |                                                      |
|-------------------------|-----------------------------------------------|-------------------------------|------------------------------------------------------|
| Scalars                 |                                               |                               |                                                      |
| <code>e(N)</code>       | number of observations                        | <code>e(k_eq)</code>          | number of equations in <code>e(b)</code>             |
| <code>e(N_over)</code>  | number of subpopulations                      | <code>e(df_r)</code>          | sample degrees of freedom                            |
| <code>e(N_clust)</code> | number of clusters                            | <code>e(rank)</code>          | rank of <code>e(V)</code>                            |
| Macros                  |                                               |                               |                                                      |
| <code>e(cmd)</code>     | <code>total</code>                            | <code>e(over_labels)</code>   | labels from <code>over()</code> variables            |
| <code>e(cmdline)</code> | command as typed                              | <code>e(over_namelist)</code> | names from <code>e(over_labels)</code>               |
| <code>e(varlist)</code> | <code>varlist</code>                          | <code>e(vce)</code>           | <code>vcetype</code> specified in <code>vce()</code> |
| <code>e(wtype)</code>   | weight type                                   | <code>e(vcetype)</code>       | title used to label Std. Err.                        |
| <code>e(wexp)</code>    | weight expression                             | <code>e(properties)</code>    | <code>b V</code>                                     |
| <code>e(title)</code>   | title in estimation output                    | <code>e(estat_cmd)</code>     | program used to implement <code>estat</code>         |
| <code>e(cluster)</code> | name of cluster variable                      | <code>e(marginsnotok)</code>  | predictions disallowed by <code>margins</code>       |
| <code>e(over)</code>    | <code>varlist</code> from <code>over()</code> |                               |                                                      |
| Matrices                |                                               |                               |                                                      |
| <code>e(b)</code>       | vector of total estimates                     |                               |                                                      |
| <code>e(V)</code>       | (co)variance estimates                        |                               |                                                      |
| <code>e(_N)</code>      | vector of numbers of nonmissing observations  |                               |                                                      |
| <code>e(error)</code>   | error code corresponding to <code>e(b)</code> |                               |                                                      |
| Functions               |                                               |                               |                                                      |
| <code>e(sample)</code>  | marks estimation sample                       |                               |                                                      |

## Methods and formulas

`total` is implemented as an ado-file.

Methods and formulas are presented under the following headings:

[The total estimator](#)  
[Survey data](#)  
[The survey total estimator](#)  
[The poststratified total estimator](#)  
[Subpopulation estimation](#)

## The total estimator

Let  $y$  denote the variable on which to calculate the total and  $y_j, j = 1, \dots, n$ , denote an individual observation on  $y$ . Let  $w_j$  be the frequency weight (or `iweight`), and if no weight is specified, define  $w_j = 1$  for all  $j$ . See the next section for `pweighted` data. The sum of the weights is an estimate of the population size:

$$\hat{N} = \sum_{j=1}^n w_j$$

If the population values of  $y$  are denoted by  $Y_j, j = 1, \dots, N$ , the associated population total is

$$Y = \sum_{j=1}^N Y_j = N\bar{y}$$

where  $\bar{y}$  is the population mean. The total is estimated as

$$\hat{Y} = \hat{N}\bar{y}$$

The variance estimator for the total is

$$\hat{V}(\hat{Y}) = \hat{N}^2 \hat{V}(\bar{y})$$

where  $\hat{V}(\bar{y})$  is the variance estimator for the mean; see [R] [mean](#). The standard error of the total is the square root of the variance.

If  $x$ ,  $x_j$ ,  $\bar{x}$ , and  $\hat{X}$  are similarly defined for another variable (observed jointly with  $y$ ), the covariance estimator between  $\hat{X}$  and  $\hat{Y}$  is

$$\widehat{\text{Cov}}(\hat{X}, \hat{Y}) = \hat{N}^2 \widehat{\text{Cov}}(\bar{x}, \bar{y})$$

where  $\widehat{\text{Cov}}(\bar{x}, \bar{y})$  is the covariance estimator between two means; see [R] [mean](#).

## Survey data

See [SVY] [variance estimation](#) and [SVY] [poststratification](#) for discussions that provide background information for the following formulas.

## The survey total estimator

Let  $Y_j$  be a survey item for the  $j$ th individual in the population, where  $j = 1, \dots, M$  and  $M$  is the size of the population. The associated population total for the item of interest is

$$Y = \sum_{j=1}^M Y_j$$

Let  $y_j$  be the survey item for the  $j$ th sampled individual from the population, where  $j = 1, \dots, m$  and  $m$  is the number of observations in the sample.

The estimator  $\hat{Y}$  for the population total  $Y$  is

$$\hat{Y} = \sum_{j=1}^m w_j y_j$$

where  $w_j$  is a sampling weight. The estimator for the number of individuals in the population is

$$\hat{M} = \sum_{j=1}^m w_j$$

The score variable for the total estimator is the variable itself,

$$z_j(\hat{Y}) = y_j$$



## The poststratified total estimator

Let  $P_k$  denote the set of sampled observations that belong to poststratum  $k$ , and define  $I_{P_k}(j)$  to indicate if the  $j$ th observation is a member of poststratum  $k$ , where  $k = 1, \dots, L_P$  and  $L_P$  is the number of poststrata. Also, let  $M_k$  denote the population size for poststratum  $k$ .  $P_k$  and  $M_k$  are identified by specifying the `poststrata()` and `postweight()` options on `svyset`.

The estimator for the poststratified total is

$$\hat{Y}^P = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \hat{Y}_k = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \sum_{j=1}^m I_{P_k}(j) w_j y_j$$

where

$$\widehat{M}_k = \sum_{j=1}^m I_{P_k}(j) w_j$$

The score variable for the poststratified total is

$$z_j(\hat{Y}^P) = \sum_{k=1}^{L_P} I_{P_k}(j) \frac{M_k}{\widehat{M}_k} \left( y_j - \frac{\hat{Y}_k}{\widehat{M}_k} \right)$$

## Subpopulation estimation

Let  $S$  denote the set of sampled observations that belong to the subpopulation of interest, and define  $I_S(j)$  to indicate if the  $j$ th observation falls within the subpopulation.

The estimator for the subpopulation total is

$$\hat{Y}^S = \sum_{j=1}^m I_S(j) w_j y_j$$

and its score variable is

$$z_j(\hat{Y}^S) = I_S(j) y_j$$

The estimator for the poststratified subpopulation total is

$$\hat{Y}^{PS} = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \hat{Y}_k^S = \sum_{k=1}^{L_P} \frac{M_k}{\widehat{M}_k} \sum_{j=1}^m I_{P_k}(j) I_S(j) w_j y_j$$

and its score variable is

$$z_j(\hat{Y}^{PS}) = \sum_{k=1}^{L_P} I_{P_k}(j) \frac{M_k}{\widehat{M}_k} \left\{ I_S(j) y_j - \frac{\hat{Y}_k^S}{\widehat{M}_k} \right\}$$

## References

Cochran, W. G. 1977. *Sampling Techniques*. 3rd ed. New York: Wiley.

Stuart, A., and J. K. Ord. 1994. *Kendall's Advanced Theory of Statistics: Distribution Theory, Vol I*. 6th ed. London: Arnold.

## Also see

[R] **total postestimation** — Postestimation tools for total

[R] **mean** — Estimate means

[R] **proportion** — Estimate proportions

[R] **ratio** — Estimate ratios

[SVY] **direct standardization** — Direct standardization of means, proportions, and ratios

[SVY] **poststratification** — Poststratification for survey data

[SVY] **subpopulation estimation** — Subpopulation estimation for survey data

[SVY] **svy estimation** — Estimation commands for survey data

[SVY] **variance estimation** — Variance estimation for survey data

[U] **20 Estimation and postestimation commands**

# Title

total postestimation — Postestimation tools for total

## Description

The following postestimation commands are available for `total`:

| command                  | description                                                                                         |
|--------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>       | VCE                                                                                                 |
| <code>estat (svy)</code> | postestimation statistics for survey data                                                           |
| <code>estimates</code>   | cataloging estimation results                                                                       |
| <code>lincom</code>      | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>nlcom</code>       | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>test</code>        | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>      | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

## Remarks

### ► Example 1

Continuing with our data on incidence of heart attacks from [example 1](#) in [\[R\] total](#), we want to test whether there are twice as many heart attacks among men than women in the population.

```
. use http://www.stata-press.com/data/r11/total
. total heartatk [pw=swgt], over(sex)
(output omitted)
. test _b[Male] = 2*_b[Female]
(1) [heartatk]Male - 2 [heartatk]Female = 0
 F(1, 4945) = 1.25
 Prob > F = 0.2643
```

Thus we do not reject our hypothesis that the total number of heart attacks for men is twice that for women in the population.



## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] **total** — Estimate totals

[U] **20 Estimation and postestimation commands**

# Title

**translate** — Print and translate logs

## Syntax

*Print log and SMCL files*

```
print filename [, like(ext) name(windowname) override_options]
```

*Translate log files to SMCL files and vice versa*

```
translate filenamein filenameout [, translator(tname) name(windowname)
 override_options replace]
```

*View translator parameter settings*

```
translator query [tname]
```

*Change translator parameter settings*

```
translator set [tname setopt setval]
```

*Return translator parameter settings to default values*

```
translator reset tname
```

*List current mappings from one extension to another*

```
transmap query [.ext]
```

*Specify that files with one extension be treated the same as files with another extension*

```
transmap define .extnew .extold
```

*filename* in `print`, in addition to being a filename to be printed, may be specified as `@Results` to mean the Results window and `@Viewer` to mean the Viewer window.

*filename*<sub>in</sub> in `translate` may be specified just as *filename* in `print`.

*tname* in `translator` specifies the name of a translator; see the `translator()` option under *Options for translate*.

## Description

`print` prints log, SMCL, and text files. Although there is considerable flexibility in how `print` (and `translate`, which `print` uses) can be set to work, they have already been set up and should just work:

```
. print mylog.smcl
. print mylog.log
```

Unix users may discover that they need to do a bit of setup before `print` works; see [Printing files, Unix](#) below. International Unix users may also wish to modify the default paper size. All users can tailor `print` and `translate` to their needs.

`print` may also be used to print the current contents of the Results window or the Viewer. For instance, the current contents of the Results window could be printed by typing

```
. print @Results
```

`translate` translates log and SMCL files from one format to another, the other typically being suitable for printing. `translate` can also translate SMCL logs (logs created by typing, say, `log using mylog`) to ASCII text:

```
. translate mylog.smcl mylog.log
```

You can use `translate` to recover a log when you have forgotten to start one. You may type

```
. translate @Results mylog.txt
```

to capture as ASCII text what is currently shown in the Results window.

This entry provides a general overview of `print` and `translate` and covers in detail the printing and translation of text (nongraphic) files.

## Options for print

`like(ext)` specifies how the file should be translated to a form suitable for printing. The default is to determine the translation method from the extension of *filename*. Thus `mylog.smcl` is translated according to the rule for translating `smcl` files, `myfile.txt` is translated according to the rule for translating `txt` files, and so on. (These rules are, in fact, `translate`'s `smcl2prn` and `txt2prn` translators, but put that aside for the moment.)

Rules for the following extensions are predefined:

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>.txt</code>  | assume input file contains ASCII text           |
| <code>.log</code>  | assume input file contains Stata log ASCII text |
| <code>.smcl</code> | assume input file contains SMCL                 |

To print a file that has an extension different from those listed above, you can define a new extension, but you do not have to do that. Assume that you wish to print the file `read.me`, which you know to contain ASCII text. If you were just to type `print read.me`, you would be told that Stata cannot translate `.me` files. (You would actually be told that the translator for `me2prn` was not found.) You could type `print read.me, like(txt)` to tell `print` to print `read.me` like a `.txt` file.

On the other hand, you could type

```
. transmap define .me .txt
```

to tell Stata that `.me` files are always to be treated like `.txt` files. If you did that, Stata would remember the new rule, even in future sessions.

When you specify the `like()` option, you override the recorded rules. So, if you were to type `print mylog.smcl, like(txt)`, the file would be printed as ASCII text (meaning that all the SMCL commands would show).

`name(windowname)` specifies which window to print when printing a Viewer. The default is for Stata to print the topmost Viewer [Unix(GUI) users: See the second [technical note](#) in [Printing files, Unix](#)]. The `name()` option is ignored when printing the Results window.

The window name is located inside parentheses in the window title. For example, if the title for a Viewer window is *Viewer (#1) [help print]*, the name for the window is #1.

*override\_options* refers to `translate`'s override options. `print` uses `translate` to translate the file into a format suitable for sending to the printer. To find out what you can override for *X* files (files ending in *.X*), type `translator query X2prn`. For instance, to find out what you can override when printing *.smcl* files, type

```
. translator query smcl2prn
(output omitted)
```

In the omitted output (which varies across operating systems), you might learn that there is an *rmargin #* tunable value, which specifies the right margin in inches. You could specify the *override\_option rmargin(#)* to temporarily override the default value, or you could type `translator set smcl2prn rmargin #` beforehand to permanently reset the value.

Alternatively, on some computers with some translators, you might discover that nothing can be set.

## Options for translate

`translator(tname)` specifies the name of the translator to be used to translate the file. The available translators are

| <i>tname</i>             | input                   | output                  |
|--------------------------|-------------------------|-------------------------|
| <code>smcl2ps</code>     | SMCL                    | PostScript              |
| <code>log2ps</code>      | Stata ASCII text log    | PostScript              |
| <code>txt2ps</code>      | generic ASCII text file | PostScript              |
| <code>Viewer2ps</code>   | Viewer window           | PostScript              |
| <code>Results2ps</code>  | Results window          | PostScript              |
| <code>smcl2prn</code>    | SMCL                    | default printer format  |
| <code>log2prn</code>     | Stata ASCII text log    | default printer format  |
| <code>txt2prn</code>     | generic ASCII text log  | default printer format  |
| <code>Results2prn</code> | Results window          | default printer format  |
| <code>Viewer2prn</code>  | Viewer window           | default printer format  |
| <code>smcl2txt</code>    | SMCL                    | generic ASCII text file |
| <code>smcl2log</code>    | SMCL                    | Stata ASCII text log    |
| <code>Results2txt</code> | Results window          | generic ASCII text file |
| <code>Viewer2txt</code>  | Viewer window           | generic ASCII text file |
| <code>smcl2pdf</code>    | SMCL                    | PDF (Mac only)          |
| <code>log2pdf</code>     | Stata ASCII text log    | PDF (Mac only)          |
| <code>txt2pdf</code>     | generic ASCII text log  | PDF (Mac only)          |
| <code>Results2pdf</code> | Results window          | PDF (Mac only)          |
| <code>Viewer2pdf</code>  | Viewer window           | PDF (Mac only)          |

If `translator()` is not specified, `translate` determines which translator to use from extensions of the filenames specified. Typing `translate myfile.smcl myfile.ps` would use the `smcl2ps` translator. Typing `translate myfile.smcl myfile.ps, translate(smcl2prn)` would override the default and use the `smcl2prn` translator.

Actually, when you type `translate a.b.c.d`, `translate` looks up *.b* in the `transmap` extension-synonym table. If *.b* is not found, the translator *b2d* is used. If *.b* is found in the table, the mapped extension is used (call it *b'*), and then the translator *b'2d* is used. For example,

| command                           | translator used                        |
|-----------------------------------|----------------------------------------|
| . translate myfile.smcl myfile.ps | smcl2ps                                |
| . translate myfile.odd myfile.ps  | odd2ps, which does not exist, so error |
| . transmap define .odd .txt       |                                        |
| . translate myfile.odd myfile.ps  | txt2ps                                 |

You can list the mappings that `translate` uses by typing `transmap query`.  
`name(windowname)` specifies which window to translate when translating a Viewer. The default is for Stata to translate the topmost Viewer. The `name()` option is ignored when translating the Results window.

The window name is located inside parentheses in the window title. For example, if the title for a Viewer window is *Viewer (#1) [help print]*, the name for the window is `#1`.

`override_options` override any of the default options of the specified or implied translator. To find out what you can override for, say, `log2ps`, type

```
. translator query log2ps
(output omitted)
```

In the omitted output, you might learn that there is an `rmargin #` tunable value, which, for `log2ps`, specifies the right margin in inches. You could specify the *override\_option* `rmargin(#)` to temporarily override the default value or type `translator set log2ps rmargin #` beforehand to permanently reset the value.

`replace` specifies that `filenameout` be replaced if it already exists.

## Remarks

Remarks are presented under the following headings:

- [Printing files](#)
- [Printing files, Mac and Windows](#)
- [Printing files, Unix](#)
- [Translating files from one format to another](#)

## Printing files

Printing should be easy; just type

```
. print mylog.smcl
. print mylog.log
```

You can use `print` to print SMCL files, ASCII text files, and even the contents of the Results and Viewer windows:

```
. print @Results
. print @Viewer
. print @Viewer, name(#2)
```

For information about printing and translating graph files, see [\[G\] graph print](#) and see [\[G\] graph export](#).



## Printing files, Mac and Windows

When you type `print`, you are using the same facility that you would be using if you had selected **Print** from the **File** menu. If you try to print a file that Stata does not know about, Stata will complain:

```
. print read.me
translator me2prn not found
(perhaps you need to specify the like() option)
r(111);
```

Then you could type

```
. print read.me, like(txt)
```

to indicate that you wanted `read.me` sent to the printer in the same fashion as if the file were named `readme.txt`, or you could type

```
. transmap define .me .txt
. print read.me
```

Here you are telling Stata once and for all that you want files ending in `.me` to be treated in the same way as files ending in `.txt`. Stata will remember this mapping, even across sessions. To clear the `.me` mapping, type

```
. transmap define .me
```

To see all the mappings, type

```
. transmap query
```

To print to a file, use the `translate` command, not `print`:

```
. translate mylog.smcl mylog.prn
```

`translate` prints to a file by using the Windows print driver when the new filename ends in `.prn`. Under Mac, the `prn` translators are the same as the `pdf` translators. We suggest that you simply use the `.pdf` file extension when printing to a file.

## Printing files, Unix

Stata assumes that you have a PostScript printer attached to your Unix computer and that the Unix command `lpr(1)` can be used to send PostScript files to it, but you can change this. On your Unix system, typing

```
mycomputer$ lpr < filename
```

may not be sufficient to print PostScript files. For instance, perhaps on your system you would need to type

```
mycomputer$ lpr -Plexmark < filename
```

or

```
mycomputer$ lpr -Plexmark filename
```

or something else. To set the print command to be `lpr -Plexmark filename` and to state that the printer expects to receive PostScript files, type

```
. printer define prn ps "lpr -Plexmark @"
```

To set the print command to `lpr -Plexmark < filename` and to state that the printer expects to receive ASCII text files, type

```
. printer define prn txt "lpr -Plexmark < @"
```

That is, just type the command necessary to send files to your printer and include an `@` sign where the filename should be substituted. Two file formats are available: `ps` and `txt`. The default setting, as shipped from the factory, is

```
. printer define prn ps "lpr < @"
```

We will return to the `printer` command in the technical note that follows because it has some other capabilities you should know about.

In any case, after you redefine the default printer, the following should just work:

```
. print mylog.smcl
. print mylog.log
```

If you try to print a file that Stata does not know about, it will complain:

```
. print read.me
translator me2prn not found
r(111);
```

Here you could type

```
. print read.me, like(txt)
```

to indicate that you wanted `read.me` sent to the printer in the same fashion as if the file were named `readme.txt`, or you could type

```
. transmap define .me .txt
. print read.me
```

Here you are telling Stata once and for all that you want files ending in `.me` to be treated in the same way as files ending in `.txt`. Stata will remember this setting for `.me`, even across sessions.

If you want to clear the `.me` setting, type

```
. transmap define .me
```

If you want to see all your settings, type

```
. transmap query
```

## □ Technical note

The syntax of the `printer` command is

```
printer define prntername [{ ps | txt } "Unix command with @"]
printer query [prntername]
```

You may define multiple printers. By default, `print` uses the printer named `prn`, but `print` has the syntax

```
print filename [, like(ext) printer(prntername) override_options]
```

so, if you define multiple printers, you may route your output to them.

For instance, if you have a second printer on your system, you might type

```
. printer define lexmark ps "lpr -Plexmark < @"
```

After doing that, you could type

```
. print myfile.smcl, printer(lexmark)
```

Any printers that you set will be remembered even across sessions. You can delete printers:

```
. printer define lexmark
```

You can list all the defined printers by typing `printer query`, and you can list the definition of a particular printer, say, `prn`, by typing `printer query prn`.

The default printer `prn` we have predefined for you is

```
. printer define prn ps "lpr < @"
```

meaning that we assume that it is a PostScript printer and that the Unix command `lpr(1)`, without options, is sufficient to cause files to print. Feel free to change the default definition. If you change it, the change will be remembered across sessions. □

## □ Technical note

Unix(GUI) users should note that X-Windows does not have the concept of a window z-order, which prevents Stata from determining which window is the topmost window. Instead, Stata determines which window is topmost based on which window has the focus. However, some window managers will set the focus to a window without bringing the window to the top. What Stata considers the topmost window may not appear topmost visually. For this reason, you should always use the `name()` option to ensure that the correct window is printed. □

## □ Technical note

When you select the Results window to print from the **Print** menu or toolbar button, the result is the same as if you were to issue the `print` command. When you select a Viewer window to print from the **Print** menu or toolbar button, the result is the same as if you were to issue the `print` command with a `name()` option. □

The translation to PostScript format is done by `translate` and, in particular, is performed by the translators `smcl2ps`, `log2ps`, and `txt2ps`. There are many tunable parameters in each of these translators. You can display the current values of these tunable parameters for, say, `smcl2ps` by typing

```
. translator query smcl2ps
(output omitted)
```

and you can set any of the tunable parameters (for instance, setting `smcl2ps`'s `rmargin` value to 1) by typing

```
. translator set smcl2ps rmargin 1
(output omitted)
```

Any settings you make will be remembered across sessions. You can reset `smcl2ps` to be as it was when Stata was shipped by typing

```
. translator reset smcl2ps
```

## Translating files from one format to another

If you have a SMCL log, which you might have created by previously typing `log using mylog`, you can translate it to an ASCII text log by typing

```
. translate myfile.smcl myfile.log
```

and you can translate it to a PostScript file by typing

```
. translate myfile.smcl myfile.ps
```

`translate` translates files from one format to another, and, in fact, `print` uses `translate` to produce a file suitable for sending to the printer.

When you type

```
. translate a.b c.d
```

`translate` looks for the predefined translator *b2d* and uses that to perform the translation. If there is a `transmap` synonym for *b*, however, the mapped value *b'* is used: *b'2d*.

Only certain translators exist, and they are listed under the description of the `translate()` option in *Options for translate* above, or you can type

```
. translator query
```

for a complete (and perhaps more up-to-date) list.

Anyway, `translate` forms the name *b2d* or *b'2d*, and if the translator does not exist, `translate` issues an error. With the `translator()` option, you can specify exactly which translator to use, and then it does not matter how your files are named.

The only other thing to know is that some translators have tunable parameters that affect how they perform their translation. You can type

```
. translator query translator_name
```

to find out what those parameters are. Some translators have no tunable parameters, and some have many:

```
. translator query smcl2ps
```

|                        |            |   |         |                        |         |
|------------------------|------------|---|---------|------------------------|---------|
| header                 | on         |   |         |                        |         |
| headertext             |            |   |         |                        |         |
| logo                   | on         |   |         |                        |         |
| user                   |            |   |         |                        |         |
| projecttext            |            |   |         |                        |         |
| cmdnumber              | on         |   |         |                        |         |
| fontsize               | 9          |   | lmargin | 1.00                   |         |
| pagesize               | letter     |   | rmargin | 1.00                   |         |
| pagewidth              | 8.50       |   | tmargin | 1.00                   |         |
| pageheight             | 11.00      |   | bmargin | 1.00                   |         |
| scheme                 | monochrome |   |         |                        |         |
| cust1_result_color     | 0          | 0 | 0       | cust2_result_color     | 0 0 0   |
| cust1_standard_color   | 0          | 0 | 0       | cust2_standard_color   | 0 0 0   |
| cust1_error_color      | 0          | 0 | 0       | cust2_error_color      | 255 0 0 |
| cust1_input_color      | 0          | 0 | 0       | cust2_input_color      | 0 0 0   |
| cust1_link_color       | 0          | 0 | 0       | cust2_link_color       | 0 0 255 |
| cust1_hilite_color     | 0          | 0 | 0       | cust2_hilite_color     | 0 0 0   |
| cust1_result_bold      | on         |   |         | cust2_result_bold      | on      |
| cust1_standard_bold    | off        |   |         | cust2_standard_bold    | off     |
| cust1_error_bold       | on         |   |         | cust2_error_bold       | on      |
| cust1_input_bold       | off        |   |         | cust2_input_bold       | off     |
| cust1_link_bold        | off        |   |         | cust2_link_bold        | off     |
| cust1_hilite_bold      | on         |   |         | cust2_hilite_bold      | on      |
| cust1_link_underline   | on         |   |         | cust2_link_underline   | on      |
| cust1_hilite_underline | off        |   |         | cust2_hilite_underline | off     |

You can temporarily override any setting by specifying the *setopt(setval)* option on the `translate` (or `print`) command. For instance, you can type

```
. translate ..., ... cmdnumber(off)
```

or you can reset the value permanently by typing

```
. translator set smcl2ps setopt setval
```

For instance,

```
. translator set smcl2ps cmdnumber off
```

If you reset a value, Stata will remember the change, even in future sessions.

Mac and Windows users: The `smcl2ps` (and the other `*2ps` translators) are not used by `print`, even when you have a PostScript printer attached to your computer. Instead, the Mac or Windows print driver is used. Resetting `smcl2ps` values will not affect printing; instead, you change the defaults in the Printers Control Panel in Windows and by selecting **Page Setup...** from the **File** menu in Mac. You can, however, `translate` files yourself using the `smcl2ps` translator and the other `*2ps` translators.

## Saved results

`transmap query .ext` saves in macro `r(suffix)` the mapped extension (without the leading period) or saves `ext` if the `ext` is not mapped.

`translator query translatorname` saves *setval* in macro `r(setopt)` for every *setopt*, *setval* pair.

`printer` query *printrname* (Unix only) saves in macro `r(suffix)` the “filetype” of the input that the printer expects (currently “ps” or “txt”) and, in macro `r(command)`, the command to send output to the printer.

## Methods and formulas

`print` is implemented as an ado-file.

## Also see

[R] [log](#) — Echo copy of session to file

[G] [graph export](#) — Export current graph

[G] [graph print](#) — Print a graph

[G] [graph set](#) — Set graphics options

[P] [smcl](#) — Stata Markup and Control Language

[U] [15 Saving and printing output—log files](#)

Syntax

Basic syntax

```
treatreg depvar [indepvars], treat(depvart = indepvarst) [twostep]
```

Full syntax for maximum likelihood estimates only

```
treatreg depvar [indepvars] [if] [in] [weight],
treat(depvart = indepvarst [, noconstant]) [treatreg_ml_options]
```

Full syntax for two-step consistent estimates only

```
treatreg depvar [indepvars] [if] [in],
treat(depvart = indepvarst [, noconstant]) twostep [treatreg_ts_options]
```

| treatreg_ml_options              | description                                                                                                      |
|----------------------------------|------------------------------------------------------------------------------------------------------------------|
| Model                            |                                                                                                                  |
| * <b>treat()</b>                 | equation for treatment effects                                                                                   |
| <b>noconstant</b>                | suppress constant term                                                                                           |
| <b>constraints</b> (constraints) | apply specified linear constraints                                                                               |
| <b>collinear</b>                 | keep collinear variables                                                                                         |
| SE/Robust                        |                                                                                                                  |
| <b>vce</b> (vcetype)             | vcetype may be oim, <b>robust</b> , <b>cluster</b> clustvar, <b>opg</b> , <b>bootstrap</b> , or <b>jackknife</b> |
| Reporting                        |                                                                                                                  |
| <b>level</b> (#)                 | set confidence level; default is level(95)                                                                       |
| <b>first</b>                     | report first-step probit estimates                                                                               |
| <b>noskip</b>                    | perform likelihood-ratio test                                                                                    |
| <b>hazard</b> (newvar)           | create newvar containing hazard from treatment equation                                                          |
| <b>nocnsreport</b>               | do not display constraints                                                                                       |
| <b>display_options</b>           | control spacing and display of omitted variables and base and empty cells                                        |
| Maximization                     |                                                                                                                  |
| <b>maximize_options</b>          | control maximization process; seldom used                                                                        |
| <sup>†</sup> <b>coeflegend</b>   | display coefficients' legend instead of coefficient table                                                        |

\***treat**(depvar<sub>t</sub> = indepvars<sub>t</sub> [, **noconstant**]) is required.

<sup>†</sup>**coeflegend** does not appear in the dialog box.

| <i>treatreg <u>ts_options</u></i> | description                                                                |
|-----------------------------------|----------------------------------------------------------------------------|
| Model                             |                                                                            |
| * <u>treat</u> ()                 | equation for treatment effects                                             |
| * <u>twostep</u>                  | produce two-step consistent estimate                                       |
| <u>noconstant</u>                 | suppress constant term                                                     |
| SE                                |                                                                            |
| <u>vce</u> ( <i>vcetype</i> )     | <i>vcetype</i> may be conventional, <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting                         |                                                                            |
| <u>level</u> (#)                  | set confidence level; default is level(95)                                 |
| <u>first</u>                      | report first-step probit estimates                                         |
| <u>hazard</u> ( <i>newvar</i> )   | create <i>newvar</i> containing hazard from treatment equation             |
| <i>display_options</i>            | control spacing and display of omitted variables and base and empty cells  |
| <sup>†</sup> <u>coeflegend</u>    | display coefficients' legend instead of coefficient table                  |

\*treat(*devar*<sub>*t*</sub> = *indepvars*<sub>*t*</sub> [ , noconstant ]) and twostep are required.

<sup>†</sup> coeflegend does not appear in the dialog box.

*indepvars* and *indepvars*<sub>*t*</sub> may contain factor variables; see [U] 11.4.3 Factor variables.

*devar*, *indepvars*, *devar*<sub>*t*</sub>, and *indepvars*<sub>*t*</sub> may contain time-series operators; see [U] 11.4.4 Time-series varlists.

bootstrap, by, jackknife, rolling, statsby, and svy are allowed; see [U] 11.1.10 Prefix commands.

Weights are not allowed with the bootstrap prefix.

aweights are not allowed with the jackknife prefix.

twostep, vce(), first, noskip, hazard(), and weights are not allowed with the svy prefix.

pweights, aweights, fweights, and iweights are allowed with maximum likelihood estimation; see [U] 11.1.6 weight. No weights are allowed if twostep is specified.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

treatreg for maximum likelihood estimates

Statistics > Sample-selection models > Treatment-effects model (ML)

treatreg for two-step consistent estimates

Statistics > Sample-selection models > Treatment-effects model (two-step)

Description

`treatreg` fits a treatment-effects model by using either a two-step consistent estimator or full maximum likelihood. The treatment-effects model considers the effect of an endogenously chosen binary treatment on another endogenous continuous variable, conditional on two sets of independent variables.



## Options for maximum likelihood estimates

### Model

`treat(depvart = indepvarst [, noconstant])` specifies the variables and options for the treatment equation. It is an integral part of specifying a treatment-effects model and is required.

`noconstant`, `constraints(constraints)`, `collinear`; see [\[R\] estimation options](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [\[R\] vce\\_option](#).

### Reporting

`level(#)`; see [\[R\] estimation options](#).

`first` specifies that the first-step probit estimates of the treatment equation be displayed before estimation.

`noskip` specifies that a full maximum-likelihood model with only a constant for the regression equation be fit. This model is not displayed but is used as the base model to compute a likelihood-ratio test for the model test statistic displayed in the estimation header. By default, the overall model test statistic is an asymptotically equivalent Wald test that all the parameters in the regression equation are zero (except the constant). For many models, this option can substantially increase estimation time.

`hazard(newvar)` will create a new variable containing the hazard from the treatment equation. The hazard is computed from the estimated parameters of the treatment equation.

`nocnsreport`; see [\[R\] estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [\[R\] estimation options](#).

### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [\[R\] maximize](#). These options are seldom used.

Setting the optimization type to `technique(bhhh)` resets the default `vcetype` to `vce(opg)`.

The following option is available with `treatreg` but is not shown in the dialog box:

`coeflegend`; see [\[R\] estimation options](#).

## Options for two-step consistent estimates

### Model

`treat(depvart = indepvarst [, noconstant])` specifies the variables and options for the treatment equation. It is an integral part of specifying a treatment-effects model and is required.

`twostep` specifies that two-step consistent estimates of the parameters, standard errors, and covariance matrix be produced, instead of the default maximum likelihood estimates.

`noconstant`; see [\[R\] estimation options](#).

## SE

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

`vce(conventional)`, the default, uses the conventionally derived variance estimator for the two-step estimator of the treatment-effects model.

## Reporting

`level(#)`; see [R] [estimation options](#).

`first` specifies that the first-step probit estimates of the treatment equation be displayed before estimation.

`hazard(newvar)` will create a new variable containing the hazard from the treatment equation. The hazard is computed from the estimated parameters of the treatment equation.

`display_options`: [noomitted](#), [vsquish](#), [noemptycells](#), [baselevels](#), [allbaselevels](#); see [R] [estimation options](#).

The following option is available with `treatreg` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

The treatment-effects model estimates the effect of an endogenous binary treatment,  $z_j$ , on a continuous, fully observed variable  $y_j$ , conditional on the independent variables  $x_j$  and  $w_j$ . The primary interest is in the regression function

$$y_j = \mathbf{x}_j\beta + \delta z_j + \epsilon_j$$

where  $z_j$  is an endogenous dummy variable indicating whether the treatment is assigned or not. The binary decision to obtain the treatment  $z_j$  is modeled as the outcome of an unobserved latent variable,  $z_j^*$ . It is assumed that  $z_j^*$  is a linear function of the exogenous covariates  $\mathbf{w}_j$  and a random component  $u_j$ . Specifically,

$$z_j^* = \mathbf{w}_j\gamma + u_j$$

and the observed decision is

$$z_j = \begin{cases} 1, & \text{if } z_j^* > 0 \\ 0, & \text{otherwise} \end{cases}$$

where  $\epsilon$  and  $u$  are bivariate normal with mean zero and covariance matrix

$$\begin{bmatrix} \sigma^2 & \rho\sigma \\ \rho\sigma & 1 \end{bmatrix}$$

There are many variations of this model in the literature. [Maddala \(1983\)](#) derives the maximum likelihood and two-step estimators of the version implemented here and also gives a brief review of several empirical applications of this model. [Barnow, Cain, and Goldberger \(1981\)](#) provide another useful derivation of this model. [Barnow, Cain, and Goldberger \(1981\)](#) concentrate on deriving the conditions for which the self-selection bias of the simple OLS estimator of the treatment effect,  $\delta$ , is nonzero and of a specific sign.

## ► Example 1

We will illustrate `treatreg` with part of the Mroz dataset distributed with [Berndt \(1996\)](#). This dataset contains 753 observations on women's labor supply. Our subsample is of 250 observations, with 150 market laborers and 100 nonmarket laborers.

```
. use http://www.stata-press.com/data/r11/labor
. describe
Contains data from http://www.stata-press.com/data/r11/labor.dta
obs: 250
vars: 15 18 Apr 2009 05:01
size: 16,000 (99.9% of memory free)
```

| variable name | storage type | display format | value label | variable label                         |
|---------------|--------------|----------------|-------------|----------------------------------------|
| lfp           | float        | %9.0g          |             | 1 if woman worked in 1975              |
| whrs          | float        | %9.0g          |             | wife's hours of work                   |
| kl6           | float        | %9.0g          |             | # of children younger than 6           |
| k618          | float        | %9.0g          |             | # of children between 6 and 18         |
| wa            | float        | %9.0g          |             | wife's age                             |
| we            | float        | %9.0g          |             | wife's education attainment            |
| ww            | float        | %9.0g          |             | wife's wage                            |
| hhrs          | float        | %9.0g          |             | husband's hours worked in 1975         |
| ha            | float        | %9.0g          |             | husband's age                          |
| he            | float        | %9.0g          |             | husband's educational attainment       |
| hw            | float        | %9.0g          |             | husband's wage                         |
| faminc        | float        | %9.0g          |             | family income                          |
| wmed          | float        | %9.0g          |             | wife's mother's educational attainment |
| wfed          | float        | %9.0g          |             | wife's father's educational attainment |
| cit           | float        | %9.0g          |             | 1 if live in large city                |

Sorted by:

```
. summarize
```

| Variable | Obs | Mean     | Std. Dev. | Min    | Max    |
|----------|-----|----------|-----------|--------|--------|
| lfp      | 250 | .6       | .4908807  | 0      | 1      |
| whrs     | 250 | 799.84   | 915.6035  | 0      | 4950   |
| kl6      | 250 | .236     | .5112234  | 0      | 3      |
| k618     | 250 | 1.364    | 1.370774  | 0      | 8      |
| wa       | 250 | 42.92    | 8.426483  | 30     | 60     |
| we       | 250 | 12.352   | 2.164912  | 5      | 17     |
| ww       | 250 | 2.27523  | 2.59775   | 0      | 14.631 |
| hhrs     | 250 | 2234.832 | 600.6702  | 768    | 5010   |
| ha       | 250 | 45.024   | 8.171322  | 30     | 60     |
| he       | 250 | 12.536   | 3.106009  | 3      | 17     |
| hw       | 250 | 7.494435 | 4.636192  | 1.0898 | 40.509 |
| faminc   | 250 | 23062.54 | 12923.98  | 3305   | 91044  |
| wmed     | 250 | 9.136    | 3.536031  | 0      | 17     |
| wfed     | 250 | 8.608    | 3.751082  | 0      | 17     |
| cit      | 250 | .624     | .4853517  | 0      | 1      |

We will assume that the wife went to college if her educational attainment is more than 12 years. Let `wc` be the dummy variable indicating whether the individual went to college. With this definition, our sample contains the following distribution of college education:

```
. generate wc = 0
. replace wc = 1 if we > 12
(69 real changes made)
. tab wc
```

| wc    | Freq. | Percent | Cum.   |
|-------|-------|---------|--------|
| 0     | 181   | 72.40   | 72.40  |
| 1     | 69    | 27.60   | 100.00 |
| Total | 250   | 100.00  |        |

We will model the wife’s wage as a function of her age, whether the family was living in a big city, and whether she went to college. An ordinary least-squares estimation produces the following results:

```
. regress ww wa cit wc
```

| Source   | SS         | df  | MS         | Number of obs = | 250    |
|----------|------------|-----|------------|-----------------|--------|
| Model    | 93.2398568 | 3   | 31.0799523 | F( 3, 246) =    | 4.82   |
| Residual | 1587.08776 | 246 | 6.45157627 | Prob > F =      | 0.0028 |
| Total    | 1680.32762 | 249 | 6.74830369 | R-squared =     | 0.0555 |
|          |            |     |            | Adj R-squared = | 0.0440 |
|          |            |     |            | Root MSE =      | 2.54   |

| ww    | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |
|-------|-----------|-----------|-------|-------|----------------------|
| wa    | -.0104985 | .0192667  | -0.54 | 0.586 | -.0484472 .0274502   |
| cit   | .1278922  | .3389058  | 0.38  | 0.706 | -.5396351 .7954195   |
| wc    | 1.332192  | .3644344  | 3.66  | 0.000 | .6143819 2.050001    |
| _cons | 2.278337  | .8432385  | 2.70  | 0.007 | .6174488 3.939225    |

Is 1.332 a consistent estimate of the marginal effect of a college education on wages? If individuals choose whether to attend college and the error term of the model that gives rise to this choice is correlated with the error term in the wage equation, then the answer is no. (See [Barnow, Cain, and Goldberger \[1981\]](#) for a good discussion of the existence and sign of selectivity bias.) We might suspect that individuals with higher abilities, either innate or due to the circumstances of their birth, would be more likely to go to college and to earn higher wages. Such ability is, of course, unobserved. Furthermore, if the error term in our model for going to college is correlated with ability, and the error term in our wage equation is correlated with ability, the two terms should be positively correlated. These conditions make the problem of signing the selectivity bias equivalent to an omitted-variable problem. In the case at hand, because we would expect the correlation between the omitted variable and a college education to be positive, we suspect that OLS is biased upward.

To account for the bias, we fit the treatment-effects model. We model the wife’s college decision as a function of her mother’s and her father’s educational attainment. Thus we are interested in fitting the model

$$ww = \beta_0 + \beta_1 wa + \beta_2 cit + \delta wc + \epsilon$$
$$wc^* = \gamma_0 + \gamma_1 wmed + \gamma_2 wfed + u$$

where

$$wc = \begin{cases} 1, & wc^* > 0, \text{ i.e., wife went to college} \\ 0, & \text{otherwise} \end{cases}$$

and where  $\epsilon$  and  $u$  have a bivariate normal distribution with zero mean and covariance matrix

$$\begin{bmatrix} \sigma^2 & \rho\sigma \\ \rho\sigma & 1 \end{bmatrix}$$

The following output gives the maximum likelihood estimates of the parameters of this model:

```
. treatreg ww wa cit, treat(wc=wmed wfed)
Iteration 0: log likelihood = -707.07237
Iteration 1: log likelihood = -707.07215
Iteration 2: log likelihood = -707.07215

Treatment-effects model -- MLE Number of obs = 250
 Wald chi2(3) = 4.11
Log likelihood = -707.07215 Prob > chi2 = 0.2501
```

|                                                                                 | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|---------------------------------------------------------------------------------|-----------|-----------|-------|-------|----------------------|-----------|
| <b>ww</b>                                                                       |           |           |       |       |                      |           |
| wa                                                                              | -.0110424 | .0199652  | -0.55 | 0.580 | -.0501735            | .0280887  |
| cit                                                                             | .127636   | .3361938  | 0.38  | 0.704 | -.5312917            | .7865638  |
| wc                                                                              | 1.271327  | .7412951  | 1.72  | 0.086 | -.1815842            | 2.724239  |
| _cons                                                                           | 2.318638  | .9397573  | 2.47  | 0.014 | .4767478             | 4.160529  |
| <b>wc</b>                                                                       |           |           |       |       |                      |           |
| wmed                                                                            | .1198055  | .0320056  | 3.74  | 0.000 | .0570757             | .1825352  |
| wfed                                                                            | .0961886  | .0290868  | 3.31  | 0.001 | .0391795             | .1531977  |
| _cons                                                                           | -2.631876 | .3309128  | -7.95 | 0.000 | -3.280453            | -1.983299 |
| /athrho                                                                         | .0178668  | .1899898  | 0.09  | 0.925 | -.3545063            | .3902399  |
| /lnsigma                                                                        | .9241584  | .0447455  | 20.65 | 0.000 | .8364588             | 1.011858  |
| rho                                                                             | .0178649  | .1899291  |       |       | -.3403659            | .371567   |
| sigma                                                                           | 2.519747  | .1127473  |       |       | 2.308179             | 2.750707  |
| lambda                                                                          | .0450149  | .4786442  |       |       | -.8931105            | .9831404  |
| LR test of indep. eqns. (rho = 0):   chi2(1) =      0.01   Prob > chi2 = 0.9251 |           |           |       |       |                      |           |

In the input, we specified that the continuous dependent variable, `ww` (wife's wage), is a linear function of `cit` and `wa`. Note the syntax for the treatment variable. The treatment `wc` is not included in the first variable list; it is specified in the `treat()` option. In this example, `wmed` and `wfed` are specified as the exogenous covariates in the treatment equation.

The output has the form of many two-equation estimators in Stata. We note that our conjecture that the OLS estimate was biased upward is verified. But it is perhaps more interesting that the size of the bias is negligible, and the likelihood-ratio test at the bottom of the output indicates that we cannot reject the null hypothesis that the two error terms are uncorrelated. This result might be due to several specification errors. We ignored the selectivity bias due to the endogeneity of entering the labor market. We have also written both the wage equation and the college-education equation in linear form, ignoring any higher power terms or interactions.

The results for the two ancillary parameters require explanation. For numerical stability during optimization, `treatreg` does not directly estimate  $\rho$  or  $\sigma$ . Instead, `treatreg` estimates the inverse hyperbolic tangent of  $\rho$ ,

$$\operatorname{atanh} \rho = \frac{1}{2} \ln \left( \frac{1 + \rho}{1 - \rho} \right)$$

and  $\ln \sigma$ . Also, `treatreg` reports  $\lambda = \rho \sigma$ , along with an estimate of the standard error of the estimate and confidence interval.

➤ Example 2

Stata also produces a two-step estimator of the model with the `twostep` option. Maximum likelihood estimation of the parameters can be time consuming with large datasets, and the two-step estimates may provide a good alternative in such cases. Continuing with the women’s wage model, we can obtain the two-step estimates with consistent covariance estimates by typing

```
. treatreg ww wa cit, treat(wc=wmed wfed) twostep
Treatment-effects model -- two-step estimates Number of obs = 250
 Wald chi2(3) = 3.67
 Prob > chi2 = 0.2998
```

|        | Coef.  | Std. Err. | z        | P> z  | [95% Conf. Interval] |                     |
|--------|--------|-----------|----------|-------|----------------------|---------------------|
| ww     | wa     | -.0111623 | .020152  | -0.55 | 0.580                | -.0506594 .0283348  |
|        | cit    | .1276102  | .33619   | 0.38  | 0.704                | -.53131 .7865305    |
|        | wc     | 1.257995  | .8007428 | 1.57  | 0.116                | -.3114319 2.827422  |
|        | _cons  | 2.327482  | .9610271 | 2.42  | 0.015                | .4439031 4.21106    |
|        |        |           |          |       |                      |                     |
| wc     | wmed   | .1198888  | .0319862 | 3.75  | 0.000                | .0571971 .1825806   |
|        | wfed   | .0960764  | .0290583 | 3.31  | 0.001                | .0391233 .1530295   |
|        | _cons  | -2.631496 | .3308389 | -7.95 | 0.000                | -3.279928 -1.983063 |
|        |        |           |          |       |                      |                     |
| hazard | lambda | .0548738  | .5283928 | 0.10  | 0.917                | -.9807571 1.090505  |
|        |        |           |          |       |                      |                     |
|        | rho    | 0.02178   |          |       |                      |                     |
|        | sigma  | 2.5198211 |          |       |                      |                     |
|        | lambda | .05487379 | .5283928 |       |                      |                     |

The reported `lambda` ( $\lambda$ ) is the parameter estimate on the hazard from the augmented regression, which is derived in [Maddala \(1983\)](#) and presented in [Methods and formulas](#) below.

□ Technical note

The difference in expected earnings between participants and nonparticipants is

$$E(y_j \mid z_j = 1) - E(y_j \mid z_j = 0) = \delta + \rho\sigma \left[ \frac{\phi(\mathbf{w}_j\gamma)}{\Phi(\mathbf{w}_j\gamma)\{1 - \Phi(\mathbf{w}_j\gamma)\}} \right]$$

where  $\phi$  is the standard normal density and  $\Phi$  is the standard normal cumulative distribution function. If the correlation between the error terms,  $\rho$ , is zero, the problem reduces to one estimable by OLS and the difference is simply  $\delta$ . Because  $\rho$  is positive in the example, least squares overestimates the treatment effect,  $\delta$ .

## Saved results

`treatreg` (maximum likelihood) saves the following in `e()`:

### Scalars

|                            |                                                |
|----------------------------|------------------------------------------------|
| <code>e(N)</code>          | number of observations                         |
| <code>e(k)</code>          | number of parameters                           |
| <code>e(k_eq)</code>       | number of equations                            |
| <code>e(k_eq_model)</code> | number of equations in model Wald test         |
| <code>e(k_aux)</code>      | number of auxiliary parameters                 |
| <code>e(k_dv)</code>       | number of dependent variables                  |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints |
| <code>e(df_m)</code>       | model degrees of freedom                       |
| <code>e(ll)</code>         | log likelihood                                 |
| <code>e(N_clust)</code>    | number of clusters                             |
| <code>e(lambda)</code>     | $\lambda$                                      |
| <code>e(selambda)</code>   | standard error of $\lambda$                    |
| <code>e(sigma)</code>      | estimate of sigma                              |
| <code>e(chi2)</code>       | $\chi^2$                                       |
| <code>e(chi2_c)</code>     | $\chi^2$ for comparison test                   |
| <code>e(p_c)</code>        | $p$ -value for comparison test                 |
| <code>e(p)</code>          | significance                                   |
| <code>e(rho)</code>        | $\rho$                                         |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                      |
| <code>e(ic)</code>         | number of iterations                           |
| <code>e(rc)</code>         | return code                                    |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                    |

*(Continued on next page)*

## Macros

|                                 |                                                                                 |
|---------------------------------|---------------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>treatreg</code>                                                           |
| <code>e(cmdline)</code>         | command as typed                                                                |
| <code>e(depvar)</code>          | name of dependent variable                                                      |
| <code>e(hazard)</code>          | variable containing hazard                                                      |
| <code>e(wtype)</code>           | weight type                                                                     |
| <code>e(wexp)</code>            | weight expression                                                               |
| <code>e(title)</code>           | title in estimation output                                                      |
| <code>e(clustvar)</code>        | name of cluster variable                                                        |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                         |
| <code>e(chi2_ct)</code>         | Wald or LR; type of model $\chi^2$ test corresponding to <code>e(chi2_c)</code> |
| <code>e(vce)</code>             | <i>vctype</i> specified in <code>vce()</code>                                   |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                                   |
| <code>e(diparm#)</code>         | display transformed parameter #                                                 |
| <code>e(opt)</code>             | type of optimization                                                            |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization        |
| <code>e(method)</code>          | ml                                                                              |
| <code>e(ml_method)</code>       | type of ml method                                                               |
| <code>e(user)</code>            | name of likelihood-evaluator program                                            |
| <code>e(technique)</code>       | maximization technique                                                          |
| <code>e(singularHmethod)</code> | n-marquardt or hybrid; method used when Hessian is singular                     |
| <code>e(crittype)</code>        | optimization criterion                                                          |
| <code>e(properties)</code>      | b v                                                                             |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                                  |
| <code>e(footnote)</code>        | program used to implement the footnote display                                  |
| <code>e(marginsok)</code>       | predictions allowed by <code>margins</code>                                     |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>                  |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>                  |

## Matrices

|                              |                                                                 |
|------------------------------|-----------------------------------------------------------------|
| <code>e(b)</code>            | coefficient vector                                              |
| <code>e(Cns)</code>          | constraints matrix                                              |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)                             |
| <code>e(gradient)</code>     | gradient vector                                                 |
| <code>e(V)</code>            | variance–covariance matrix of the estimators                    |
| <code>e(V_modelbased)</code> | model-based variance                                            |
| <code>e(ml_h)</code>         | derivative tolerance, $(\text{abs}(b)+1\text{e-}3)*1\text{e-}3$ |
| <code>e(ml_scale)</code>     | derivative scale factor                                         |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|



`treatreg` (two-step) saves the following in `e()`:

#### Scalars

|                          |                             |
|--------------------------|-----------------------------|
| <code>e(N)</code>        | number of observations      |
| <code>e(df_m)</code>     | model degrees of freedom    |
| <code>e(lambda)</code>   | $\lambda$                   |
| <code>e(selambda)</code> | standard error of $\lambda$ |
| <code>e(sigma)</code>    | estimate of sigma           |
| <code>e(chi2)</code>     | $\chi^2$                    |
| <code>e(p)</code>        | significance                |
| <code>e(rho)</code>      | $\rho$                      |
| <code>e(rank)</code>     | rank of $e(V)$              |

#### Macros

|                                |                                                                |
|--------------------------------|----------------------------------------------------------------|
| <code>e(cmd)</code>            | <code>treatreg</code>                                          |
| <code>e(cmdline)</code>        | command as typed                                               |
| <code>e(depvar)</code>         | name of dependent variable                                     |
| <code>e(title)</code>          | title in estimation output                                     |
| <code>e(chi2type)</code>       | Wald or LR; type of model $\chi^2$ test                        |
| <code>e(vce)</code>            | <i>vcetype</i> specified in <code>vce()</code>                 |
| <code>e(vcetype)</code>        | title used to label Std. Err.                                  |
| <code>e(margins_prolog)</code> | program to prepare estimates for use by <code>margins</code>   |
| <code>e(margins_epilog)</code> | program to restore estimates after use by <code>margins</code> |
| <code>e(method)</code>         | ml or twostep                                                  |
| <code>e(properties)</code>     | b V                                                            |
| <code>e(predict)</code>        | program used to implement <code>predict</code>                 |
| <code>e(footnote)</code>       | program used to implement the footnote display                 |
| <code>e(marginsok)</code>      | predictions allowed by <code>margins</code>                    |
| <code>e(marginsnotok)</code>   | predictions disallowed by <code>margins</code>                 |
| <code>e(asbalanced)</code>     | factor variables <code>fvset</code> as <code>asbalanced</code> |
| <code>e(asobserved)</code>     | factor variables <code>fvset</code> as <code>asobserved</code> |

#### Matrices

|                   |                                              |
|-------------------|----------------------------------------------|
| <code>e(b)</code> | coefficient vector                           |
| <code>e(V)</code> | variance–covariance matrix of the estimators |

#### Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`treatreg` is implemented as an ado-file. [Maddala \(1983, 117–122\)](#) derives both the maximum likelihood and the two-step estimator implemented here. [Greene \(2008, 889–890\)](#) also provides an introduction to the treatment-effects model.

The primary regression equation of interest is

$$y_j = \mathbf{x}_j\boldsymbol{\beta} + \delta z_j + \epsilon_j$$

where  $z_j$  is a binary decision variable that is assumed to stem from an unobservable latent variable:

$$z_j^* = \mathbf{w}_j\boldsymbol{\gamma} + u_j$$

The decision to obtain the treatment is made according to the rule

$$z_j = \begin{cases} 1, & \text{if } z_j^* > 0 \\ 0, & \text{otherwise} \end{cases}$$

where  $\epsilon$  and  $u$  are bivariate normal with mean zero and covariance matrix

$$\begin{bmatrix} \sigma^2 & \rho\sigma \\ \rho\sigma & 1 \end{bmatrix}$$

The likelihood function for this model is given in [Maddala \(1983, 122\)](#). [Greene \(2000, 180\)](#) discusses the standard method of reducing a bivariate normal to a function of a univariate normal and the correlation  $\rho$ . The following is the log likelihood for observation  $j$ ,

$$\ln L_j = \begin{cases} \ln \Phi \left\{ \frac{\mathbf{w}_j \boldsymbol{\gamma} + (y_j - \mathbf{x}_j \boldsymbol{\beta} - \delta) \rho / \sigma}{\sqrt{1 - \rho^2}} \right\} - \frac{1}{2} \left( \frac{y_j - \mathbf{x}_j \boldsymbol{\beta} - \delta}{\sigma} \right)^2 - \ln(\sqrt{2\pi} \sigma), & z_j = 1 \\ \ln \Phi \left\{ \frac{-\mathbf{w}_j \boldsymbol{\gamma} - (y_j - \mathbf{x}_j \boldsymbol{\beta}) \rho / \sigma}{\sqrt{1 - \rho^2}} \right\} - \frac{1}{2} \left( \frac{y_j - \mathbf{x}_j \boldsymbol{\beta}}{\sigma} \right)^2 - \ln(\sqrt{2\pi} \sigma), & z_j = 0 \end{cases}$$

where  $\Phi(\cdot)$  is the cumulative distribution function of the standard normal distribution.

In the maximum likelihood estimation,  $\sigma$  and  $\rho$  are not directly estimated. Rather  $\ln \sigma$  and  $\operatorname{atanh} \rho$  are directly estimated, where

$$\operatorname{atanh} \rho = \frac{1}{2} \ln \left( \frac{1 + \rho}{1 - \rho} \right)$$

The standard error of  $\lambda = \rho\sigma$  is approximated through the delta method, which is given by

$$\operatorname{Var}(\lambda) \approx \mathbf{D} \operatorname{Var}\{(\operatorname{atanh} \rho \quad \ln \sigma)\} \mathbf{D}'$$

where  $\mathbf{D}$  is the Jacobian of  $\lambda$  with respect to  $\operatorname{atanh} \rho$  and  $\ln \sigma$ .

With maximum likelihood estimation, this command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [\[P\] \\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

The maximum likelihood version of `treatreg` also supports estimation with survey data. For details on VCEs with survey data, see [\[SVY\] variance estimation](#).

[Maddala \(1983, 120–122\)](#) also derives the two-step estimator. In the first stage, probit estimates are obtained of the treatment equation

$$\Pr(z_j = 1 \mid \mathbf{w}_j) = \Phi(\mathbf{w}_j \boldsymbol{\gamma})$$

From these estimates, the hazard,  $h_j$ , for each observation  $j$  is computed as

$$h_j = \begin{cases} \phi(\mathbf{w}_j \hat{\boldsymbol{\gamma}}) / \Phi(\mathbf{w}_j \hat{\boldsymbol{\gamma}}), & z_j = 1 \\ -\phi(\mathbf{w}_j \hat{\boldsymbol{\gamma}}) / \{1 - \Phi(\mathbf{w}_j \hat{\boldsymbol{\gamma}})\}, & z_j = 0 \end{cases}$$

where  $\phi$  is the standard normal density function. If

$$d_j = h_j(h_j + \hat{\boldsymbol{\gamma}} \mathbf{w}_j)$$

then

$$E(y_j | z_j) = \mathbf{x}_j\boldsymbol{\beta} + \delta\mathbf{z}_j + \rho\sigma h_j$$

$$\text{Var}(y_j | z_j) = \sigma^2 (1 - \rho^2 d_j)$$

The two-step parameter estimates of  $\boldsymbol{\beta}$  and  $\delta$  are obtained by augmenting the regression equation with the hazard  $h$ . Thus the regressors become  $[\mathbf{x} \ \mathbf{z} \ h]$ , and the additional parameter estimate  $\beta_h$  is obtained on the variable containing the hazard. A consistent estimate of the regression disturbance variance is obtained using the residuals from the augmented regression and the parameter estimate on the hazard

$$\hat{\sigma}^2 = \frac{\mathbf{e}'\mathbf{e} + \beta_h^2 \sum_{j=1}^N d_j}{N}$$

The two-step estimate of  $\rho$  is then

$$\hat{\rho} = \frac{\beta_h}{\hat{\sigma}}$$

To understand how the consistent estimates of the coefficient covariance matrix based on the augmented regression are derived, let  $\mathbf{A} = [\mathbf{x} \ \mathbf{z} \ h]$  and  $\mathbf{D}$  be a square diagonal matrix of size  $N$  with  $(1 - \hat{\rho}^2 d_j)$  on the diagonal elements. The conventional VCE is

$$\mathbf{V}_{\text{twostep}} = \hat{\sigma}^2 (\mathbf{A}'\mathbf{A})^{-1} (\mathbf{A}'\mathbf{D}\mathbf{A} + \mathbf{Q}) (\mathbf{A}'\mathbf{A})^{-1}$$

where

$$\mathbf{Q} = \hat{\rho}^2 (\mathbf{A}'\mathbf{D}\mathbf{A}) \mathbf{V}_p (\mathbf{A}'\mathbf{D}\mathbf{A})$$

and  $\mathbf{V}_p$  is the variance–covariance estimate from the probit estimation of the treatment equation.

## References

- Barnow, B. S., G. G. Cain, and A. S. Goldberger. 1981. Issues in the analysis of selectivity bias. In Vol. 5 of *Evaluation Studies Review Annual*, ed. E. W. Stromsdorfer and G. Farkas, 123–126. Beverly Hills: Sage.
- Berndt, E. R. 1996. *The Practice of Econometrics: Classic and Contemporary*. New York: Addison–Wesley.
- Cong, R., and D. M. Drukker. 2000. [sg141: Treatment effects model](#). *Stata Technical Bulletin* 55: 25–33. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 159–169. College Station, TX: Stata Press.
- Greene, W. H. 2000. *Econometric Analysis*. 4th ed. Upper Saddle River, NJ: Prentice–Hall.
- . 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.
- Maddala, G. S. 1983. *Limited-Dependent and Qualitative Variables in Econometrics*. Cambridge: Cambridge University Press.
- Nannicini, T. 2007. [Simulation-based sensitivity analysis for matching estimators](#). *Stata Journal* 7: 334–350.
- Nichols, A. 2007. [Causal inference with observational data](#). *Stata Journal* 7: 507–541.

## Also see

- [R] [treatreg postestimation](#) — Postestimation tools for treatreg
- [R] [heckman](#) — Heckman selection model
- [R] [probit](#) — Probit regression
- [R] [regress](#) — Linear regression
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `treatreg`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code> <sup>1</sup>  | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code> <sup>2</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code> <sup>1</sup>  | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `estat ic` and `suest` are not appropriate after `treatreg`, `twostep`.

<sup>2</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Syntax for predict

After *ML* or *twostep*

```
predict [type] newvar [if] [in] [, statistic]
```

After *ML*

```
predict [type] { stub*|newvarreg newvartreat newvarathrho newvarlnsigma }
[if] [in] , scores
```

| <i>statistic</i>      | <i>description</i>                                             |
|-----------------------|----------------------------------------------------------------|
| Main                  |                                                                |
| <code>xb</code>       | linear prediction; the default                                 |
| <code>stdp</code>     | standard error of the prediction                               |
| <code>stdf</code>     | standard error of the forecast                                 |
| <code>yctrtr</code>   | $E(y_j \mid \text{treatment} = 1)$                             |
| <code>ycntrtr</code>  | $E(y_j \mid \text{treatment} = 0)$                             |
| <code>ptrtr</code>    | $\Pr(\text{treatment} = 1)$                                    |
| <code>xbtrtr</code>   | linear prediction for treatment equation                       |
| <code>stdptrtr</code> | standard error of the linear prediction for treatment equation |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

`stdf` is not allowed with `svy` estimation results.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

`xb`, the default, calculates the linear prediction,  $\mathbf{x}_j\mathbf{b}$ .

`stdp` calculates the standard error of the prediction, which can be thought of as the standard error of the predicted expected value or mean for the observation's covariate pattern. The standard error of the prediction is also referred to as the standard error of the fitted value.

`stdf` calculates the standard error of the forecast, which is the standard error of the point prediction for one observation. It is commonly referred to as the standard error of the future or forecast value. By construction, the standard errors produced by `stdf` are always larger than those produced by `stdp`; see [Methods and formulas](#) in [R] [regress postestimation](#).

`yctrtr` calculates the expected value of the dependent variable conditional on the presence of the treatment:  $E(y_j \mid \text{treatment} = 1)$ .

`ycntrtr` calculates the expected value of the dependent variable conditional on the absence of the treatment:  $E(y_j \mid \text{treatment} = 0)$ .

`ptrtr` calculates the probability of the presence of the treatment:  
 $\Pr(\text{treatment} = 1) = \Pr(\mathbf{w}_j\boldsymbol{\gamma} + u_j > 0)$ .

`xbtrtr` calculates the linear prediction for the treatment equation.

`stdptrtr` calculates the standard error of the linear prediction for the treatment equation.

`scores`, not available with `twostep`, calculates equation-level score variables.

The first new variable will contain  $\partial \ln L / \partial (\mathbf{x}_j\boldsymbol{\beta})$ .

The second new variable will contain  $\partial \ln L / \partial (\mathbf{w}_j\boldsymbol{\gamma})$ .

The third new variable will contain  $\partial \ln L / \partial \text{atanh } \rho$ .

The fourth new variable will contain  $\partial \ln L / \partial \ln \sigma$ .

# Remarks

## ► Example 1

The default statistic produced by `predict` after `treatreg` is the expected value of the dependent variable from the underlying distribution of the regression model. For [example 1](#) in [\[R\] treatreg](#), this model is

$$ww = \beta_0 + \beta_1 wa + \beta_2 cit + \delta wc + \epsilon$$

Several other interesting aspects of the treatment-effects model can be explored with `predict`. We continue with our wage model, the wife’s expected wage, conditional on attending college, can be obtained with the `ycntrt` option. The wife’s expected wages, conditional on not attending college, can be obtained with the `ycntrt` option. Thus the difference in expected wages between participants and nonparticipants is the difference between `ycntrt` and `ycntrt`. For the case at hand, we have the following calculation:

```
. predict wwcntrt, ycntrt
. predict wwcntrt, ycntrt
. generate diff = wwcntrt - wwcntrt
. summarize diff
```

| Variable | Obs | Mean     | Std. Dev. | Min     | Max      |
|----------|-----|----------|-----------|---------|----------|
| diff     | 250 | 1.356912 | .0134202  | 1.34558 | 1.420173 |



# Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

- [\[R\] treatreg](#) — Treatment-effects model
- [\[U\] 20 Estimation and postestimation commands](#)

Syntax

truncreg *depvar* [*indepvars*] [*if*] [*in*] [*weight*] [, *options*]

| <i>options</i>                               | description                                                                                                                                                                 |
|----------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model                                        |                                                                                                                                                                             |
| <code>noconstant</code>                      | suppress constant term                                                                                                                                                      |
| <code>ll(<i>varname</i>   #)</code>          | lower limit for left truncation                                                                                                                                             |
| <code>ul(<i>varname</i>   #)</code>          | upper limit for right truncation                                                                                                                                            |
| <code>offset(<i>varname</i>)</code>          | include <i>varname</i> in model with coefficient constrained to 1                                                                                                           |
| <code>constraints(<i>constraints</i>)</code> | apply specified linear constraints                                                                                                                                          |
| <code>collinear</code>                       | keep collinear variables                                                                                                                                                    |
| SE/Robust                                    |                                                                                                                                                                             |
| <code>vce(<i>vcetype</i>)</code>             | <i>vcetype</i> may be <code>oim</code> , <code>robust</code> , <code>cluster <i>clustvar</i></code> , <code>opg</code> , <code>bootstrap</code> , or <code>jackknife</code> |
| Reporting                                    |                                                                                                                                                                             |
| <code>level(#)</code>                        | set confidence level; default is <code>level(95)</code>                                                                                                                     |
| <code>noskip</code>                          | perform likelihood-ratio test                                                                                                                                               |
| <code>nocnsreport</code>                     | do not display constraints                                                                                                                                                  |
| <code>display_options</code>                 | control spacing and display of omitted variables and base and empty cells                                                                                                   |
| Maximization                                 |                                                                                                                                                                             |
| <code>maximize_options</code>                | control maximization process; seldom used                                                                                                                                   |
| <sup>†</sup> <code>coeflegend</code>         | display coefficients' legend instead of coefficient table                                                                                                                   |

<sup>†</sup>`coeflegend` does not appear in the dialog box.

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

`bootstrap`, `by`, `jackknife`, `mi estimate`, `rolling`, `statsby`, and `svy` are allowed; see [U] 11.1.10 Prefix commands.

`vce(bootstrap)` and `vce(jackknife)` are not allowed with the `mi estimate` prefix.

Weights are not allowed with the `bootstrap` prefix.

`aweight`s are not allowed with the `jackknife` prefix.

`vce()`, `noskip`, and weights are not allowed with the `svy` prefix.

`aweight`s, `fweight`s, `iweight`s, and `pweight`s are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Linear models and related > Truncated regression

## Description

**truncreg** fits a regression model of *depvar* on *varlist* from a sample drawn from a restricted part of the population. Under the normality assumption for the whole population, the error terms in the truncated regression model have a truncated normal distribution, which is a normal distribution that has been scaled upward so that the distribution integrates to one over the restricted range.

## Options

### Model

**noconstant**; see [\[R\] estimation options](#).

**ll**(*varname* | #) and **ul**(*varname* | #) indicate the lower and upper limits for truncation, respectively.

You may specify one or both. Observations with *depvar*  $\leq$  **ll**() are left-truncated, observations with *depvar*  $\geq$  **ul**() are right-truncated, and the remaining observations are not truncated. See [\[R\] tobit](#) for a more detailed description.

**offset**(*varname*), **constraints**(*constraints*), **collinear**; see [\[R\] estimation options](#).

### SE/Robust

**vce**(*vcetype*) specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [\[R\] vce\\_option](#).

### Reporting

**level**(#); see [\[R\] estimation options](#).

**noskip** specifies that a full maximum-likelihood model with only a constant for the regression equation be fit. This model is not displayed but is used as the base model to compute a likelihood-ratio test for the model test statistic displayed in the estimation header. By default, the overall model test statistic is an asymptotically equivalent Wald test of all the parameters in the regression equation being zero (except the constant). For many models, this option can substantially increase estimation time.

**nocnsreport**; see [\[R\] estimation options](#).

**display\_options**: **noomitted**, **vsquish**, **noemptycells**, **baselevels**, **allbaselevels**; see [\[R\] estimation options](#).

### Maximization

**maximize\_options**: **difficult**, **technique**(*algorithm\_spec*), **iterate**(#), **[no]log**, **trace**, **gradient**, **showstep**, **hessian**, **showtolerance**, **tolerance**(#), **ltolerance**(#), **nrtolerance**(#), **nonrtolerance**, **from**(*init\_specs*); see [\[R\] maximize](#). These options are seldom used, but you may use the **ltol**(#) option to relax the convergence criterion; the default is 1e-6 during specification searches.

Setting the optimization type to **technique**(bhhh) resets the default *vcetype* to **vce**(opg).

The following option is available with **truncreg** but is not shown in the dialog box:

**coeflegend**; see [\[R\] estimation options](#).



## Remarks

Truncated regression fits a model of a dependent variable on independent variables from a restricted part of a population. Truncation is essentially a characteristic of the distribution from which the sample data are drawn. If  $x$  has a normal distribution with mean  $\mu$  and standard deviation  $\sigma$ , the density of the truncated normal distribution is

$$\begin{aligned} f(x \mid a < x < b) &= \frac{f(x)}{\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)} \\ &= \frac{\frac{1}{\sigma}\phi\left(\frac{x-\mu}{\sigma}\right)}{\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)} \end{aligned}$$

where  $\phi$  and  $\Phi$  are the density and distribution functions of the standard normal distribution.

Compared with the mean of the untruncated variable, the mean of the truncated variable is greater if the truncation is from below, and the mean of the truncated variable is smaller if the truncation is from above. Moreover, truncation reduces the variance compared with the variance in the untruncated distribution.

### ► Example 1

We will demonstrate `truncreg` with part of the Mroz dataset distributed with [Berndt \(1996\)](#). This dataset contains 753 observations on women's labor supply. Our subsample is of 250 observations, with 150 market laborers and 100 nonmarket laborers.

```
. use http://www.stata-press.com/data/r11/laborsub
. describe
Contains data from http://www.stata-press.com/data/r11/laborsub.dta
 obs: 250
 vars: 6 25 Sep 2008 18:36
 size: 2,750 (99.9% of memory free)
```

| variable name | storage<br>type | display<br>format | value<br>label | variable label                 |
|---------------|-----------------|-------------------|----------------|--------------------------------|
| lfp           | byte            | %9.0g             |                | 1 if woman worked in 1975      |
| whrs          | int             | %9.0g             |                | Wife's hours of work           |
| kl6           | byte            | %9.0g             |                | # of children younger than 6   |
| k618          | byte            | %9.0g             |                | # of children between 6 and 18 |
| wa            | byte            | %9.0g             |                | Wife's age                     |
| we            | byte            | %9.0g             |                | Wife's educational attainment  |

Sorted by:

```
. summarize, sep(0)
```

| Variable | Obs | Mean   | Std. Dev. | Min | Max  |
|----------|-----|--------|-----------|-----|------|
| lfp      | 250 | .6     | .4908807  | 0   | 1    |
| whrs     | 250 | 799.84 | 915.6035  | 0   | 4950 |
| kl6      | 250 | .236   | .5112234  | 0   | 3    |
| k618     | 250 | 1.364  | 1.370774  | 0   | 8    |
| wa       | 250 | 42.92  | 8.426483  | 30  | 60   |
| we       | 250 | 12.352 | 2.164912  | 5   | 17   |

We first perform ordinary least-squares estimation on the market laborers.

```
. regress whrs k16 k618 wa we if whrs > 0
```

| Source   | SS         | df  | MS         |
|----------|------------|-----|------------|
| Model    | 7326995.15 | 4   | 1831748.79 |
| Residual | 94793104.2 | 145 | 653745.546 |
| Total    | 102120099  | 149 | 685369.794 |

```
Number of obs = 150
F(4, 145) = 2.80
Prob > F = 0.0281
R-squared = 0.0717
Adj R-squared = 0.0461
Root MSE = 808.55
```

| whrs  | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|-------|-----------|-----------|-------|-------|----------------------|-----------|
| kl6   | -421.4822 | 167.9734  | -2.51 | 0.013 | -753.4748            | -89.48953 |
| k618  | -104.4571 | 54.18616  | -1.93 | 0.056 | -211.5538            | 2.639668  |
| wa    | -4.784917 | 9.690502  | -0.49 | 0.622 | -23.9378             | 14.36797  |
| we    | 9.353195  | 31.23793  | 0.30  | 0.765 | -52.38731            | 71.0937   |
| _cons | 1629.817  | 615.1301  | 2.65  | 0.009 | 414.0371             | 2845.597  |

Now we use `truncreg` to perform truncated regression with truncation from below zero.

```
. truncreg whrs kl6 k618 wa we, ll(0)
(note: 100 obs. truncated)
```

Fitting full model:

```
Iteration 0: log likelihood = -1205.6992
Iteration 1: log likelihood = -1200.9873
Iteration 2: log likelihood = -1200.9159
Iteration 3: log likelihood = -1200.9157
Iteration 4: log likelihood = -1200.9157
```

## Truncated regression

```
Limit: lower = 0
 upper = +inf
Log likelihood = -1200.9157
```

```
Number of obs = 150
Wald chi2(4) = 10.05
Prob > chi2 = 0.0395
```

| whrs   | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|--------|-----------|-----------|-------|-------|----------------------|-----------|
| kl6    | -803.0042 | 321.3614  | -2.50 | 0.012 | -1432.861            | -173.1474 |
| k618   | -172.875  | 88.72898  | -1.95 | 0.051 | -346.7806            | 1.030579  |
| wa     | -8.821123 | 14.36848  | -0.61 | 0.539 | -36.98283            | 19.34059  |
| we     | 16.52873  | 46.50375  | 0.36  | 0.722 | -74.61695            | 107.6744  |
| _cons  | 1586.26   | 912.355   | 1.74  | 0.082 | -201.9233            | 3374.442  |
| /sigma | 983.7262  | 94.44303  | 10.42 | 0.000 | 798.6213             | 1168.831  |

If we assume that our data were censored, the tobit model is

```
. tobit whrs k16 k618 wa we, ll(0)
Tobit regression
Log likelihood = -1367.0903
Number of obs = 250
LR chi2(4) = 23.03
Prob > chi2 = 0.0001
Pseudo R2 = 0.0084
```

| whrs   | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|--------|-----------|-----------|-------|-------|----------------------|-----------|
| k16    | -827.7657 | 214.7407  | -3.85 | 0.000 | -1250.731            | -404.8008 |
| k618   | -140.0192 | 74.22303  | -1.89 | 0.060 | -286.2129            | 6.174547  |
| wa     | -24.97919 | 13.25639  | -1.88 | 0.061 | -51.08969            | 1.131317  |
| we     | 103.6896  | 41.82393  | 2.48  | 0.014 | 21.31093             | 186.0683  |
| _cons  | 589.0001  | 841.5467  | 0.70  | 0.485 | -1068.556            | 2246.556  |
| /sigma | 1309.909  | 82.73335  |       |       | 1146.953             | 1472.865  |

Obs. summary: 100 left-censored observations at whrs<=0  
150 uncensored observations  
0 right-censored observations

□ Technical note

Whether truncated regression is more appropriate than the ordinary least-squares estimation depends on the purpose of that estimation. If we are interested in the mean of wife’s working hours conditional on the subsample of market laborers, least-squares estimation is appropriate. However if we are interested in the mean of wife’s working hours regardless of market or nonmarket labor status, least-squares estimates could be seriously misleading.

Truncation and censoring are different concepts. A sample has been censored if no observations have been systematically excluded but some of the information contained in them has been suppressed. In a truncated distribution, only the part of the distribution above (or below, or between) the truncation points is relevant to our computations. We need to scale it up by the probability that an observation falls in the range that interests us to make the distribution integrate to one. The censored distribution used by tobit, however, is a mixture of discrete and continuous distributions. Instead of rescaling over the observable range, we simply assign the full probability from the censored regions to the censoring points. The truncated regression model is sometimes less well behaved than the tobit model. [Davidson and MacKinnon \(1993\)](#) provide an example where truncation results in more inconsistency than censoring.

(Continued on next page)

## Saved results

**truncreg** saves the following in **e()**:

### Scalars

|                      |                                                |
|----------------------|------------------------------------------------|
| <b>e(N)</b>          | number of observations                         |
| <b>e(N_bf)</b>       | number of obs. before truncation               |
| <b>e(chi2)</b>       | model $\chi^2$                                 |
| <b>e(k_eq)</b>       | number of equations in <b>e(b)</b>             |
| <b>e(k_eq_model)</b> | number of equations in model Wald test         |
| <b>e(k_aux)</b>      | number of auxiliary parameters                 |
| <b>e(k_autoCns)</b>  | number of base, empty, and omitted constraints |
| <b>e(df_m)</b>       | model degrees of freedom                       |
| <b>e(ll)</b>         | log likelihood                                 |
| <b>e(ll_0)</b>       | log likelihood, constant-only model            |
| <b>e(N_clust)</b>    | number of clusters                             |
| <b>e(sigma)</b>      | estimate of sigma                              |
| <b>e(p)</b>          | significance                                   |
| <b>e(rank)</b>       | rank of <b>e(V)</b>                            |
| <b>e(ic)</b>         | number of iterations                           |
| <b>e(rc)</b>         | return code                                    |
| <b>e(converged)</b>  | 1 if converged, 0 otherwise                    |

### Macros

|                           |                                                                          |
|---------------------------|--------------------------------------------------------------------------|
| <b>e(cmd)</b>             | <b>truncreg</b>                                                          |
| <b>e(cmdline)</b>         | command as typed                                                         |
| <b>e(llopt)</b>           | contents of <b>ll()</b> , if specified                                   |
| <b>e(ulopt)</b>           | contents of <b>ul()</b> , if specified                                   |
| <b>e(depvar)</b>          | name of dependent variable                                               |
| <b>e(wtype)</b>           | weight type                                                              |
| <b>e(wexp)</b>            | weight expression                                                        |
| <b>e(title)</b>           | title in estimation output                                               |
| <b>e(clustvar)</b>        | name of cluster variable                                                 |
| <b>e(offset1)</b>         | offset                                                                   |
| <b>e(chi2type)</b>        | Wald or LR; type of model $\chi^2$ test                                  |
| <b>e(vce)</b>             | <i>vcetype</i> specified in <b>vce()</b>                                 |
| <b>e(vcetype)</b>         | title used to label Std. Err.                                            |
| <b>e(opt)</b>             | type of optimization                                                     |
| <b>e(which)</b>           | max or min; whether optimizer is to perform maximization or minimization |
| <b>e(ml_method)</b>       | type of <b>ml</b> method                                                 |
| <b>e(user)</b>            | name of likelihood-evaluator program                                     |
| <b>e(technique)</b>       | maximization technique                                                   |
| <b>e(singularHmethod)</b> | m-marquardt or hybrid; method used when Hessian is singular              |
| <b>e(crittype)</b>        | optimization criterion                                                   |
| <b>e(properties)</b>      | <b>b V</b>                                                               |
| <b>e(predict)</b>         | program used to implement <b>predict</b>                                 |
| <b>e(asbalanced)</b>      | factor variables <b>fvset</b> as <b>asbalanced</b>                       |
| <b>e(asobserved)</b>      | factor variables <b>fvset</b> as <b>asobserved</b>                       |

## Matrices

|                              |                                                   |
|------------------------------|---------------------------------------------------|
| <code>e(b)</code>            | coefficient vector                                |
| <code>e(Cns)</code>          | constraints matrix                                |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)               |
| <code>e(gradient)</code>     | gradient vector                                   |
| <code>e(V)</code>            | variance–covariance matrix of the estimators      |
| <code>e(V_modelbased)</code> | model-based variance                              |
| <code>e(ml_h)</code>         | derivative tolerance, $(\text{abs}(b)+1e-3)*1e-3$ |
| <code>e(ml_scale)</code>     | derivative scale factor                           |
| <code>e(means)</code>        | means of independent variables                    |
| <code>e(dummy)</code>        | indicator for dummy variables                     |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`truncreg` is implemented as an ado-file. [Greene \(2008, 863–869\)](#) and [Davidson and MacKinnon \(1993, 534–537\)](#) provide introductions to the truncated regression model.

Let  $\mathbf{y} = \mathbf{X}\beta + \epsilon$  be the model.  $\mathbf{y}$  represents continuous outcomes either observed or not observed. Our model assumes that  $\epsilon \sim N(0, \sigma^2 \mathbf{I})$ .

Let  $a$  be the lower limit and  $b$  be the upper limit. The log likelihood is

$$\ln L = -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{j=1}^n (y_j - \mathbf{x}_j\beta)^2 - \sum_{j=1}^n \log \left\{ \Phi\left(\frac{b - \mathbf{x}_j\beta}{\sigma}\right) - \Phi\left(\frac{a - \mathbf{x}_j\beta}{\sigma}\right) \right\}$$

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [\[P\] `\_robust`](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

`truncreg` also supports estimation with survey data. For details on VCEs with survey data, see [\[SVY\] variance estimation](#).

## References

- Berndt, E. R. 1996. *The Practice of Econometrics: Classic and Contemporary*. New York: Addison–Wesley.
- Cong, R. 1999. [sg122: Truncated regression](#). *Stata Technical Bulletin* 52: 47–52. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 248–255. College Station, TX: Stata Press.
- Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.
- Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.

## Also see

- [\[R\] `truncreg postestimation`](#) — Postestimation tools for `truncreg`
- [\[R\] `regress`](#) — Linear regression
- [\[R\] `tobit`](#) — Tobit regression
- [\[U\] 20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `truncreg`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic nooffset]
```

```
predict [type] { stub*| newvarreg newvarlnsigma } [if] [in] , scores
```

| options                 | description                                 |
|-------------------------|---------------------------------------------|
| Main                    |                                             |
| <code>xb</code>         | linear prediction; the default              |
| <code>stdp</code>       | standard error of the prediction            |
| <code>stdf</code>       | standard error of the forecast              |
| <code>pr(a,b)</code>    | $\Pr(a < y_j < b)$                          |
| <code>e(a,b)</code>     | $E(y_j   a < y_j < b)$                      |
| <code>ystar(a,b)</code> | $E(y_j^*), y_j^* = \max\{a, \min(y_j, b)\}$ |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

`stdf` is not allowed with `svy` estimation results.

where  $a$  and  $b$  may be numbers or variables;  $a$  missing ( $a \geq .$ ) means  $-\infty$ , and  $b$  missing ( $b \geq .$ ) means  $+\infty$ ; see [U] [12.2.1 Missing values](#).

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

### Main

**xb**, the default, calculates the linear prediction.

**stdp** calculates the standard error of the prediction, which can be thought of as the standard error of the predicted expected value or mean for the observation's covariate pattern. The standard error of the prediction is also referred to as the standard error of the fitted value.

**stdf** calculates the standard error of the forecast, which is the standard error of the point prediction for 1 observation. It is commonly referred to as the standard error of the future or forecast value. By construction, the standard errors produced by **stdf** are always larger than those produced by **stdp**; see [Methods and formulas](#) in [R] [regress](#).

**pr( $a, b$ )** calculates  $\Pr(a < \mathbf{x}_j \mathbf{b} + u_j < b)$ , the probability that  $y_j | \mathbf{x}_j$  would be observed in the interval  $(a, b)$ .

$a$  and  $b$  may be specified as numbers or variable names;  $lb$  and  $ub$  are variable names;

**pr(20,30)** calculates  $\Pr(20 < \mathbf{x}_j \mathbf{b} + u_j < 30)$ ;

**pr( $lb, ub$ )** calculates  $\Pr(lb < \mathbf{x}_j \mathbf{b} + u_j < ub)$ ; and

**pr(20,  $ub$ )** calculates  $\Pr(20 < \mathbf{x}_j \mathbf{b} + u_j < ub)$ .

$a$  missing ( $a \geq .$ ) means  $-\infty$ ; **pr(. ,30)** calculates  $\Pr(-\infty < \mathbf{x}_j \mathbf{b} + u_j < 30)$ ;

**pr( $lb, 30$ )** calculates  $\Pr(-\infty < \mathbf{x}_j \mathbf{b} + u_j < 30)$  in observations for which  $lb \geq .$  and calculates  $\Pr(lb < \mathbf{x}_j \mathbf{b} + u_j < 30)$  elsewhere.

$b$  missing ( $b \geq .$ ) means  $+\infty$ ; **pr(20, .)** calculates  $\Pr(+\infty > \mathbf{x}_j \mathbf{b} + u_j > 20)$ ;

**pr(20,  $ub$ )** calculates  $\Pr(+\infty > \mathbf{x}_j \mathbf{b} + u_j > 20)$  in observations for which  $ub \geq .$  and calculates  $\Pr(20 < \mathbf{x}_j \mathbf{b} + u_j < ub)$  elsewhere.

**e( $a, b$ )** calculates  $E(\mathbf{x}_j \mathbf{b} + u_j \mid a < \mathbf{x}_j \mathbf{b} + u_j < b)$ , the expected value of  $y_j | \mathbf{x}_j$  conditional on  $y_j | \mathbf{x}_j$  being in the interval  $(a, b)$ , meaning that  $y_j | \mathbf{x}_j$  is censored.

$a$  and  $b$  are specified as they are for **pr()**.

**ystar( $a, b$ )** calculates  $E(y_j^*)$ , where  $y_j^* = a$  if  $\mathbf{x}_j \mathbf{b} + u_j \leq a$ ,  $y_j^* = b$  if  $\mathbf{x}_j \mathbf{b} + u_j \geq b$ , and  $y_j^* = \mathbf{x}_j \mathbf{b} + u_j$  otherwise, meaning that  $y_j^*$  is truncated.  $a$  and  $b$  are specified as they are for **pr()**.

**nooffset** is relevant only if you specified **offset(varname)**. It modifies the calculations made by **predict** so that they ignore the offset variable; the linear prediction is treated as  $\mathbf{x}_j \mathbf{b}$  rather than as  $\mathbf{x}_j \mathbf{b} + \text{offset}_j$ .

**scores** calculates equation-level score variables.

The first new variable will contain  $\partial \ln L / \partial (\mathbf{x}_j \boldsymbol{\beta})$ .

The second new variable will contain  $\partial \ln L / \partial \sigma$ .

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [truncreg](#) — Truncated regression

[U] [20 Estimation and postestimation commands](#)



Syntax

One-sample mean-comparison test

```
ttest varname == # [if] [in] [, level(#)]
```

Two-sample mean-comparison test (unpaired)

```
ttest varname1 == varname2 [if] [in], unpaired [unequal welch level(#)]
```

Two-sample mean-comparison test (paired)

```
ttest varname1 == varname2 [if] [in] [, level(#)]
```

Two-group mean-comparison test

```
ttest varname [if] [in], by(groupvar) [options1]
```

Immediate form of one-sample mean-comparison test

```
ttesti #obs #mean #sd #val [, level(#)]
```

Immediate form of two-sample mean-comparison test

```
ttesti #obs1 #mean1 #sd1 #obs2 #mean2 #sd2 [, options2]
```

| <i>options</i> <sub>1</sub> | description                                |
|-----------------------------|--------------------------------------------|
| Main                        |                                            |
| *by( <i>groupvar</i> )      | variable defining the groups               |
| <u>unequal</u>              | unpaired data have unequal variances       |
| <u>welch</u>                | use Welch's approximation                  |
| <u>level</u> (#)            | set confidence level; default is level(95) |

\*by(*groupvar*) is required.

| <i>options</i> <sub>2</sub> | description                                |
|-----------------------------|--------------------------------------------|
| Main                        |                                            |
| <u>unequal</u>              | unpaired data have unequal variances       |
| <u>welch</u>                | use Welch's approximation                  |
| <u>level</u> (#)            | set confidence level; default is level(95) |

by is allowed with test; see [D] by.

## Menu

### one-sample

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > One-sample mean-comparison test

### two-sample, unpaired

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > Two-sample mean-comparison test

### two-sample, paired

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > Mean-comparison test, paired data

### two-group

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > Two-group mean-comparison test

### immediate command: one-sample

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > One-sample mean-comparison calculator

### immediate command: two-sample

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > Two-sample mean-comparison calculator

## Description

`ttest` performs  $t$  tests on the equality of means. In the first form, `ttest` tests that *varname* has a mean of *#*. In the second form, `ttest` tests that *varname*<sub>1</sub> and *varname*<sub>2</sub> have the same mean, assuming *unpaired* data. In the third form, `ttest` tests that *varname*<sub>1</sub> and *varname*<sub>2</sub> have the same mean, assuming *paired* data. In the fourth form, `ttest` tests that *varname* has the same mean within the two groups defined by *groupvar*.

`ttesti` is the immediate form of `ttest`; see [U] 19 [Immediate commands](#).

For the equivalent of a two-sample  $t$  test with sampling weights (*pweights*), use the `svy: mean` command with the `over()` option, and then use `lincom`; see [R] [mean](#) and [SVY] [svy postestimation](#).

## Options

### Main

`by(groupvar)` specifies the *groupvar* that defines the two groups that `ttest` will use to test the hypothesis that their means are equal. Specifying `by(groupvar)` implies an unpaired (two sample)  $t$  test. Do not confuse the `by()` option with the `by` prefix; you can specify both.

`unpaired` specifies that the data be treated as unpaired. The `unpaired` option is used when the two sets of values to be compared are in different variables.

`unequal` specifies that the unpaired data not be assumed to have equal variances.

`welch` specifies that the approximate degrees of freedom for the test be obtained from Welch's formula (1947) rather than from Satterthwaite's approximation formula (1946), which is the default when `unequal` is specified. Specifying `welch` implies `unequal`.

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see [U] 20.7 [Specifying the width of confidence intervals](#).

## Remarks

### ► Example 1: One-sample mean-comparison test

In the first form, `ttest` tests whether the mean of the sample is equal to a known constant under the assumption of unknown variance. Assume that we have a sample of 74 automobiles. We know each automobile's average mileage rating and wish to test whether the overall average for the sample is 20 miles per gallon.

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)
. ttest mpg==20
One-sample t test
```

| Variable           | Obs | Mean                   | Std. Err. | Std. Dev.               | [95% Conf. Interval] |          |
|--------------------|-----|------------------------|-----------|-------------------------|----------------------|----------|
| mpg                | 74  | 21.2973                | .6725511  | 5.785503                | 19.9569              | 22.63769 |
| mean = mean(mpg)   |     |                        |           | t = 1.9289              |                      |          |
| Ho: mean = 20      |     |                        |           | degrees of freedom = 73 |                      |          |
| Ha: mean < 20      |     | Ha: mean != 20         |           | Ha: mean > 20           |                      |          |
| Pr(T < t) = 0.9712 |     | Pr( T  >  t ) = 0.0576 |           | Pr(T > t) = 0.0288      |                      |          |

The test indicates that the underlying mean is not 20 with a significance level of 5.8%.

◀

### ► Example 2: Two-sample mean-comparison test

We are testing the effectiveness of a new fuel additive. We run an experiment with 12 cars. We run the cars without and with the fuel treatment. The results of the experiment are as follows:

| Without treatment | With treatment | Without treatment | With treatment |
|-------------------|----------------|-------------------|----------------|
| 20                | 24             | 18                | 17             |
| 23                | 25             | 24                | 28             |
| 21                | 21             | 20                | 24             |
| 25                | 22             | 24                | 27             |
| 18                | 23             | 23                | 21             |
| 17                | 18             | 19                | 23             |

By creating two variables called `mpg1` and `mpg2` representing mileage without and with the treatment, respectively, we can test the equality of means by typing

```
. use http://www.stata-press.com/data/r11/fuel
. ttest mpg1==mpg2
Paired t test
```

| Variable                       | Obs | Mean                   | Std. Err. | Std. Dev.               | [95% Conf. Interval] |           |
|--------------------------------|-----|------------------------|-----------|-------------------------|----------------------|-----------|
| mpg1                           | 12  | 21                     | .7881701  | 2.730301                | 19.26525             | 22.73475  |
| mpg2                           | 12  | 22.75                  | .9384465  | 3.250874                | 20.68449             | 24.81551  |
| diff                           | 12  | -1.75                  | .7797144  | 2.70101                 | -3.46614             | -.0338602 |
| mean(diff) = mean(mpg1 - mpg2) |     |                        |           | t = -2.2444             |                      |           |
| Ho: mean(diff) = 0             |     |                        |           | degrees of freedom = 11 |                      |           |
| Ha: mean(diff) < 0             |     | Ha: mean(diff) != 0    |           | Ha: mean(diff) > 0      |                      |           |
| Pr(T < t) = 0.0232             |     | Pr( T  >  t ) = 0.0463 |           | Pr(T > t) = 0.9768      |                      |           |

We find that the means are statistically different from each other at any level greater than 4.6%.

➤ Example 3: Group mean-comparison test

Let's pretend that the preceding data were collected by running 24 cars: 12 cars with the additive and 12 without. Although we might be tempted to enter the data in the same way, we should resist (see the technical note below). Instead, we enter the data as 24 observations on mpg with an additional variable, `treated`, taking on 1 if the car received the fuel treatment and 0 otherwise:

```
. use http://www.stata-press.com/data/r11/fuel3
. ttest mpg, by(treated)
Two-sample t test with equal variances
```

| Group    | Obs | Mean   | Std. Err. | Std. Dev. | [95% Conf. Interval] |          |
|----------|-----|--------|-----------|-----------|----------------------|----------|
| 0        | 12  | 21     | .7881701  | 2.730301  | 19.26525             | 22.73475 |
|          | 12  | 22.75  | .9384465  | 3.250874  | 20.68449             | 24.81551 |
| combined | 24  | 21.875 | .6264476  | 3.068954  | 20.57909             | 23.17091 |
| diff     |     | -1.75  | 1.225518  |           | -4.291568            | .7915684 |

diff = mean(0) - mean(1)

Ho: diff = 0

Ha: diff < 0

Pr(T < t) = 0.0837

t = -1.4280

degrees of freedom = 22

Ha: diff != 0

Pr(|T| > |t|) = 0.1673

Ha: diff > 0

Pr(T > t) = 0.9163

This time we do not find a statistically significant difference.

If we were not willing to assume that the variances were equal and wanted to use Welch's formula, we could type

```
. ttest mpg, by(treated) welch
Two-sample t test with unequal variances
```

| Group    | Obs | Mean   | Std. Err. | Std. Dev. | [95% Conf. Interval] |          |
|----------|-----|--------|-----------|-----------|----------------------|----------|
| 0        | 12  | 21     | .7881701  | 2.730301  | 19.26525             | 22.73475 |
|          | 12  | 22.75  | .9384465  | 3.250874  | 20.68449             | 24.81551 |
| combined | 24  | 21.875 | .6264476  | 3.068954  | 20.57909             | 23.17091 |
| diff     |     | -1.75  | 1.225518  |           | -4.28369             | .7836902 |

diff = mean(0) - mean(1)

Ho: diff = 0

Ha: diff < 0

Pr(T < t) = 0.0833

t = -1.4280

Welch's degrees of freedom = 23.2465

Ha: diff != 0

Pr(|T| > |t|) = 0.1666

Ha: diff > 0

Pr(T > t) = 0.9167

□ Technical note

In two-group randomized designs, subjects will sometimes refuse the assigned treatment but still be measured for an outcome. In this case, take care to specify the group properly. You might be tempted to let `varname` contain missing where the subject refused and thus let `ttest` drop such observations from the analysis. [Zelen \(1979\)](#) argues that it would be better to specify that the subject belongs to the group in which he or she was randomized, even though such inclusion will dilute the measured effect.

## □ Technical note

There is a second, inferior way to organize the data in the preceding example. Remember, we ran a test on 24 cars, 12 without the additive and 12 with. Nevertheless, we could have entered the data in the same way as we did when we had 12 cars, each run without and with the additive, by creating two variables—`mpg1` and `mpg2`.

This method is inferior because it suggests a connection that is not there. For the 12-car experiment, there was most certainly a connection—it was the same car. In the 24-car experiment, however, it is arbitrary which `mpg` results appear next to which. Nevertheless, if our data are organized like this, `ttest` can accommodate us.

```
. use http://www.stata-press.com/data/r11/fuel
```

```
. ttest mpg1==mpg2, unpaired
```

Two-sample t test with equal variances

| Variable | Obs | Mean   | Std. Err. | Std. Dev. | [95% Conf. Interval] |          |
|----------|-----|--------|-----------|-----------|----------------------|----------|
| mpg1     | 12  | 21     | .7881701  | 2.730301  | 19.26525             | 22.73475 |
| mpg2     | 12  | 22.75  | .9384465  | 3.250874  | 20.68449             | 24.81551 |
| combined | 24  | 21.875 | .6264476  | 3.068954  | 20.57909             | 23.17091 |
| diff     |     | -1.75  | 1.225518  |           | -4.291568            | .7915684 |

```
diff = mean(mpg1) - mean(mpg2) t = -1.4280
Ho: diff = 0 degrees of freedom = 22
```

```
Ha: diff < 0 Ha: diff != 0 Ha: diff > 0
Pr(T < t) = 0.0837 Pr(|T| > |t|) = 0.1673 Pr(T > t) = 0.9163
```

□

## ▷ Example 4

`ttest` can be used to test the equality of a pair of means; see [R] [oneway](#) for testing the equality of more than two means.

Suppose that we have data on the 50 states. The dataset contains the median age of the population (`medage`) and the region of the country (`region`) for each state. Region 1 refers to the Northeast, region 2 to the North Central, region 3 to the South, and region 4 to the West. Using `oneway`, we can test the equality of all four means.

```
. use http://www.stata-press.com/data/r11/census
(1980 Census data by state)
```

```
. oneway medage region
```

| Source         | Analysis of Variance |    |            | F    | Prob > F |
|----------------|----------------------|----|------------|------|----------|
|                | SS                   | df | MS         |      |          |
| Between groups | 46.3961903           | 3  | 15.4653968 | 7.56 | 0.0003   |
| Within groups  | 94.1237947           | 46 | 2.04616945 |      |          |
| Total          | 140.519985           | 49 | 2.8677548  |      |          |

```
Bartlett's test for equal variances: chi2(3) = 10.5757 Prob>chi2 = 0.014
```

We find that the means are different, but we are interested only in testing whether the means for the Northeast (`region==1`) and West (`region==4`) are different. We could use `oneway`,

```
. oneway medage region if region==1 | region==4
```

| Source         | Analysis of Variance |    |            | F     | Prob > F |
|----------------|----------------------|----|------------|-------|----------|
|                | SS                   | df | MS         |       |          |
| Between groups | 46.241247            | 1  | 46.241247  | 20.02 | 0.0002   |
| Within groups  | 46.1969169           | 20 | 2.30984584 |       |          |
| Total          | 92.4381638           | 21 | 4.40181733 |       |          |

Bartlett's test for equal variances: chi2(1) = 2.4679 Prob>chi2 = 0.116

or we could use ttest:

```
. ttest medage if region==1 | region==4, by(region)
```

Two-sample t test with equal variances

| Group    | Obs | Mean     | Std. Err. | Std. Dev. | [95% Conf. Interval] |          |
|----------|-----|----------|-----------|-----------|----------------------|----------|
| NE       | 9   | 31.23333 | .3411581  | 1.023474  | 30.44662             | 32.02005 |
| West     | 13  | 28.28462 | .4923577  | 1.775221  | 27.21186             | 29.35737 |
| combined | 22  | 29.49091 | .4473059  | 2.098051  | 28.56069             | 30.42113 |
| diff     |     | 2.948718 | .6590372  |           | 1.57399              | 4.323445 |

diff = mean(NE) - mean(West) t = 4.4743  
Ho: diff = 0 degrees of freedom = 20  
Ha: diff < 0 Ha: diff != 0 Ha: diff > 0  
Pr(T < t) = 0.9999 Pr(|T| > |t|) = 0.0002 Pr(T > t) = 0.0001

The significance levels of both tests are the same.



Immediate form

Example 5

ttesti is like ttest, except that we specify summary statistics rather than variables as arguments. For instance, we are reading an article that reports the mean number of sunspots per month as 62.6 with a standard deviation of 15.8. There are 24 months of data. We wish to test whether the mean is 75:

```
. ttesti 24 62.6 15.8 75
```

One-sample t test

|   | Obs | Mean | Std. Err. | Std. Dev. | [95% Conf. Interval] |          |
|---|-----|------|-----------|-----------|----------------------|----------|
| x | 24  | 62.6 | 3.225161  | 15.8      | 55.92825             | 69.27175 |

mean = mean(x) t = -3.8448  
Ho: mean = 75 degrees of freedom = 23  
Ha: mean < 75 Ha: mean != 75 Ha: mean > 75  
Pr(T < t) = 0.0004 Pr(|T| > |t|) = 0.0008 Pr(T > t) = 0.9996



► Example 6

There is no immediate form of `ttest` with paired data because the test is also a function of the covariance, a number unlikely to be reported in any published source. For nonpaired data, however, we might type

```
. ttesti 20 20 5 32 15 4
Two-sample t test with equal variances
```

|          | Obs | Mean     | Std. Err. | Std. Dev. | [95% Conf. Interval] |          |
|----------|-----|----------|-----------|-----------|----------------------|----------|
| x        | 20  | 20       | 1.118034  | 5         | 17.65993             | 22.34007 |
| y        | 32  | 15       | .7071068  | 4         | 13.55785             | 16.44215 |
| combined | 52  | 16.92308 | .6943785  | 5.007235  | 15.52905             | 18.3171  |
| diff     |     | 5        | 1.256135  |           | 2.476979             | 7.523021 |

```
diff = mean(x) - mean(y) t = 3.9805
Ho: diff = 0 degrees of freedom = 50
Ha: diff < 0 Ha: diff != 0 Ha: diff > 0
Pr(T < t) = 0.9999 Pr(|T| > |t|) = 0.0002 Pr(T > t) = 0.0001
```

If we had typed `ttesti 20 20 5 32 15 4, unequal`, the test would have assumed unequal variances. ◀

Saved results

`ttest` and `ttesti` save the following in `r()`:

|         |                            |         |                                        |
|---------|----------------------------|---------|----------------------------------------|
| Scalars |                            |         |                                        |
| r(N_1)  | sample size $n_1$          | r(t)    | $t$ statistic                          |
| r(N_2)  | sample size $n_2$          | r(sd_1) | standard deviation for first variable  |
| r(p_l)  | lower one-sided $p$ -value | r(sd_2) | standard deviation for second variable |
| r(p_u)  | upper one-sided $p$ -value | r(mu_1) | $\bar{x}_1$ mean for population 1      |
| r(p)    | two-sided $p$ -value       | r(mu_2) | $\bar{x}_2$ mean for population 2      |
| r(se)   | estimate of standard error | r(df_t) | degrees of freedom                     |

Methods and formulas

`ttest` and `ttesti` are implemented as `ado`-files.

See, for instance, [Hoel \(1984, 140–161\)](#) or [Dixon and Massey Jr. \(1983, 121–130\)](#) for an introduction and explanation of the calculation of these tests. [Acock \(2008, 143–153\)](#) and [Hamilton \(2009, 157–162\)](#) describe  $t$  tests using applications in Stata.

The test for  $\mu = \mu_0$  for unknown  $\sigma$  is given by

$$t = \frac{(\bar{x} - \mu_0)\sqrt{n}}{s}$$

The statistic is distributed as Student’s  $t$  with  $n - 1$  degrees of freedom ([Gosset \[Student, pseud.\] 1908](#)).

The test for  $\mu_x = \mu_y$  when  $\sigma_x$  and  $\sigma_y$  are unknown but  $\sigma_x = \sigma_y$  is given by

$$t = \frac{\bar{x} - \bar{y}}{\left\{ \frac{(n_x - 1)s_x^2 + (n_y - 1)s_y^2}{n_x + n_y - 2} \right\}^{1/2} \left( \frac{1}{n_x} + \frac{1}{n_y} \right)^{1/2}}$$

The result is distributed as Student's  $t$  with  $n_x + n_y - 2$  degrees of freedom.

You could perform `ttest` (without the `unequal` option) in a regression setting given that regression assumes a homoskedastic error model. To compare with the `ttest` command, denote the underlying observations on  $x$  and  $y$  by  $x_j$ ,  $j = 1, \dots, n_x$ , and  $y_j$ ,  $j = 1, \dots, n_y$ . In a regression framework, typing `ttest` without the `unequal` option is equivalent to

1. creating a new variable  $z_j$  that represents the stacked observations on  $x$  and  $y$  (so that  $z_j = x_j$  for  $j = 1, \dots, n_x$  and  $z_{n_x+j} = y_j$  for  $j = 1, \dots, n_y$ )
2. and then estimating the equation  $z_j = \beta_0 + \beta_1 d_j + \epsilon_j$ , where  $d_j = 0$  for  $j = 1, \dots, n_x$  and  $d_j = 1$  for  $j = n_x + 1, \dots, n_x + n_y$  (i.e.,  $d_j = 0$  when the  $z$  observations represent  $x$ , and  $d_j = 1$  when the  $z$  observations represent  $y$ ).

The estimated value of  $\beta_1$ ,  $b_1$ , will equal  $\bar{y} - \bar{x}$ , and the reported  $t$  statistic will be the same  $t$  statistic as given by the formula above.

The test for  $\mu_x = \mu_y$  when  $\sigma_x$  and  $\sigma_y$  are unknown and  $\sigma_x \neq \sigma_y$  is given by

$$t = \frac{\bar{x} - \bar{y}}{\left( s_x^2/n_x + s_y^2/n_y \right)^{1/2}}$$

The result is distributed as Student's  $t$  with  $\nu$  degrees of freedom, where  $\nu$  is given by (with Satterthwaite's [1946] formula)

$$\frac{\left( s_x^2/n_x + s_y^2/n_y \right)^2}{\frac{\left( s_x^2/n_x \right)^2}{n_x - 1} + \frac{\left( s_y^2/n_y \right)^2}{n_y - 1}}$$

With Welch's formula (1947), the number of degrees of freedom is given by

$$-2 + \frac{\left( s_x^2/n_x + s_y^2/n_y \right)^2}{\frac{\left( s_x^2/n_x \right)^2}{n_x + 1} + \frac{\left( s_y^2/n_y \right)^2}{n_y + 1}}$$

The test for  $\mu_x = \mu_y$  for matched observations (also known as paired observations, correlated pairs, or permanent components) is given by

$$t = \frac{\bar{d}\sqrt{n}}{s_d}$$

where  $\bar{d}$  represents the mean of  $x_i - y_i$  and  $s_d$  represents the standard deviation. The test statistic  $t$  is distributed as Student's  $t$  with  $n - 1$  degrees of freedom.



You can also use `ttest` without the `unpaired` option in a regression setting because a paired comparison includes the assumption of constant variance. The `ttest` with an unequal variance assumption does not lend itself to an easy representation in regression settings and is not discussed here.  $(x_j - y_j) = \beta_0 + \epsilon_j$ .

William Sealy Gosset (1876–1937) was born in Canterbury, England. He studied chemistry and mathematics at Oxford and worked as a chemist with the brewers Guinness in Dublin. Gosset became interested in statistical problems, which he discussed with Karl Pearson and later with Fisher and Neyman, and published several important papers under the pseudonym “Student”, including that on the test that usually bears his name.

## References

- Acocck, A. C. 2008. *A Gentle Introduction to Stata*. 2nd ed. College Station, TX: Stata Press.
- Boland, P. J. 2000. William Sealy Gosset—alias ‘Student’ 1876–1937. In *Creators of Mathematics: The Irish Connection*, ed. K. Houston, 105–112. Dublin: University College Dublin Press.
- Dixon, W. J., and F. J. Massey Jr. 1983. *Introduction to Statistical Analysis*. 4th ed. New York: McGraw–Hill.
- Gleason, J. R. 1999. [sg101: Pairwise comparisons of means, including the Tukey wsd method](#). *Stata Technical Bulletin* 47: 31–37. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 225–233. College Station, TX: Stata Press.
- Gosset, W. S. 1943. “Student’s” *Collected Papers*. London: Biometrika Office, University College.
- Gosset [Student, pseud.], W. S. 1908. The probable error of a mean. *Biometrika* 6: 1–25.
- Hamilton, L. C. 2009. *Statistics with Stata (Updated for Version 10)*. Belmont, CA: Brooks/Cole.
- Hoel, P. G. 1984. *Introduction to Mathematical Statistics*. 5th ed. New York: Wiley.
- Pearson, E. S., R. L. Plackett, and G. A. Barnard. 1990. ‘Student’: A Statistical Biography of William Sealy Gosset. Oxford: Oxford University Press.
- Preece, D. A. 1982. t is for trouble (and textbooks): A critique of some examples of the paired-samples t-test. *Statistician* 31: 169–195.
- Satterthwaite, F. E. 1946. An approximate distribution of estimates of variance components. *Biometrics Bulletin* 2: 110–114.
- Senn, S. J., and W. Richardson. 1994. The first t-test. *Statistics in Medicine* 13: 785–803.
- Welch, B. L. 1947. The generalization of ‘student’s’ problem when several different population variances are involved. *Biometrika* 34: 28–35.
- Zelen, M. 1979. A new design for randomized clinical trials. *New England Journal of Medicine* 300: 1242–1245.

## Also see

- [R] [bitest](#) — Binomial probability test
- [R] [ci](#) — Confidence intervals for means, proportions, and counts
- [R] [mean](#) — Estimate means
- [R] [oneway](#) — One-way analysis of variance
- [R] [prtest](#) — One- and two-sample tests of proportions
- [R] [sdtest](#) — Variance-comparison tests
- [MV] [hotelling](#) — Hotelling’s T-squared generalized means test

# Title

**update** — Update Stata

## Syntax

*Report on update level of currently installed Stata*

```
update
```

*Set update source*

```
update from location
```

*Compare update level of currently installed Stata with that of source*

```
update query [, from(location)]
```

*Update ado-files from source if necessary*

```
update ado [, from(location) into(dirname)]
```

*Update executable from source if necessary*

```
update executable [, from(location) into(dirname) force]
```

*Update utilities from source if necessary*

```
update utilities [, from(location) into(dirname) force]
```

*Perform update ado, update executable, and update utilities if necessary*

```
update all [, from(location)]
```

*Complete installation of newly updated executable*

```
update swap [, clear]
```

*Set automatic updates (Mac and Windows only)*

```
set update_query { on | off }
```

```
set update_interval #
```

```
set update_prompt { on | off }
```

## Menu

Help > Official Updates

## Description

The `update` command reports on the current update level and installs official updates to Stata. Official updates are updates to Stata as it was originally shipped from StataCorp, not the additions to Stata published in, for instance, the *Stata Journal* (SJ) or *Stata Technical Bulletin* (STB). Those additions are installed using the `net` command and updated using the `adoupdate` command; see [\[R\] net](#) and [\[R\] adoupdate](#).

`update` without arguments reports on the update level of the currently installed Stata.

`update from` sets an update source, where *location* is a directory name or URL. If you are on the Internet, type `update from http://www.stata.com`.

`update query` compares the update level of the currently installed Stata with that available from the update source and displays a report.

`update ado` compares the update level of the official ado-files of the currently installed Stata with those available from the update source. If the currently installed ado-files need updating, `update ado` copies and installs those files from the update source that are necessary to bring the ado-files up to date. `update ado` is seldom used by itself; it is called automatically by `update all`.

`update executable` compares the update level of the currently installed Stata executable with that available from the update source. If the currently installed Stata needs updating, `update executable` copies the new executable from the update source, but the last step of the installation—erasing the old executable and renaming the new executable—must be performed by the `update swap` command. `update executable` displays instructions on how to do this. `update executable` is seldom used by itself; it is called automatically by `update all`.

`update utilities` compares the update level of the currently installed Stata utilities, including Stata's PDF documentation, with that available from the update source. If the currently installed Stata utilities need updating, `update utilities` copies and installs the new utilities from the update source. `update utilities` is seldom used by itself; it is called automatically by `update all`.

`update all` does the same as `update ado` followed by `update executable` and `update utilities`. This is what you should type to check for and install updates.

`update swap` completes the installation of a newly updated executable by automatically swapping the newly downloaded executable with the currently running executable. During this process, Stata will shut down to perform the swap before automatically launching the new executable. This command is necessary after either the `update executable` or the `update all` command when a new executable is downloaded.

`set update_query` determines if `update query` is to be automatically performed when Stata is launched. Only Mac and Windows platforms can be set for automatic updating.

`set update_interval #` sets the number of days to elapse before performing the next automatic update query. The default `#` is 7. The interval starts from the last time an update query was performed (automatically or manually). Only Mac and Windows platforms can be set for automatic updating.

`set update_prompt` determines whether a dialog is to be displayed before performing an automatic update query. The dialog allows you to perform an update query now, perform one the next

time Stata is launched, perform one after the next interval has passed, or disable automatic update query. Only Mac and Windows platforms can be set for automatic updating.

## Options

`from(location)` specifies the location of the update source. You can specify the `from()` option on the individual `update` commands or use the `update from` command. Which you do makes no difference. You typically do not need to use this option.

`into(dirname)` specifies the name of the directory into which the updates are to be copied. *dirname* can be a directory name or a `sysdir` codeword, such as `UPDATES` or `STATA`; see [P] [sysdir](#). This option is seldom used.

For `update ado`, the default is `into(UPDATES)`, the official update directory. Network computer managers might want to specify `into()` if they want to download the updates but leave the last step—copying the files into the official directory—to do themselves.

For `update executable`, the default is `into(STATA)`, the official Stata directory. Network computer managers might want to specify `into()` so that they can copy the update into a more accessible directory. Then the last step of copying the new executable over the existing executable would be left for them to perform.

For `update utilities`, the default is `into(STATA/utilities)`. That is, the default is the directory `utilities` within the official Stata directory. We recommend you not use `into()` to change where Stata wants to update its utilities; if you do so, certain parts of Stata such as the online PDF documentation and future updates may not work properly.

`force` is used with `update executable` or `update utilities` to force downloading a new executable or utilities even if, based on the date comparison, Stata does not think it is necessary. There is seldom a reason to specify this option. There is no such option for `update ado` because, if you wanted to force the reinstallation of all ado-file updates, you would need to erase only the `UPDATES` directory. You can type `sysdir` to see where the `UPDATES` directory is on your computer; see [P] [sysdir](#).

`clear` specifies that it is okay for `update swap` to shut down this Stata and start the new one, even if data in memory have not been saved to disk.

## Remarks

`update` updates the three official components of Stata—its binary executable, ado-files, and utilities—from the official source: <http://www.stata.com>. Jumping ahead of the story, the easiest thing to do if you are connected to the Internet is to type

```
. update all
```

and follow the instructions. If Stata is up to date, `update all` will do nothing. Otherwise, it will download whatever is necessary and display detailed instructions on what, if anything, needs to be done next. If you first want to know what `update all` will do, type

```
. update query
```

`update query` will present a report comparing what you have installed with what is available and will recommend that you do nothing or that you type `update all`. You should never update one part of Stata without updating the others if more than one part has available updates, which is why `update all` will be recommended no matter what type of updates are available.

If you want only a report on what you have installed without comparing with what is available, type

```
. update
```

`update` will show you what you have installed and where it is and will recommend that you type `update query` to compare that with what is available.

Before doing any of this, you can type

```
. update from http://www.stata.com
```

but that is not really necessary because <http://www.stata.com> is the default location.

In addition to using the `update` command, you may select **Official Updates** from the **Help** menu. The menu item does the same thing as the command, but it does not provide the file redirection option `into()`, which managers of networked computers may wish to use so that they can download the updates and then copy the files to the official locations for themselves.

For examples of using `update`, see

|         |                                                                                                                                                  |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| Windows | [GSW] <a href="#">19 Updating and extending Stata—Internet functionality</a> and<br>[U] <a href="#">28 Using the Internet to keep up to date</a> |
| Mac     | [GSM] <a href="#">19 Updating and extending Stata—Internet functionality</a> and<br>[U] <a href="#">28 Using the Internet to keep up to date</a> |
| Unix    | [GSU] <a href="#">19 Updating and extending Stata—Internet functionality</a> and<br>[U] <a href="#">28 Using the Internet to keep up to date</a> |

## Notes for multiuser system administrators

There are three types of updates: ado-file updates, utilities updates, and the binary executable update. Stata's periodic updates are usually ado-file updates, but sometimes they may also include the executable or utilities.

By default, `update` installs the ado-file updates and utilities updates, which may include lots of small files. `update` is careful about how it does this. First it downloads all the files you need to a temporary place. Next it closes the connection to <http://www.stata.com> and checks the files to make sure that they are complete. Only after all that does `update` copy the files to the official UPDATES directory or the official location for utilities; see [P] [sysdir](#). This process is all designed so that should anything go wrong at any step along the way, no damage is done.

Updated binary executables, on the other hand, are just downloaded, and then you can either automatically or manually swap the executables. To automatically swap the executables, type `update swap`, and Stata will briefly quit to perform the copy before relaunching the newly downloaded executable. To manually swap the executables, exit Stata, rename the current executable, rename the updated executable, try Stata, and erase the old executable. It is very important to type `update swap` or to manually swap the executables after downloading an executable update.

`update` must have write access to both the STATA and UPDATES directories. You can obtain the names of these directories by typing `sysdir`. As system administrator, you must decide whether you are going to start Stata with such permissions (Unix users could first become superuser to ensure write access to the STATA and UPDATES directories) and trust Stata to do the right thing. That is what we recommend you do, but we provide the `into()` option for those who do not trust our recommendation.

To perform the final copying by hand, obtain the new executable by typing

```
. update executable, into(.)
```

That will place the new executable in the current directory. You need no special permissions to do this. Later you can copy the file into the appropriate place and give it the appropriate name. Type `update` without arguments; the default output will make it clear where the file goes and what its name must be. When you copy the file, be sure to make it executable by everybody.

To obtain the ado-file updates, make a new, empty directory, and then place the updates into it. For example, typing

```
. mkdir mydir
. update ado, into(mydir)
```

places the new, empty directory in the current directory under the name `mydir`. You need no special permissions to perform this step. Later, you can copy all the files in `mydir` to the official place. Type `update` without arguments; the default output will make it clear where the files go. When you copy the files, be sure to copy all of them and to make all of them readable by everybody.

## Saved results

`update` without a subcommand, `update from`, and `update query` save the following in `r()`:

### Scalars

|                                |                                       |          |
|--------------------------------|---------------------------------------|----------|
| <code>r(inst_exe)</code>       | date of executable installed          | (*)      |
| <code>r(avbl_exe)</code>       | date of executable available over web | (*) (**) |
| <code>r(inst_ado)</code>       | date of ado-files installed           | (*)      |
| <code>r(avbl_ado)</code>       | date of ado-files available over web  | (*) (**) |
| <code>r(inst_utilities)</code> | date of utilities installed           | (*)      |
| <code>r(avbl_utilities)</code> | date of utilities available over web  | (*) (**) |

### Macros

|                               |                                         |
|-------------------------------|-----------------------------------------|
| <code>r(dir_exe)</code>       | directory in which executable is stored |
| <code>r(dir_ado)</code>       | directory in which ado-files are stored |
| <code>r(dir_utilities)</code> | directory in which utilities are stored |
| <code>r(name_exe)</code>      | name of the Stata executable            |

Notes:

- \* Dates are stored as integers counting the number of days since January 1, 1960; see [\[D\] dates and times](#).
- \*\* These dates are not saved by `update` without a subcommand because `update` by itself reports information solely about the local computer and does not check what is available on the web.

## Also see

- [R] [adoupdate](#) — Update user-written ado-files
- [R] [net](#) — Install and manage user-written additions from the Internet
- [R] [ssc](#) — Install and uninstall packages from SSC
- [P] [sysdir](#) — Query and set system directories
- [U] [28 Using the Internet to keep up to date](#)
- [GSM] [19 Updating and extending Stata—Internet functionality](#)
- [GSU] [19 Updating and extending Stata—Internet functionality](#)
- [GSW] [19 Updating and extending Stata—Internet functionality](#)

Syntax

```
estimation_cmd ... [, vce(vcetype) ...]
```

| <i>vcetype</i>                                        | description                                 |
|-------------------------------------------------------|---------------------------------------------|
| Likelihood based                                      |                                             |
| <code>oim</code>                                      | observed information matrix (OIM)           |
| <code>opg</code>                                      | outer product of the gradient (OPG) vectors |
| Sandwich estimators                                   |                                             |
| <code>robust</code>                                   | Huber/White/sandwich estimator              |
| <code>cluster clustvar</code>                         | clustered sandwich estimator                |
| Replication based                                     |                                             |
| <code>bootstrap</code> [ , <i>bootstrap_options</i> ] | bootstrap estimation                        |
| <code>jackknife</code> [ , <i>jackknife_options</i> ] | jackknife estimation                        |

Description

This entry describes the `vce()` option, which is common to most estimation commands. `vce()` specifies how to estimate the variance–covariance matrix (VCE) corresponding to the parameter estimates. The standard errors reported in the table of parameter estimates are the square root of the variances (diagonal elements) of the VCE.

Options

SE/Robust

`vce(oim)` is usually the default for models fit using maximum likelihood. `vce(oim)` uses the observed information matrix (OIM); see [\[R\] ml](#).

`vce(opg)` uses the sum of the outer product of the gradient (OPG) vectors; see [\[R\] ml](#). This is the default VCE when the `technique(bhhh)` option is specified; see [\[R\] maximize](#).

`vce(robust)` uses the robust or sandwich estimator of variance. This estimator is robust to some types of misspecification so long as the observations are independent; see [\[U\] 20.16 Obtaining robust variance estimates](#).

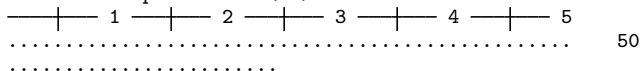
If the command allows `pweights` and you specify them, `vce(robust)` is implied; see [\[U\] 20.18.3 Sampling weights](#).





```
. jackknife: regress mpg turn trunk
(running regress on estimation sample)
```

Jackknife replications (74)



## Linear regression

|               |   |        |
|---------------|---|--------|
| Number of obs | = | 74     |
| Replications  | = | 74     |
| F( 2, 73)     | = | 66.26  |
| Prob > F      | = | 0.0000 |
| R-squared     | = | 0.5521 |
| Adj R-squared | = | 0.5395 |
| Root MSE      | = | 3.9260 |

| mpg   | Coef.     | Jackknife<br>Std. Err. | t     | P> t  | [95% Conf. Interval] |           |
|-------|-----------|------------------------|-------|-------|----------------------|-----------|
| turn  | -.7610113 | .150726                | -5.05 | 0.000 | -1.061408            | -.4606147 |
| trunk | -.3161825 | .1282326               | -2.47 | 0.016 | -.5717498            | -.0606152 |
| _cons | 55.82001  | 5.031107               | 11.09 | 0.000 | 45.79303             | 65.84699  |

Here it does not matter whether we specify the `vce(jackknife)` option or instead use the `jackknife` prefix.

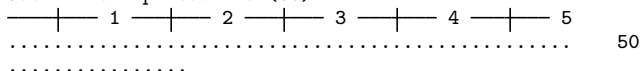
However, `vce(jackknife)` should be used in place of the `jackknife` prefix whenever available because they are not always equivalent. For example, to use the `jackknife` prefix with `clogit` properly, you must tell `jackknife` to omit whole groups rather than individual observations. Specifying `vce(jackknife)` does this automatically.

```
. use http://www.stata-press.com/data/r11/clogitid
. jackknife, cluster(id): clogit y x1 x2, group(id)
(output omitted)
```

This extra information is automatically communicated to `jackknife` by `clogit` when the `vce()` option is specified.

```
. clogit y x1 x2, group(id) vce(jackknife)
(running clogit on estimation sample)
```

Jackknife replications (66)



### Conditional (fixed-effects) logistic regression

|               |   |        |
|---------------|---|--------|
| Number of obs | = | 369    |
| Replications  | = | 66     |
| F( 2, 65)     | = | 4.58   |
| Prob > F      | = | 0.0137 |
| Pseudo R2     | = | 0.0355 |

Log likelihood = -123.41386

(Replications based on 66 clusters in id)

| y  | Coef.    | Jackknife<br>Std. Err. | t    | P> t  | [95% Conf. Interval] |          |
|----|----------|------------------------|------|-------|----------------------|----------|
| x1 | .653363  | .3010608               | 2.17 | 0.034 | .052103              | 1.254623 |
| x2 | .0659169 | .0487858               | 1.35 | 0.181 | -.0315151            | .1633489 |

## Passing options in vce()

If you wish to specify more options to the bootstrap or jackknife estimation, you can include them within the `vce()` option. Below we request 300 bootstrap replications and save the replications in `bsreg.dta`:

```
. use http://www.stata-press.com/data/r11/auto
(1978 Automobile Data)

. regress mpg turn trunk, vce(bootstrap, nodots seed(123) rep(300) saving(bsreg))
```

|                   |               |   |        |
|-------------------|---------------|---|--------|
| Linear regression | Number of obs | = | 74     |
|                   | Replications  | = | 300    |
|                   | Wald chi2(2)  | = | 127.28 |
|                   | Prob > chi2   | = | 0.0000 |
|                   | R-squared     | = | 0.5521 |
|                   | Adj R-squared | = | 0.5395 |
|                   | Root MSE      | = | 3.9260 |

| mpg   | Observed<br>Coef. | Bootstrap<br>Std. Err. | z     | P> z  | Normal-based<br>[95% Conf. Interval] |           |
|-------|-------------------|------------------------|-------|-------|--------------------------------------|-----------|
| turn  | -.7610113         | .1361786               | -5.59 | 0.000 | -1.027916                            | -.4941062 |
| trunk | -.3161825         | .1145728               | -2.76 | 0.006 | -.540741                             | -.0916239 |
| _cons | 55.82001          | 4.69971                | 11.88 | 0.000 | 46.60875                             | 65.03127  |

```
. bstat using bsreg
```

|                   |               |   |     |
|-------------------|---------------|---|-----|
| Bootstrap results | Number of obs | = | 74  |
|                   | Replications  | = | 300 |

command: regress mpg turn trunk

|       | Observed<br>Coef. | Bootstrap<br>Std. Err. | z     | P> z  | Normal-based<br>[95% Conf. Interval] |           |
|-------|-------------------|------------------------|-------|-------|--------------------------------------|-----------|
| turn  | -.7610113         | .1361786               | -5.59 | 0.000 | -1.027916                            | -.4941062 |
| trunk | -.3161825         | .1145728               | -2.76 | 0.006 | -.540741                             | -.0916239 |
| _cons | 55.82001          | 4.69971                | 11.88 | 0.000 | 46.60875                             | 65.03127  |

## Methods and formulas

By default, Stata’s maximum likelihood estimators display standard errors based on variance estimates given by the inverse of the negative Hessian (second derivative) matrix. If `vce(robust)`, `vce(cluster clustvar)`, or `pweights` is specified, standard errors are based on the robust variance estimator (see [U] 20.16 Obtaining robust variance estimates); likelihood-ratio tests are not appropriate here (see [SVY] survey), and the model  $\chi^2$  is from a Wald test. If `vce(opg)` is specified, the standard errors are based on the outer product of the gradients; this option has no effect on likelihood-ratio tests, though it does affect Wald tests.

If `vce(bootstrap)` or `vce(jackknife)` is specified, the standard errors are based on the chosen replication method; here the model  $\chi^2$  or  $F$  statistic is from a Wald test using the respective replication-based covariance matrix. The  $t$  distribution is used in the coefficient table when the `vce(jackknife)` option is specified. `vce(bootstrap)` and `vce(jackknife)` are also available with some commands that are not maximum likelihood estimators.

## Also see

[R] [bootstrap](#) — Bootstrap sampling and estimation

[R] [jackknife](#) — Jackknife estimation

[XT] [vce\\_options](#) — Variance estimators

[U] [20 Estimation and postestimation commands](#)

# Title

**view** — View files and logs

## Syntax

*Display file in Viewer*

```
view [file] ["filename"] [, asis adopath]
```

*Bring up browser pointed to specified URL*

```
view browse ["url"]
```

*Display help results in Viewer*

```
view help [topic_or_command_name]
```

*Display search results in Viewer*

```
view search keywords
```

*Display news results in Viewer*

```
view news
```

*Display net results in Viewer*

```
view net [netcmd]
```

*Display ado-results in Viewer*

```
view ado [adocmd]
```

*Display update results in Viewer*

```
view update [updatecmd]
```

*Programmer's analog to view file and view browse*

```
view view_d
```

*Programmer's analog to view help*

```
view help_d
```

*Programmer's analog to view search*

```
view search_d
```

*Programmer's analog to view net*

```
view net_d
```

*Programmer's analog to view ado*

```
view ado_d
```

*Programmer's analog to view update*

```
view update_d
```

## Menu

File > View...

## Description

**view** displays file contents in the Viewer.

**view file** displays the specified file. **file** is optional, so if you had a SMCL session log created by typing `log using mylog`, you could view it by typing `view mylog.smcl`. **view file** can properly display `.smcl` files (logs and the like), `.sthlp` files, and ASCII text files. **view file**'s **asis** option specifies that the file be displayed as straight ASCII text, regardless of the *filename*'s extension.

**view browse** opens your browser pointed to *url*. Typing

```
view browse http://www.stata.com
```

 would bring up your browser pointed to the <http://www.stata.com> web site.

**view help** does the same as the **help** command—see [R] [help](#)—but displays the result in the Viewer. For example, to review the help for Stata's **print** command, you could type `view help print`.

**view search** does the same as the **search** command—see [R] [search](#)—but displays the result in the Viewer. For instance, to search the online help for information on robust regression, you could type `view search robust regression`.

**view news** does the same as the **news** command—see [R] [news](#)—but displays the results in the Viewer. (**news** displays the latest news from <http://www.stata.com>.)

**view net** does the same as the **net** command—see [R] [net](#)—but displays the result in the Viewer. For instance, typing `view net search hausman test` would search the Internet for additions to Stata related to the Hausman test. Typing `view net from http://www.stata.com` would go to the Stata download site at <http://www.stata.com>.

**view ado** does the same as the **ado** command—see [R] [net](#)—but displays the result in the Viewer. For instance, typing `view ado dir` would show a list of files you have installed.

**view update** does the same as the **update** command—see [R] [update](#)—but displays the result in the Viewer. Typing `view update` would show the dates of what you have installed, and from there you could click to compare those dates with the latest updates available. Typing `view update query` would skip the first step and show the comparison.

The **view \*\_d** commands are more useful in programming contexts than they are interactively.

`view view_d` displays a dialog box from which you may type the name of a file or a URL to be displayed in the Viewer.

`view help_d` displays a help dialog box from which you may obtain interactive help on any Stata command.

`view search_d` displays a search dialog box from which you may obtain interactive help based on keywords.

`view net_d` displays a search dialog box from which you may search the Internet for additions to Stata (which you could then install).

`view ado_d` displays a dialog box from which you may search the user-written routines you have previously installed.

`view update_d` displays an update dialog box in which you may type the source from which updates are to be obtained.

## Options

`asis`, allowed with `view file`, specifies that the file be displayed as ASCII text, regardless of the *filename*'s extension. `view file`'s default action is to display files ending in `.smcl` and `.sthlp` as SMCL.

`adopath`, allowed with `view file`, specifies that Stata search the `S_ADO` path for *filename* and display it, if found.

## Remarks

Most users access the Viewer by selecting **File > View...** and proceeding from there. The `view` command allows you to skip that step. Some common interactive uses of `view` are

```
. view mysession.smcl
. view mysession.log
. view help print
. view help regress
. view news
. view browse http://www.stata.com
. view net search hausman test
. view net
. view ado
. view update query
```

Also, programmers find `view` useful for creating special effects.

(Continued on next page)

## Also see

[R] **help** — Display online help

[R] **net** — Install and manage user-written additions from the Internet

[R] **news** — Report Stata news

[R] **search** — Search Stata documentation

[R] **update** — Update Stata

[D] **type** — Display contents of a file

[GSM] **3 Using the Viewer**

[GSU] **3 Using the Viewer**

[GSW] **3 Using the Viewer**



Syntax

```
vwls depvar indepvars [if] [in] [weight] [, options]
```

| <i>options</i>                       | description                                                               |
|--------------------------------------|---------------------------------------------------------------------------|
| Model                                |                                                                           |
| <code>noconstant</code>              | suppress constant term                                                    |
| <code>sd(<i>varname</i>)</code>      | variable containing estimate of conditional standard deviation            |
| Reporting                            |                                                                           |
| <code>level(#)</code>                | set confidence level; default is <code>level(95)</code>                   |
| <code>display_options</code>         | control spacing and display of omitted variables and base and empty cells |
| <sup>†</sup> <code>coeflegend</code> | display coefficients' legend instead of coefficient table                 |

<sup>†</sup>`coeflegend` does not appear in the dialog box.  
*indepvars* may contain factor variables; see [U] 11.4.3 **Factor variables**.  
`bootstrap`, `by`, `jackknife`, `rolling`, and `statsby` are allowed; see [U] 11.1.10 **Prefix commands**.  
Weights are not allowed with the `bootstrap` prefix.  
`fweights` are allowed; see [U] 11.1.6 **weight**.  
See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Menu

Statistics > Linear models and related > Other > Variance-weighted least squares

Description

`vwls` estimates a linear regression using variance-weighted least squares. It differs from ordinary least-squares (OLS) regression in that it does not assume homogeneity of variance, but requires that the conditional variance of *depvar* be estimated prior to the regression. The estimated variance need not be constant across observations. `vwls` treats the estimated variance as if it were the true variance when it computes the standard errors of the coefficients.

You must supply an estimate of the conditional standard deviation of *depvar* to `vwls` by using the `sd(varname)` option, or you must have grouped data with the groups defined by the *indepvars* variables. In the latter case, `vwls` treats all *indepvars* as categorical variables, computes the mean and standard deviation of *depvar* separately for each subgroup, and computes the regression of the subgroup means on *indepvars*.

`regress` with analytic weights can be used to produce another kind of “variance-weighted least squares”; see [Remarks](#) for an explanation of the difference.

## Options

### Model

`noconstant`; see [\[R\] estimation options](#).

`sd(varname)` is an estimate of the conditional standard deviation of *depvar* (that is, it can vary observation by observation). All values of *varname* must be  $> 0$ . If you specify `sd()`, you cannot use `fweights`.

If `sd()` is not given, the data will be grouped by *indepvars*. Here *indepvars* are treated as categorical variables, and the means and standard deviations of *depvar* for each subgroup are calculated and used for the regression. Any subgroup for which the standard deviation is zero is dropped.

### Reporting

`level(#)`; see [\[R\] estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [\[R\] estimation options](#).

The following option is available with `vwls` but is not shown in the dialog box:

`coeflegend`; see [\[R\] estimation options](#).

## Remarks

The `vwls` command is intended for use with two special—and different—types of data. The first contains data that consist of measurements from physical science experiments in which all error is due solely to measurement errors and the sizes of the measurement errors are known.

You can also use variance-weighted least-squares linear regression for certain problems in categorical data analysis, such as when all the independent variables are categorical and the outcome variable is either continuous or a quantity that can sensibly be averaged. If each of the subgroups defined by the categorical variables contains a reasonable number of subjects, then the variance of the outcome variable can be estimated independently within each subgroup. For the purposes of estimation, `vwls` treats each subgroup as one observation, with the dependent variable being the subgroup mean of the outcome variable.

The `vwls` command fits the model

$$y_i = \mathbf{x}_i\boldsymbol{\beta} + \varepsilon_i$$

where the errors  $\varepsilon_i$  are independent normal random variables with the distribution  $\varepsilon_i \sim N(0, \nu_i)$ . The independent variables  $\mathbf{x}_i$  are assumed to be known without error.

As described above, `vwls` assumes that you already have estimates  $s_i^2$  for the variances  $\nu_i$ . The error variance is not estimated in the regression. The estimates  $s_i^2$  are used to compute the standard errors of the coefficients; see [Methods and formulas](#) below.

In contrast, weighted OLS regression assumes that the errors have the distribution  $\varepsilon_i \sim N(0, \sigma^2/w_i)$ , where the  $w_i$  are known weights and  $\sigma^2$  is an unknown parameter that is estimated in the regression. This is the difference from variance-weighted least squares: in weighted OLS, the magnitude of the error variance is estimated in the regression using all the data.

## ➤ Example 1

An artificial, but informative, example illustrates the difference between variance-weighted least squares and weighted OLS.

We measure the quantities  $x_i$  and  $y_i$  and estimate that the standard deviation of  $y_i$  is  $s_i$ . We enter the data into Stata:

```
. use http://www.stata-press.com/data/r11/vwlsxmpl
. list
```

|    | x | y   | s  |
|----|---|-----|----|
| 1. | 1 | 1.2 | .5 |
| 2. | 2 | 1.9 | .5 |
| 3. | 3 | 3.2 | 1  |
| 4. | 4 | 4.3 | 1  |
| 5. | 5 | 4.9 | 1  |
| 6. | 6 | 6.0 | 2  |
| 7. | 7 | 7.2 | 2  |
| 8. | 8 | 7.9 | 2  |

Because we want observations with smaller variance to carry larger weight in the regression, we compute an OLS regression with analytic weights proportional to the inverse of the squared standard deviations:

```
. regress y x [aweight=s^(-2)]
(sum of wgt is 1.1750e+01)
```

| Source   | SS         | df | MS         | Number of obs = | 8      |
|----------|------------|----|------------|-----------------|--------|
| Model    | 22.6310183 | 1  | 22.6310183 | F( 1, 6) =      | 702.26 |
| Residual | .193355117 | 6  | .032225853 | Prob > F =      | 0.0000 |
|          |            |    |            | R-squared =     | 0.9915 |
|          |            |    |            | Adj R-squared = | 0.9901 |
| Total    | 22.8243734 | 7  | 3.26062477 | Root MSE =      | .17952 |

| y     | Coef.    | Std. Err. | t     | P> t  | [95% Conf. Interval] |          |
|-------|----------|-----------|-------|-------|----------------------|----------|
| x     | .9824683 | .0370739  | 26.50 | 0.000 | .8917517             | 1.073185 |
| _cons | .1138554 | .1120078  | 1.02  | 0.349 | -.1602179            | .3879288 |

If we compute a variance-weighted least-squares regression by using `vwls`, we get the same results for the coefficient estimates but very different standard errors:

```
. vwls y x, sd(s)
```

|                                            |                 |        |
|--------------------------------------------|-----------------|--------|
| Variance-weighted least-squares regression | Number of obs = | 8      |
| Goodness-of-fit chi2(6) = 0.28             | Model chi2(1) = | 33.24  |
| Prob > chi2 = 0.9996                       | Prob > chi2 =   | 0.0000 |

| y     | Coef.    | Std. Err. | z    | P> z  | [95% Conf. Interval] |          |
|-------|----------|-----------|------|-------|----------------------|----------|
| x     | .9824683 | .170409   | 5.77 | 0.000 | .6484728             | 1.316464 |
| _cons | .1138554 | .51484    | 0.22 | 0.825 | -.8952124            | 1.122923 |

Although the values of  $y_i$  were nicely linear with  $x_i$ , the `vwls` regression used the large estimates for the standard deviations to compute large standard errors for the coefficients. For weighted OLS regression, however, the scale of the analytic weights has no effect on the standard errors of the coefficients—only the relative proportions of the analytic weights affect the regression.

If we are sure of the sizes of our error estimates for  $y_i$ , using `vwls` is valid. However, if we can estimate only the relative proportions of error among the  $y_i$ , then `vwls` is not appropriate.



➤ Example 2

Let’s now consider an example of the use of `vwls` with categorical data. Suppose that we have blood pressure data for  $n = 400$  subjects, categorized by gender and race (black or white). Here is a description of the data:

```
. use http://www.stata-press.com/data/r11/bp
. table gender race, c(mean bp sd bp freq) row col format(%8.1f)
```

| Gender | Race  |       |       |
|--------|-------|-------|-------|
|        | White | Black | Total |
| Female | 117.1 | 118.5 | 117.8 |
|        | 10.3  | 11.6  | 10.9  |
|        | 100   | 100   | 200   |
| Male   | 122.1 | 125.8 | 124.0 |
|        | 10.6  | 15.5  | 13.3  |
|        | 100   | 100   | 200   |
| Total  | 119.6 | 122.2 | 120.9 |
|        | 10.7  | 14.1  | 12.6  |
|        | 200   | 200   | 400   |

Performing a variance-weighted regression using `vwls` gives

```
. vwls bp gender race
Variance-weighted least-squares regression Number of obs = 400
Goodness-of-fit chi2(1) = 0.88 Model chi2(2) = 27.11
Prob > chi2 = 0.3486 Prob > chi2 = 0.0000
```

| bp     | Coef.    | Std. Err. | z      | P> z  | [95% Conf. Interval] |          |
|--------|----------|-----------|--------|-------|----------------------|----------|
| gender | 5.876522 | 1.170241  | 5.02   | 0.000 | 3.582892             | 8.170151 |
| race   | 2.372818 | 1.191683  | 1.99   | 0.046 | .0371631             | 4.708473 |
| _cons  | 116.6486 | .9296297  | 125.48 | 0.000 | 114.8266             | 118.4707 |

By comparison, an OLS regression gives the following result:

| . regress bp gender race |            |     |            |                 |        |
|--------------------------|------------|-----|------------|-----------------|--------|
| Source                   | SS         | df  | MS         |                 |        |
| Model                    | 4485.66639 | 2   | 2242.83319 | Number of obs = | 400    |
| Residual                 | 58442.7305 | 397 | 147.210908 | F( 2, 397) =    | 15.24  |
|                          |            |     |            | Prob > F =      | 0.0000 |
|                          |            |     |            | R-squared =     | 0.0713 |
|                          |            |     |            | Adj R-squared = | 0.0666 |
| Total                    | 62928.3969 | 399 | 157.71528  | Root MSE =      | 12.133 |

| bp     | Coef.    | Std. Err. | t      | P> t  | [95% Conf. Interval] |          |
|--------|----------|-----------|--------|-------|----------------------|----------|
| gender | 6.1775   | 1.213305  | 5.09   | 0.000 | 3.792194             | 8.562806 |
| race   | 2.5875   | 1.213305  | 2.13   | 0.034 | .2021938             | 4.972806 |
| _cons  | 116.4862 | 1.050753  | 110.86 | 0.000 | 114.4205             | 118.552  |

Note the larger value for the `race` coefficient (and smaller  $p$ -value) in the OLS regression. The assumption of homogeneity of variance in OLS means that the mean for black men pulls the regression line higher than in the `vwls` regression, which takes into account the larger variance for black men and reduces its effect on the regression.

◀

## Saved results

`vwls` saves the following in `e()`:

### Scalars

|                         |                                    |
|-------------------------|------------------------------------|
| <code>e(N)</code>       | number of observations             |
| <code>e(df_m)</code>    | model degrees of freedom           |
| <code>e(chi2)</code>    | model $\chi^2$                     |
| <code>e(df_gf)</code>   | goodness-of-fit degrees of freedom |
| <code>e(chi2_gf)</code> | goodness-of-fit $\chi^2$           |
| <code>e(rank)</code>    | rank of <code>e(V)</code>          |

### Macros

|                            |                                                                |
|----------------------------|----------------------------------------------------------------|
| <code>e(cmd)</code>        | <code>vwls</code>                                              |
| <code>e(cmdline)</code>    | command as typed                                               |
| <code>e(depvar)</code>     | name of dependent variable                                     |
| <code>e(properties)</code> | <code>b V</code>                                               |
| <code>e(predict)</code>    | program used to implement <code>predict</code>                 |
| <code>e(asbalanced)</code> | factor variables <code>fvset</code> as <code>asbalanced</code> |
| <code>e(asobserved)</code> | factor variables <code>fvset</code> as <code>asobserved</code> |

### Matrices

|                   |                                              |
|-------------------|----------------------------------------------|
| <code>e(b)</code> | coefficient vector                           |
| <code>e(V)</code> | variance-covariance matrix of the estimators |

### Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`vwls` is implemented as an ado-file.

Let  $\mathbf{y} = (y_1, y_2, \dots, y_n)'$  be the vector of observations of the dependent variable, where  $n$  is the number of observations. When `sd()` is specified, let  $s_1, s_2, \dots, s_n$  be the standard deviations supplied by `sd()`. For categorical data, when `sd()` is not given, the means and standard deviations of  $y$  for each subgroup are computed, and  $n$  becomes the number of subgroups,  $\mathbf{y}$  is the vector of subgroup means, and  $s_i$  are the standard deviations for the subgroups.

Let  $\mathbf{V} = \text{diag}(s_1^2, s_2^2, \dots, s_n^2)$  denote the estimate of the variance of  $\mathbf{y}$ . Then the estimated regression coefficients are

$$\mathbf{b} = (\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{y}$$

and their estimated covariance matrix is

$$\widehat{\text{Cov}}(\mathbf{b}) = (\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}$$

A statistic for the goodness of fit of the model is

$$Q = (\mathbf{y} - \mathbf{X}\mathbf{b})' \mathbf{V}^{-1} (\mathbf{y} - \mathbf{X}\mathbf{b})$$

where  $Q$  has a  $\chi^2$  distribution with  $n - k$  degrees of freedom ( $k$  is the number of independent variables plus the constant, if any).

## References

- Gini, R., and J. Pasquini. 2006. [Automatic generation of documents](#). *Stata Journal* 6: 22–39.
- Grizzle, J. E., C. F. Starmer, and G. G. Koch. 1969. Analysis of categorical data by linear models. *Biometrics* 25: 489–504.
- Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 2007. *Numerical Recipes in C: The Art of Scientific Computing*. 3rd ed. Cambridge: Cambridge University Press.

## Also see

- [R] [vwls postestimation](#) — Postestimation tools for `vwls`
- [R] [regress](#) — Linear regression
- [U] [11.1.6 weight](#)
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `vwls`:

| command                | description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>     | VCE and estimation sample summary                                                                   |
| <code>estimates</code> | cataloging estimation results                                                                       |
| <code>lincom</code>    | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>linktest</code>  | link test for model specification                                                                   |
| <code>margins</code>   | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>     | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>   | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code> | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>test</code>      | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>    | Wald tests of nonlinear hypotheses                                                                  |

See the corresponding entries in the *Base Reference Manual* for details.

Syntax for predict

```
predict [type] newvar [if] [in] [, xb stdp]
```

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

Menu

Statistics > Postestimation > Predictions, residuals, etc.

Options for predict

- `xb`, the default, calculates the linear prediction.
- `stdp` calculates the standard error of the linear prediction.

Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [vwls](#) — Variance-weighted least squares

[U] [20 Estimation and postestimation commands](#)



# Title

**which** — Display location and version for an ado-file

## Syntax

```
which fname [.ftype] [, all]
```

## Description

**which** looks for *fname.ftype* along the `S_ADO` path. If Stata finds the file, **which** displays the full path and filename, along with, if the file is ASCII, all lines in the file that begin with “\*!” in the first column. If Stata cannot find the file, **which** issues the message “file not found along adopath” and sets the return code to 111. *ftype* must be a file type for which Stata usually looks along the ado-path to find. Allowable *ftypes* are

```
.ado, .class, .dlg, .idlg, .sthlp, .ihlp, .hlp, .key, .maint, .mata, .mlib, .mo, .mnu,
.plugin, .scheme, and .style
```

If *ftype* is omitted, **which** assumes `.ado`. When searching for `.ado` files, if Stata cannot find the file, Stata then checks to see if *fname* is a built-in Stata command, allowing for valid abbreviations. If it is, the message “built-in command” is displayed; if not, the message “command not found as either built-in or ado-file” is displayed, and the return code is set to 111.

For information about internal version control, see [\[P\] version](#).

## Option

**all** forces **which** to report the location of all files matching the *fname.ftype* found along the search path. The default is to report just the first one found.

## Remarks

If you write programs, you know that you make changes to the programs over time. If you are like us, you also end up with multiple versions of the program stored on your disk, perhaps in different directories. You may even have given copies of your programs to other Stata users, and you may not remember which version of a program you or your friends are using. The **which** command helps you solve this problem.

### ► Example 1

The **which** command displays the path for *filename.ado* and any lines in the code that begin with “\*!”. For example, we might want information about the **notes** command, described in [\[D\] notes](#), which is an ado-file written by StataCorp. Here is what happens when we type **which notes**:

```
. which notes
C:\Program Files\Stata11\ado\base\n\notes.ado
*! version 2.0.9 09feb2009
```

`which` displays the path for the `notes.ado` file and also a line beginning with “\*!” that indicates the version of the file. This is how we, at StataCorp, do version control—see [U] [18.11.1 Version](#) for an explanation of our version control numbers.

We do not need to be so formal. `which` will display anything typed after lines that begin with ‘\*!’. For instance, we might write `myprog.ado`:

```
. which myprog
.\myprog.ado
*! first written 1/03/2009
*! bug fix on 1/05/2009 (no variance case)
*! updated 1/24/2009 to include noconstant option
*! still suspicious if variable takes on only two values
```

It does not matter where in the program the lines beginning with \*! are—`which` will list them (in particular, our “still suspicious” comment was buried about 50 lines down in the code). All that is important is that the \*! marker appear in the first two columns of a line.



## ▷ Example 2

If we type `which command`, where *command* is a built-in command rather than an ado-file, Stata responds with

```
. which summarize
built-in command: summarize
```

If *command* was neither a built-in command nor an ado-file, Stata would respond with

```
. which junk
command junk not found as either built-in or ado-file
r(111);
```



## Also see

[P] [findfile](#) — Find file in path

[P] [version](#) — Version control

[U] [17 Ado-files](#)

[U] [18.11.1 Version](#)

## Syntax

`xi` [`, prefix(string) noomit`] *term(s)*

`xi` [`, prefix(string) noomit`] : *any\_stata\_command varlist\_with\_terms ...*

where a *term* has the form

|                                                          |    |                                                          |
|----------------------------------------------------------|----|----------------------------------------------------------|
| <code>i.varname</code>                                   | or | <code>I.varname</code>                                   |
| <code>i.varname<sub>1</sub>*i.varname<sub>2</sub></code> |    | <code>I.varname<sub>1</sub>*I.varname<sub>2</sub></code> |
| <code>i.varname<sub>1</sub>*varname<sub>3</sub></code>   |    | <code>I.varname<sub>1</sub>*varname<sub>3</sub></code>   |
| <code>i.varname<sub>1</sub> varname<sub>3</sub></code>   |    | <code>I.varname<sub>1</sub> varname<sub>3</sub></code>   |

*varname*, *varname<sub>1</sub>*, and *varname<sub>2</sub>* denote numeric or string categorical variables. *varname<sub>3</sub>* denotes a continuous, numeric variable.

## Menu

Data > Create or change data > Other variable-creation commands > Interaction expansion

Most commands in Stata now allow factor variables; see [\[U\] 11.4.3 Factor variables](#). To determine if a command allows factor variables, see the information printed below the options table for the command. If the command allows factor variables, it will say something like “*indepvars* may contain factor variables”.

We recommend that you use factor variables instead of `xi` if a command allows factor variables.

We include [\[R\] xi](#) in our documentation so that readers can consult it when using a Stata command that does not allow factor variables.

## Description

`xi` expands terms containing categorical variables into indicator (also called dummy) variable sets by creating new variables and, in the second syntax (`xi : any_stata_command`), executes the specified command with the expanded terms. The dummy variables created are

|                                                          |                                                                                                                                                                                                                                                  |
|----------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>i.varname</code>                                   | creates dummies for categorical variable <i>varname</i>                                                                                                                                                                                          |
| <code>i.varname<sub>1</sub>*i.varname<sub>2</sub></code> | creates dummies for categorical variables <i>varname<sub>1</sub></i> and <i>varname<sub>2</sub></i> :<br>all interactions and main effects                                                                                                       |
| <code>i.varname<sub>1</sub>*varname<sub>3</sub></code>   | creates dummies for categorical variable <i>varname<sub>1</sub></i> and continuous variable <i>varname<sub>3</sub></i> :<br>all interactions and main effects                                                                                    |
| <code>i.varname<sub>1</sub> varname<sub>3</sub></code>   | creates dummies for categorical variable <i>varname<sub>1</sub></i> and continuous variable <i>varname<sub>3</sub></i> :<br>all interactions and main effect of <i>varname<sub>3</sub></i> ,<br>but no main effect of <i>varname<sub>1</sub></i> |

## Options

`prefix(string)` allows you to choose a prefix other than `_I` for the newly created interaction variables. The prefix cannot be longer than four characters. By default, `xi` will create interaction variables starting with `_I`. When you use `xi`, it drops all previously created interaction variables starting with the prefix specified in the `prefix(string)` option or with `_I` by default. Therefore, if you want to keep the variables with a certain prefix, specify a different prefix in the `prefix(string)` option.

`noomit` prevents `xi` from omitting groups. This option provides a way to generate an indicator variable for every category having one or more variables, which is useful when combined with the `noconstant` option of an estimation command.

## Remarks

Remarks are presented under the following headings:

*Background*  
*Indicator variables for simple effects*  
*Controlling the omitted dummy*  
*Categorical variable interactions*  
*Interactions with continuous variables*  
*Using xi: Interpreting output*  
*How xi names variables*  
*xi as a command rather than a command prefix*  
*Warnings*

`xi` provides a convenient way to include dummy or indicator variables when fitting a model (say, with `regress` or `logistic`). For instance, assume that the categorical variable `agegrp` contains 1 for ages 20–24, 2 for ages 25–39, 3 for ages 40–44, etc. Typing

```
. xi: logistic outcome weight i.agegrp bp
```

estimates a logistic regression of `outcome` on `weight`, dummies for each `agegrp` category, and `bp`. That is, `xi` searches out and expands terms starting with “`i.`” or “`I.`” but ignores the other variables. `xi` will expand both numeric and string categorical variables, so if you had a string variable `race` containing “white”, “black”, and “other”, typing

```
. xi: logistic outcome weight bp i.agegrp i.race
```

would include indicator variables for the race group as well.

The `i.` indicator variables `xi` expands may appear anywhere in the *varlist*, so

```
. xi: logistic outcome i.agegrp weight i.race bp
```

would fit the same model.

You can also create interactions of categorical variables; typing

```
xi: logistic outcome weight bp i.agegrp*i.race
```

fits a model with indicator variables for all `agegrp` and `race` combinations, including the `agegrp` and `race` main-effect terms (i.e., the terms that are created when you just type `i.agegrp i.race`).

You can interact dummy variables with continuous variables; typing

```
xi: logistic outcome bp i.agegrp*weight i.race
```

fits a model with indicator variables for all `agegrp` categories interacted with `weight`, plus the main-effect terms `weight` and `i.agegrp`.

You can get the interaction terms without the `agegrp` main effect (but with the `weight` main effect) by typing

```
xi: logistic outcome bp i.agegrp|weight i.race
```

You can also include multiple interactions:

```
xi: logistic outcome bp i.agegrp*weight i.agegrp*i.race
```

We will now back up and describe the construction of dummy variables in more detail.

## Background

The terms *continuous*, *categorical*, and *indicator* or *dummy* variables are used below. Continuous variables measure something—such as height or weight—and at least conceptually can take on any real number over some range. Categorical variables, on the other hand, take on a finite number of values, each denoting membership in a subclass—for example, excellent, good, and poor, which might be coded 0, 1, 2, or 1, 2, 3, or even “Excellent”, “Good”, and “Poor”. An indicator or dummy variable—the terms are used interchangeably—is a special type of two-valued categorical variable that contains values 0, denoting false, and 1, denoting true. The information contained in any  $k$ -valued categorical variable can be equally well represented by  $k$  indicator variables. Instead of one variable recording values representing excellent, good, and poor, you can have three indicator variables, indicating the truth or falseness of “result is excellent”, “result is good”, and “result is poor”.

`xi` provides a convenient way to convert categorical variables to dummy or indicator variables when you fit a model (say, with `regress` or `logistic`).

### ► Example 1

For instance, assume that the categorical variable `agegrp` contains 1 for ages 20–24, 2 for ages 25–39, and 3 for ages 40–44. (There is no one over 44 in our data.) As it stands, `agegrp` would be a poor candidate for inclusion in a model even if we thought age affected the outcome. The reason is that the coding would restrict the effect of being in the second age group to be twice the effect of being in the first, and, similarly, the effect of being in the third to be three times the first. That is, if we fit the model,

$$y = \beta_0 + \beta_1 \text{agegrp} + X\beta_2$$

the effect of being in the first age group is  $\beta_1$ , the second  $2\beta_1$ , and the third  $3\beta_1$ . If the coding 1, 2, and 3 is arbitrary, we could just as well have coded the age groups 1, 4, and 9, making the effects  $\beta_1$ ,  $4\beta_1$ , and  $9\beta_1$ .

The solution is to convert the categorical variable `agegrp` to a set of indicator variables,  $a_1$ ,  $a_2$ , and  $a_3$ , where  $a_i$  is 1 if the individual is a member of the  $i$ th age group and 0 otherwise. We can then fit the model

$$y = \beta_0 + \beta_{11}a_1 + \beta_{12}a_2 + \beta_{13}a_3 + X\beta_2$$

The effect of being in age group 1 is now  $\beta_{11}$ ; 2,  $\beta_{12}$ ; and 3,  $\beta_{13}$ ; and these results are independent of our (arbitrary) coding. The only difficulty at this point is that the model is unidentified in the sense that there are an infinite number of  $(\beta_0, \beta_{11}, \beta_{12}, \beta_{13})$  that fit the data equally well.

To see this, pretend that  $(\beta_0, \beta_{11}, \beta_{12}, \beta_{13}) = (1, 1, 3, 4)$ . The predicted values of  $y$  for the various age groups are

$$y = \begin{cases} 1 + 1 + X\beta_2 = 2 + X\beta_2 & \text{(age group 1)} \\ 1 + 3 + X\beta_2 = 4 + X\beta_2 & \text{(age group 2)} \\ 1 + 4 + X\beta_2 = 5 + X\beta_2 & \text{(age group 3)} \end{cases}$$

Now pretend that  $(\beta_0, \beta_{11}, \beta_{12}, \beta_{13}) = (2, 0, 2, 3)$ . The predicted values of  $y$  are

$$y = \begin{cases} 2 + 0 + X\beta_2 = 2 + X\beta_2 & \text{(age group 1)} \\ 2 + 2 + X\beta_2 = 4 + X\beta_2 & \text{(age group 2)} \\ 2 + 3 + X\beta_2 = 5 + X\beta_2 & \text{(age group 3)} \end{cases}$$

These two sets of predictions are indistinguishable: for age group 1,  $y = 2 + X\beta_2$  regardless of the coefficient vector used, and similarly for age groups 2 and 3. This arises because we have three equations and four unknowns. Any solution is as good as any other, and, for our purposes, we merely need to choose one of them. The popular selection method is to set the coefficient on the first indicator variable to 0 (as we have done in our second coefficient vector). This is equivalent to fitting the model

$$y = \beta_0 + \beta_{12}a_2 + \beta_{13}a_3 + X\beta_2$$

How we select a particular coefficient vector (identifies the model) does not matter. It does, however, affect the *interpretation* of the coefficients.

For instance, we could just as well choose to omit the second group. In our artificial example, this would yield  $(\beta_0, \beta_{11}, \beta_{12}, \beta_{13}) = (4, -2, 0, 1)$  instead of  $(2, 0, 2, 3)$ . These coefficient vectors are the same in the sense that

$$y = \begin{cases} 2 + 0 + X\beta_2 = 2 + X\beta_2 = 4 - 2 + X\beta_2 & \text{(age group 1)} \\ 2 + 2 + X\beta_2 = 4 + X\beta_2 = 4 + 0 + X\beta_2 & \text{(age group 2)} \\ 2 + 3 + X\beta_2 = 5 + X\beta_2 = 4 + 1 + X\beta_2 & \text{(age group 3)} \end{cases}$$

But what does it mean that  $\beta_{13}$  can just as well be 3 or 1? We obtain  $\beta_{13} = 3$  when we set  $\beta_{11} = 0$ , so  $\beta_{13} = \beta_{13} - \beta_{11}$  and  $\beta_{13}$  measures the difference between age groups 3 and 1.

In the second case, we obtain  $\beta_{13} = 1$  when we set  $\beta_{12} = 0$ , so  $\beta_{13} - \beta_{12} = 1$  and  $\beta_{13}$  measures the difference between age groups 3 and 2. There is no inconsistency. According to our  $\beta_{12} = 0$  model, the difference between age groups 3 and 1 is  $\beta_{13} - \beta_{11} = 1 - (-2) = 3$ , the same result we got in the  $\beta_{11} = 0$  model.

◀

## ► Example 2

The issue of interpretation is important because it can affect the way we discuss results. Imagine that we are studying recovery after a coronary bypass operation. Assume that the age groups are children under 13 (we have two of them), young adults under 25 (we have a handful of them), adults under 46 (of which we have even more), mature adults under 56, older adults under 65, and elderly adults. We follow the prescription of omitting the first group, so all our results are reported relative to children under 13. While there is nothing statistically wrong with this, readers will be suspicious when we make statements like “compared with young children, older and elder adults . . .”. Moreover, we will probably have to end each statement with “although results are not statistically significant” because we have only two children in our comparison group. Of course, even with results reported in this way, we can do reasonable comparisons (say, with mature adults), but we will have to do extra work to perform the appropriate linear hypothesis test using Stata’s `test` command.

Here it would be better to force the omitted group to be more reasonable, such as mature adults. There is, however, a generic rule for automatic comparison group selection that, although less popular, tends to work better than the omit-the-first-group rule. That rule is to omit the most prevalent group. The most prevalent is usually a reasonable baseline.



In any case, the prescription for categorical variables is

1. Convert each  $k$ -valued categorical variable to  $k$  indicator variables.
2. Drop one of the  $k$  indicator variables; any one will do, but dropping the first is popular, dropping the most prevalent is probably better in terms of having the computer guess at a reasonable interpretation, and dropping a specified one often eases interpretation the most.
3. Fit the model on the remaining  $k - 1$  indicator variables.

`xi` automates this procedure.

We will now consider each of `xi`'s features in detail.

## Indicator variables for simple effects

When you type `i.varname`, `xi` internally tabulates *varname* (which may be a string or a numeric variable) and creates indicator (dummy) variables for each observed value, omitting the indicator for the smallest value. For instance, say that `agegrp` takes on the values 1, 2, 3, and 4. Typing

```
xi: logistic outcome i.agegrp
```

creates indicator variables named `_Iagegrp_2`, `_Iagegrp_3`, and `_Iagegrp_4`. (`xi` chooses the names and tries to make them readable; `xi` guarantees that the names are unique.) The expanded logistic model is

```
. logistic outcome _Iagegrp_2 _Iagegrp_3 _Iagegrp_4
```

Afterward, you can drop the new variables `xi` leaves behind by typing `'drop _I*'` (note the capitalization).

`xi` provides the following features when you type `i.varname`:

- *varname* may be string or numeric.
- Dummy variables are created automatically.
- By default, the dummy-variable set is identified by dropping the dummy corresponding to the smallest value of the variable (how to specify otherwise is discussed below).
- The new dummy variables are left in your dataset. By default, the names of the new dummy variables start with `_I`; therefore, you can drop them by typing `'drop _I*'`. You do not have to do this; each time you use `xi`, any automatically generated dummies with the same prefix as the one specified in the `prefix(string)` option, or `_I` by default, are dropped and new ones are created.
- The new dummy variables have variable labels so that you can determine what they correspond to by typing `'describe'`.
- `xi` may be used with any Stata command (not just `logistic`).

## Controlling the omitted dummy

By default, `i.varname` omits the dummy corresponding to the smallest value of `varname`; for a string variable, this is interpreted as dropping the first in an alphabetical, case-sensitive sort. `xi` provides two alternatives to dropping the first: `xi` will drop the dummy corresponding to the most prevalent value of `varname`, or `xi` will let you choose the particular dummy to be dropped.

To change `xi`'s behavior to dropping the most prevalent dummy, type

```
. char _dta[omit] prevalent
```

although whether you type “prevalent” or “yes” or anything else does not matter. Setting this characteristic affects the expansion of all categorical variables in the dataset. If you resave your dataset, the prevalent preference will be remembered. If you want to change the behavior back to the default drop-the-first rule, type

```
. char _dta[omit]
```

to clear the characteristic.

Once you set `_dta[omit]`, `i.varname` omits the dummy corresponding to the most prevalent value of `varname`. Thus the coefficients on the dummies have the interpretation of change from the most prevalent group. For example,

```
. char _dta[omit] prevalent
. xi: regress y i.agegrp
```

might create `_Iagegrp_1` through `_Iagegrp_4`, resulting in `_Iagegrp_2` being omitted if `agegrp = 2` is most common (as opposed to the default dropping of `_Iagegrp_1`). The model is then

$$y = b_0 + b_1 \_Iagegrp\_1 + b_3 \_Iagegrp\_3 + b_4 \_Iagegrp\_4 + u$$

Then

$$\begin{array}{ll} \text{Predicted } y \text{ for agegrp } 1 = b_0 + b_1 & \text{Predicted } y \text{ for agegrp } 3 = b_0 + b_3 \\ \text{Predicted } y \text{ for agegrp } 2 = b_0 & \text{Predicted } y \text{ for agegrp } 4 = b_0 + b_4 \end{array}$$

Thus the model's reported *t* or *Z* statistics are for a test of whether each group is different from the most prevalent group.

Perhaps you wish to omit the dummy for `agegrp 3` instead. You do this by setting the variable's `omit` characteristic:

```
. char agegrp[omit] 3
```

This overrides `_dta[omit]` if you have set it. Now when you type

```
. xi: regress y i.agegrp
```

`_Iagegrp_3` will be omitted, and you will fit the model

$$y = b'_0 + b'_1 \_Iagegrp\_1 + b'_2 \_Iagegrp\_2 + b'_4 \_Iagegrp\_4 + u$$

Later if you want to return to the default omission, type

```
. char agegrp[omit]
```

to clear the characteristic.

In summary, `i.varname` omits the first group by default, but if you define

```
. char _dta[omit] prevalent
```



the default behavior changes to dropping the most prevalent group. Either way, if you define a characteristic of the form

```
. char varname[omit] #
```

or, if *varname* is a string,

```
. char varname[omit] string-literal
```

the specified value will be omitted.

Examples:

```
. char agegrp[omit] 1
. char race[omit] White (for race, a string variable)
. char agegrp[omit] (to restore default for agegrp)
```

## Categorical variable interactions

`i.varname1*i.varname2` creates the dummy variables associated with the interaction of the categorical variables *varname<sub>1</sub>* and *varname<sub>2</sub>*. The identification rules—which categories are omitted—are the same as those for `i.varname`. For instance, assume that `agegrp` takes on four values and `race` takes on three values. Typing

```
. xi: regress y i.agegrp*i.race
```

results in

|                                                                  |               |
|------------------------------------------------------------------|---------------|
| model:                                                           | dummies for:  |
| $y = a + b_2 \_Iagegrp\_2 + b_3 \_Iagegrp\_3 + b_4 \_Iagegrp\_4$ | (agegrp)      |
| $+ c_2 \_Irace\_2 + c_3 \_Irace\_3$                              | (race)        |
| $+ d_{22} \_IageXrac\_2\_2 + d_{23} \_IageXrac\_2\_3$            |               |
| $+ d_{32} \_IageXrac\_3\_2 + d_{33} \_IageXrac\_3\_3$            | (agegrp*race) |
| $+ d_{42} \_IageXrac\_4\_2 + d_{43} \_IageXrac\_4\_3$            |               |
| $+ u$                                                            |               |

That is, typing

```
. xi: regress y i.agegrp*i.race
```

is the same as typing

```
. xi: regress y i.agegrp i.race i.agegrp*i.race
```

Although there are many other ways the interaction could have been parameterized, this method has the advantage that you can test the joint significance of the interactions by typing

```
. testparm _IageXrac*
```

When you perform the estimation step, whether you specify `i.agegrp*i.race` or `i.race*i.agegrp` makes no difference (other than in the names given to the interaction terms; in the first case, the names will begin with `_IageXrac`; in the second, `_IracXage`). Thus

```
. xi: regress y i.race*i.agegrp
```

fits the same model.

You may also include multiple interactions simultaneously:

```
. xi: regress y i.agegrp*i.race i.agegrp*i.sex
```

The model fit is

model:

$$y = a + b_2\_Iagegrp\_2 + b_3\_Iagegrp\_3 + b_4\_Iagegrp\_4 + c_2\_Irace\_2 + c_3\_Irace\_3 + d_{22}\_IageXrac\_2\_2 + d_{23}\_IageXrac\_2\_3 + d_{32}\_IageXrac\_3\_2 + d_{33}\_IageXrac\_3\_3 + d_{42}\_IageXrac\_4\_2 + d_{43}\_IageXrac\_4\_3 + e_2\_Isex\_2 + f_{22}\_IageXsex\_2\_2 + f_{23}\_IageXsex\_2\_3 + f_{24}\_IageXsex\_2\_4 + u$$

dummies for:

(agegrp)  
(race)  
  
(agegrp\*race)  
  
(sex)  
(agegrp\*sex)

The agegrp dummies are (correctly) included only once.

Interactions with continuous variables

i.varname<sub>1</sub>\*varname<sub>2</sub> (as distinguished from i.varname<sub>1</sub>\*i.varname<sub>2</sub>—note the second i.) specifies an interaction of a categorical variable with a continuous variable. For instance,

```
. xi: regress y i.agegrp*wgt
```

results in the model

$$y = a + b_2\_Iagegrp\_2 + b_3\_Iagegrp\_3 + b_4\_Iagegrp\_4 + c\ wgt + d_2\_IageXwgt\_2 + d_3\_IageXwgt\_3 + d_4\_IageXwgt\_4 + u$$

(agegrp dummies)  
(continuous wgt effect)  
(agegrp\*wgt interactions)

A variation on this notation, using | rather than \*, omits the agegrp dummies. Typing

```
. xi: regress y i.agegrp|wgt
```

fits the model

$$y = a' + c'\ wgt + d'_2\_IageXwgt\_2 + d'_3\_IageXwgt\_3 + d'_4\_IageXwgt\_4 + u'$$

(continuous wgt effect)  
(agegrp\*wgt interactions)

The predicted values of y are

| agegrp*wgt model              | agegrp wgt model           |               |
|-------------------------------|----------------------------|---------------|
| $y = a + c\ wgt$              | $a' + c'\ wgt$             | if agegrp = 1 |
| $a + c\ wgt + b_2 + d_2\ wgt$ | $a' + c'\ wgt + d'_2\ wgt$ | if agegrp = 2 |
| $a + c\ wgt + b_3 + d_3\ wgt$ | $a' + c'\ wgt + d'_3\ wgt$ | if agegrp = 3 |
| $a + c\ wgt + b_4 + d_4\ wgt$ | $a' + c'\ wgt + d'_4\ wgt$ | if agegrp = 4 |

That is, typing

```
. xi: regress y i.agegrp*wgt
```

is equivalent to typing

```
. xi: regress y i.agegrp i.agegrp|wgt
```

In either case, you do not need to specify separately the continuous variable `wgt`; it is included automatically.

Using xi: Interpreting output

```
. xi: regress mpg i.rep78
i.rep78 _Irep78_1-5 (naturally coded; _Irep78_1 omitted)
(output from regress appears)
```

Interpretation: `i.rep78` expanded to the dummies `_Irep78_1`, `_Irep78_2`, ..., `_Irep78_5`. The numbers on the end are “natural” in the sense that `_Irep78_1` corresponds to `rep78 = 1`, `_Irep78_2` to `rep78 = 2`, and so on. Finally, the dummy for `rep78 = 1` was omitted.

```
. xi: regress mpg i.make
i.make _Imake_1-74 (_Imake_1 for make==AMC Concord omitted)
(output from regress appears)
```

Interpretation: `i.make` expanded to `_Imake_1`, `_Imake_2`, ..., `_Imake_74`. The coding is not natural because `make` is a string variable. `_Imake_1` corresponds to one make, `_Imake_2` to another, and so on. You can find out the coding by typing `describe`. `_Imake_1` for the AMC Concord was omitted.

How xi names variables

By default, `xi` assigns to the dummy variables it creates names having the form

*\_Istub\_groupid*

You may subsequently refer to the entire set of variables by typing `'Istub*'`. For example,

| name                       | =               | _I + stub            | + _ + groupid      | Entire set              |
|----------------------------|-----------------|----------------------|--------------------|-------------------------|
| <code>_Iagegrp_1</code>    | <code>_I</code> | <code>agegrp</code>  | <code>_ 1</code>   | <code>_Iagegrp*</code>  |
| <code>_Iagegrp_2</code>    | <code>_I</code> | <code>agegrp</code>  | <code>_ 2</code>   | <code>_Iagegrp*</code>  |
| <code>_IageXwgt_1</code>   | <code>_I</code> | <code>ageXwgt</code> | <code>_ 1</code>   | <code>_IageXwgt*</code> |
| <code>_IageXrac_1_2</code> | <code>_I</code> | <code>ageXrac</code> | <code>_ 1_2</code> | <code>_IageXrac*</code> |
| <code>_IageXrac_2_1</code> | <code>_I</code> | <code>ageXrac</code> | <code>_ 2_1</code> | <code>_IageXrac*</code> |

If you specify a prefix in the `prefix(string)` option, say, `_S`, then `xi` will name the variables starting with the prefix

*\_Sstub\_groupid*

## xi as a command rather than a command prefix

xi can be used as a command prefix or as a command by itself. In the latter form, xi merely creates the indicator and interaction variables. Typing

```
. xi: regress y i.agegrp*wtg
i.agegrp _Iagegrp_1-4 (naturally coded; _Iagegrp_1 omitted)
i.agegrp*wtg _IageXwtg_1-4 (coded as above)
(output from regress appears)
```

is equivalent to typing

```
. xi i.agegrp*wtg
i.agegrp _Iagegrp_1-4 (naturally coded; _Iagegrp_1 omitted)
i.agegrp*wtg _IageXwtg_1-4 (coded as above)
. regress y _Iagegrp* _IageXwtg*
(output from regress appears)
```

## Warnings

1. xi creates new variables in your dataset; most are bytes, but interactions with continuous variables will have the storage type of the underlying continuous variable. You may get the message “insufficient memory”. If so, you will need to increase the amount of memory allocated to Stata’s data areas; see [U] [6 Setting the size of memory](#).
2. When using xi with an estimation command, you may get the message “matsize too small”. If so, see [R] [matsize](#).

## Saved results

xi saves the following characteristics:

```
_dta[___xi___Vars___Prefix___] prefix names
_dta[___xi___Vars___To___Drop___] variables created
```

## Methods and formulas

xi is implemented as an ado-file.

## References

- Hendrickx, J. 1999. [dm73: Using categorical variables in Stata](#). *Stata Technical Bulletin* 52: 2–8. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 51–59. College Station, TX: Stata Press.
- . 2000. [dm73.1: Contrasts for categorical variables: Update](#). *Stata Technical Bulletin* 54: 7. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 60–61. College Station, TX: Stata Press.
- . 2001a. [dm73.2: Contrasts for categorical variables: Update](#). *Stata Technical Bulletin* 59: 2–5. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 9–14. College Station, TX: Stata Press.
- . 2001b. [dm73.3: Contrasts for categorical variables: Update](#). *Stata Technical Bulletin* 61: 5. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 14–15. College Station, TX: Stata Press.

---

## Also see

[U] [11.1.10 Prefix commands](#)

[U] [20 Estimation and postestimation commands](#)

zinp — Zero-inflated negative binomial regression

Syntax

```
zinp depvar [indepvars] [if] [in] [weight] ,
 inflate(varlist [, offset(varname)] | _cons) [options]
```

| <i>options</i>                                  | description                                                                                                             |
|-------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| Model                                           |                                                                                                                         |
| * <u>inflate</u> ( )                            | equation that determines whether the count is zero                                                                      |
| <u>noconstant</u>                               | suppress constant term                                                                                                  |
| <u>exposure</u> ( <i>varname</i> <sub>e</sub> ) | include ln( <i>varname</i> <sub>e</sub> ) in model with coefficient constrained to 1                                    |
| <u>offset</u> ( <i>varname</i> <sub>o</sub> )   | include <i>varname</i> <sub>o</sub> in model with coefficient constrained to 1                                          |
| <u>constraints</u> ( <i>constraints</i> )       | apply specified linear constraints                                                                                      |
| <u>collinear</u>                                | keep collinear variables                                                                                                |
| probit                                          | use probit model to characterize excess zeros; default is logit                                                         |
| SE/Robust                                       |                                                                                                                         |
| <u>vce</u> ( <i>vcetype</i> )                   | <i>vcetype</i> may be oim, <u>robust</u> , <u>cluster</u> <i>clustvar</i> , opg, <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting                                       |                                                                                                                         |
| <u>level</u> (#)                                | set confidence level; default is level(95)                                                                              |
| irr                                             | report incidence-rate ratios                                                                                            |
| vuong                                           | perform Vuong test                                                                                                      |
| zip                                             | perform ZIP likelihood-ratio test                                                                                       |
| <u>nocnsreport</u>                              | do not display constraints                                                                                              |
| <u>display_options</u>                          | control spacing and display of omitted variables and base and empty cells                                               |
| Maximization                                    |                                                                                                                         |
| <u>maximize_options</u>                         | control maximization process; seldom used                                                                               |
| <sup>†</sup> <u>coeflegend</u>                  | display coefficients' legend instead of coefficient table                                                               |

\* inflate(*varlist* [ , offset(*varname*) ] | \_cons) is required.

<sup>†</sup> coeflegend does not appear in the dialog box.

*indepvars* and *varlist* may contain factor variables; see [U] 11.4.3 Factor variables.

bootstrap, by, jackknife, rolling, statsby, and svy are allowed; see [U] 11.1.10 Prefix commands.

Weights are not allowed with the bootstrap prefix.

vce( ), vuong, zip, and weights are not allowed with the svy prefix.

fweights, iweights, and pweights are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

## Menu

Statistics > Count outcomes > Zero-inflated negative binomial regression

## Description

`zinb` estimates a zero-inflated negative binomial (ZINB) regression of *depvar* on *indepvars*, where *depvar* is a nonnegative count variable.

## Options

### Model

`inflate(varlist [ , offset(varname)] | _cons)` specifies the equation that determines whether the observed count is zero. Conceptually, omitting `inflate()` would be equivalent to fitting the model with `nbreg`.

`inflate(varlist [ , offset(varname)])` specifies the variables in the equation. You may optionally include an offset for this *varlist*.

`inflate(_cons)` specifies that the equation determining whether the count is zero contains only an intercept. To run a zero-inflated model of *depvar* with only an intercept in both equations, type `zinb depvar, inflate(_cons)`.

`noconstant`, `exposure(varnamee)`, `offset(varnameo)`, `constraints(constraints)`, `collinear`; see [\[R\] estimation options](#).

`probit` requests that a probit, instead of logit, model be used to characterize the excess zeros in the data.

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [\[R\] vce\\_option](#).

### Reporting

`level(#)`; see [\[R\] estimation options](#).

`irr` reports estimated coefficients transformed to incidence-rate ratios, that is,  $e^{\beta_i}$  rather than  $\beta_i$ . Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated or stored. `irr` may be specified at estimation or when replaying previously estimated results.

`vuong` specifies that the [Vuong \(1989\)](#) test of ZINB versus negative binomial be reported. This test statistic has a standard normal distribution with large positive values favoring the ZINB model and large negative values favoring the negative binomial model.

`zip` requests that a likelihood-ratio test comparing the ZINB model with the zero-inflated Poisson model be included in the output.

`nocnsreport`; see [\[R\] estimation options](#).

*display\_options*: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [\[R\] estimation options](#).

## Maximization

*maximize\_options*: difficult, technique(*algorithm\_spec*), iterate(#), [no]log, trace, gradient, showstep, hessian, showtolerance, tolerance(#), ltolerance(#), nrtolerance(#), nonnrtolerance, from(*init\_specs*); see [R] [maximize](#). These options are seldom used.

Setting the optimization type to `technique(bhhh)` resets the default *vcetype* to `vce(opg)`.

The following option is available with `zinb` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

See [Long \(1997, 242–247\)](#) and [Greene \(2008, 922–924\)](#) for a discussion of zero-modified count models. For information about the test developed by [Vuong \(1989\)](#), see [Greene \(2008, 923–924\)](#) and [Long \(1997\)](#). [Greene \(1994\)](#) applied the test to zero-inflated Poisson and negative binomial models, and there is a description of that work in [Greene \(2008\)](#).

Negative binomial regression fits models of the number of occurrences (counts) of an event. You could use `nbreg` for this (see [R] [nbreg](#)), but in some count-data models, you might want to account for the prevalence of zero counts in the data.

For instance, you could count how many fish each visitor to a park catches. Many visitors may catch zero, because they do not fish (as opposed to being unsuccessful). You may be able to model whether a person fishes depending on several covariates related to fishing activity and model how many fish a person catches depending on several covariates having to do with the success of catching fish (type of lure/bait, time of day, temperature, season, etc.). This is the type of data for which the `zinb` command is useful.

The zero-inflated (or zero-altered) negative binomial model allows overdispersion through the splitting process that models the outcomes as zero or nonzero.

### ► Example 1

We have data on the number of fish caught by visitors to a national park. Some of the visitors do not fish, but we do not have the data on whether a person fished; we have data merely on how many fish were caught, together with several covariates. Because our data have a preponderance of zeros (142 of 250), we use the `zinb` command to model the outcome.

```
. use http://www.stata-press.com/data/r11/fish
. zinb count persons livebait, inf(child camper) vuong

Fitting constant-only model:
Iteration 0: log likelihood = -519.33992
 (output omitted)
Iteration 8: log likelihood = -442.66299

Fitting full model:
Iteration 0: log likelihood = -442.66299 (not concave)
 (output omitted)
Iteration 8: log likelihood = -401.54776
```



Zero-inflated negative binomial regression

Number of obs = 250

Nonzero obs = 108

Zero obs = 142

Inflation model = logit

LR chi2(2) = 82.23

Log likelihood = -401.5478

Prob > chi2 = 0.0000

|          | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| count    |           |           |       |       |                      |           |
| persons  | .9742984  | .1034938  | 9.41  | 0.000 | .7714543             | 1.177142  |
| livebait | 1.557523  | .4124424  | 3.78  | 0.000 | .7491503             | 2.365895  |
| _cons    | -2.730064 | .476953   | -5.72 | 0.000 | -3.664874            | -1.795253 |
| inflate  |           |           |       |       |                      |           |
| child    | 3.185999  | .7468551  | 4.27  | 0.000 | 1.72219              | 4.649808  |
| camper   | -2.020951 | .872054   | -2.32 | 0.020 | -3.730146            | -.3117567 |
| _cons    | -2.695385 | .8929071  | -3.02 | 0.003 | -4.44545             | -.9453189 |
| /lnalpha | .5110429  | .1816816  | 2.81  | 0.005 | .1549535             | .8671323  |
| alpha    | 1.667029  | .3028685  |       |       | 1.167604             | 2.380076  |

Vuong test of zinvb vs. standard negative binomial: z = 5.59 Pr>z = 0.0000

In general, Vuong test statistics that are significantly positive favor the zero-inflated models, whereas those that are significantly negative favor the non-zero-inflated models. Thus, in the above model, the zero inflation is significant.

Saved results

zinvb saves the following in e():

- Scalars
- e(N)

number of observations
- e(N\_zero)

number of zero observations
- e(k)

number of parameters
- e(k\_eq)

number of equations
- e(k\_eq\_model)

number of equations in model Wald test
- e(k\_aux)

number of auxiliary parameters
- e(k\_dv)

number of dependent variables
- e(k\_autoCns)

number of base, empty, and omitted constraints
- e(df\_m)

model degrees of freedom
- e(ll)

log likelihood
- e(ll\_0)

log likelihood, constant-only model
- e(df\_c)

degrees of freedom for comparison test
- e(N\_clust)

number of clusters
- e(chi2)

$\chi^2$
- e(p)

significance of model test
- e(chi2\_cp)

$\chi^2$  for test of  $\alpha = 0$
- e(vuong)

Vuong test statistic
- e(rank)

rank of e(V)
- e(ic)

number of iterations
- e(rc)

return code
- e(converged)

1 if converged, 0 otherwise

## Macros

|                                 |                                                                                  |
|---------------------------------|----------------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>zinb</code>                                                                |
| <code>e(cmdline)</code>         | command as typed                                                                 |
| <code>e(depvar)</code>          | name of dependent variable                                                       |
| <code>e(inflate)</code>         | logit or probit                                                                  |
| <code>e(wtype)</code>           | weight type                                                                      |
| <code>e(wexp)</code>            | weight expression                                                                |
| <code>e(title)</code>           | title in estimation output                                                       |
| <code>e(clustvar)</code>        | name of cluster variable                                                         |
| <code>e(offset1)</code>         | offset                                                                           |
| <code>e(offset2)</code>         | offset for <code>inflate()</code>                                                |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                          |
| <code>e(chi2_cpt)</code>        | Wald or LR; type of model $\chi^2$ test corresponding to <code>e(chi2_cp)</code> |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                                   |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                                    |
| <code>e(diparm#)</code>         | display transformed parameter #                                                  |
| <code>e(opt)</code>             | type of optimization                                                             |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization         |
| <code>e(ml_method)</code>       | type of ml method                                                                |
| <code>e(user)</code>            | name of likelihood-evaluator program                                             |
| <code>e(technique)</code>       | maximization technique                                                           |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular                      |
| <code>e(crittype)</code>        | optimization criterion                                                           |
| <code>e(properties)</code>      | b V                                                                              |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                                   |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>                   |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>                   |

## Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(gradient)</code>     | gradient vector                              |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`zinb` is implemented as an ado-file.

Several models in the literature are (correctly) described as zero inflated. The `zinb` command maximizes the log likelihood  $\ln L$ , defined by

$$\begin{aligned}
m &= 1/\alpha \\
p_j &= 1/(1 + \alpha\mu_j) \\
\xi_j^\beta &= \mathbf{x}_j\beta + \text{offset}_j^\beta \\
\xi_j^\gamma &= \mathbf{z}_j\gamma + \text{offset}_j^\gamma \\
\mu_j &= \exp(\xi_j^\beta) \\
\ln L &= \sum_{j \in S} w_j \ln [F(\xi_j^\gamma) + \{1 - F(\xi_j^\gamma)\} p_j^m] \\
&\quad + \sum_{j \notin S} w_j \left[ \ln \{1 - F(\xi_j^\gamma)\} + \ln \Gamma(m + y_j) - \ln \Gamma(y_j + 1) \right. \\
&\quad \left. - \ln \Gamma(m) + m \ln p_j + y_j \ln(1 - p_j) \right]
\end{aligned}$$

where  $w_j$  are the weights,  $F$  is the logit link (or probit link if `probit` was specified), and  $S$  is the set of observations for which the outcome  $y_j = 0$ .

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

`zlnb` also supports estimation with survey data. For details on VCEs with survey data, see [SVY] [variance estimation](#).

## References

- Greene, W. H. 1994. Accounting for excess zeros and sample selection in Poisson and negative binomial regression models. Working paper EC-94-10, Department of Economics, Stern School of Business, New York University. <http://ideas.repec.org/p/ste/nystbu/94-10.html>.
- . 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice-Hall.
- Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.
- Long, J. S., and J. Freese. 2001. Predicted probabilities for count models. *Stata Journal* 1: 51–57.
- . 2006. *Regression Models for Categorical Dependent Variables Using Stata*. 2nd ed. College Station, TX: Stata Press.
- Mullahy, J. 1986. Specification and testing of some modified count data models. *Journal of Econometrics* 33: 341–365.
- Vuong, Q. H. 1989. Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica* 57: 307–333.

## Also see

- [R] [zlnb postestimation](#) — Postestimation tools for `zlnb`
- [R] [zip](#) — Zero-inflated Poisson regression
- [R] [ztnb](#) — Zero-truncated negative binomial regression
- [R] [nbreg](#) — Negative binomial regression
- [R] [poisson](#) — Poisson regression
- [XT] [xtnbreg](#) — Fixed-effects, random-effects, & population-averaged negative binomial models
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `zinb`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic nooffset]

predict [type] { stub* | newvarreg newvarinflate newvarlnalpha } [if] [in] , scores
```

| statistic         | description                             |
|-------------------|-----------------------------------------|
| Main              |                                         |
| <code>n</code>    | number of events; the default           |
| <code>ir</code>   | incidence rate                          |
| <code>pr</code>   | probability of a zero outcome           |
| <code>xb</code>   | linear prediction                       |
| <code>stdp</code> | standard error of the linear prediction |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

**n**, the default, calculates the predicted number of events, which is  $(1 - p_j) \exp(\mathbf{x}_j \beta)$  if neither `offset()` nor `exposure()` was specified when the model was fit, where  $p_j$  is the predicted probability of a zero outcome;  $(1 - p_j) \exp((\mathbf{x}_j \beta) + \text{offset}_j)$  if `offset()` was specified; or  $(1 - p_j)(\exp(\mathbf{x}_j \beta) \times \text{exposure}_j)$  if `exposure()` was specified.

**ir** calculates the incidence rate  $\exp(\mathbf{x}_j \beta)$ , which is the predicted number of events when exposure is 1. This is equivalent to specifying both **n** and **nooffset** options.

**pr** calculates the probability of a zero outcome  $F(\mathbf{z}_j \gamma)$ , where  $F()$  is the logit or probit link function. If `offset()` was specified within the `inflate()` option, then  $F(\mathbf{z}_j \gamma + \text{offset}_j^\gamma)$  is calculated.

**xb** calculates the linear prediction, which is  $\mathbf{x}_j \beta$  if neither `offset()` nor `exposure()` was specified;  $\mathbf{x}_j \beta + \text{offset}_j$  if `offset()` was specified; or  $\mathbf{x}_j \beta + \ln(\text{exposure}_j)$  if `exposure()` was specified; see **nooffset** below.

**stdp** calculates the standard error of the linear prediction.

**nooffset** is relevant only if you specified `offset()` or `exposure()` when you fit the model. It modifies the calculations made by **predict** so that they ignore the offset or exposure variable; the linear prediction is treated as  $\mathbf{x}_j \beta$  rather than as  $\mathbf{x}_j \beta + \text{offset}_j$  or  $\mathbf{x}_j \beta + \ln(\text{exposure}_j)$ . Specifying `predict ... , nooffset` is equivalent to specifying `predict ... , ir`.

**scores** calculates equation-level score variables.

The first new variable will contain  $\partial \ln L / \partial (\mathbf{x}_j \beta)$ .

The second new variable will contain  $\partial \ln L / \partial (\mathbf{z}_j \gamma)$ .

The third new variable will contain  $\partial \ln L / \partial \ln \alpha$ .

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] **zinb** — Zero-inflated negative binomial regression

[U] **20 Estimation and postestimation commands**

Syntax

```
zip depvar [indepvars] [if] [in] [weight] ,
 inflate(varlist [, offset(varname)] | _cons) [options]
```

| <i>options</i>                                  | description                                                                                                             |
|-------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| Model                                           |                                                                                                                         |
| * <u>inflate</u> ( )                            | equation that determines whether the count is zero                                                                      |
| <u>noconstant</u>                               | suppress constant term                                                                                                  |
| <u>exposure</u> ( <i>varname</i> <sub>e</sub> ) | include ln( <i>varname</i> <sub>e</sub> ) in model with coefficient constrained to 1                                    |
| <u>offset</u> ( <i>varname</i> <sub>o</sub> )   | include <i>varname</i> <sub>o</sub> in model with coefficient constrained to 1                                          |
| <u>constraints</u> ( <i>constraints</i> )       | apply specified linear constraints                                                                                      |
| <u>collinear</u>                                | keep collinear variables                                                                                                |
| <u>probit</u>                                   | use probit model to characterize excess zeros; default is logit                                                         |
| SE/Robust                                       |                                                                                                                         |
| <u>vce</u> ( <i>vcetype</i> )                   | <i>vcetype</i> may be oim, <u>robust</u> , <u>cluster</u> <i>clustvar</i> , opg, <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting                                       |                                                                                                                         |
| <u>level</u> (#)                                | set confidence level; default is level(95)                                                                              |
| <u>irr</u>                                      | report incidence-rate ratios                                                                                            |
| <u>vuong</u>                                    | perform Vuong test                                                                                                      |
| <u>nocnsreport</u>                              | do not display constraints                                                                                              |
| <u>display_options</u>                          | control spacing and display of omitted variables and base and empty cells                                               |
| Maximization                                    |                                                                                                                         |
| <u>maximize_options</u>                         | control maximization process; seldom used                                                                               |
| <sup>†</sup> <u>coeflegend</u>                  | display coefficients' legend instead of coefficient table                                                               |

\* inflate(*varlist* [ , offset(*varname*) ] | \_cons) is required.

<sup>†</sup> coeflegend does not appear in the dialog box.

*indepvars* and *varlist* may contain factor variables; see [U] 11.4.3 Factor variables.

bootstrap, by, jackknife, rolling, statsby, and svy are allowed; see [U] 11.1.10 Prefix commands.

Weights are not allowed with the bootstrap prefix.

vce( ), vuong, and weights are not allowed with the svy prefix.

fweights, iweights, and pweights are allowed; see [U] 11.1.6 weight.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

## Menu

Statistics > Count outcomes > Zero-inflated Poisson regression

## Description

zip estimates a zero-inflated Poisson (ZIP) regression of *depvar* on *indepvars*, where *depvar* is a nonnegative count variable.

## Options

### Model

`inflate(varlist[, offset(varname)] | _cons)` specifies the equation that determines whether the observed count is zero. Conceptually, omitting `inflate()` would be equivalent to fitting the model with `poisson`.

`inflate(varlist[, offset(varname)])` specifies the variables in the equation. You may optionally include an offset for this *varlist*.

`inflate(_cons)` specifies that the equation determining whether the count is zero contains only an intercept. To run a zero-inflated model of *depvar* with only an intercept in both equations, type `zip depvar, inflate(_cons)`.

`noconstant`, `exposure(varnamee)`, `offset(varnameo)`, `constraints(constraints)`, `collinear`; see [\[R\] estimation options](#).

`probit` requests that a probit, instead of logit, model be used to characterize the excess zeros in the data.

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [\[R\] vce\\_option](#).

### Reporting

`level(#)`; see [\[R\] estimation options](#).

`irr` reports estimated coefficients transformed to incidence-rate ratios, that is,  $e^b$  rather than  $b$ . Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated or stored. `irr` may be specified at estimation or when replaying previously estimated results.

`vuong` specifies that the [Vuong \(1989\)](#) test of ZIP versus Poisson be reported. This test statistic has a standard normal distribution with large positive values favoring the ZIP model and large negative values favoring the Poisson model.

`nocnsreport`; see [\[R\] estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [\[R\] estimation options](#).

*maximize\_options*: difficult, technique(*algorithm\_spec*), iterate(#), [no]log, trace, gradient, showstep, hessian, showtolerance, tolerance(#), ltolerance(#), nrtolerance(#), nonrtolerance, from(*init\_specs*); see [R] [maximize](#). These options are seldom used.

Setting the optimization type to `technique(bhhh)` resets the default *vcetype* to `vce(opg)`.

The following option is available with `zip` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

## Remarks

See [Long \(1997, 242–247\)](#) and [Greene \(2008, 922–924\)](#) for a discussion of zero-modified count models. For information about the test developed by [Vuong \(1989\)](#), see [Greene \(2008\)](#) and [Long \(1997\)](#). [Greene \(1994\)](#) applied the test to ZIP and ZINB models, as described in [Greene \(2008, 923–924\)](#).

Poisson regression fits models of the number of occurrences (counts) of an event. You could use `poisson` for this (see [R] [poisson](#)), but in some count-data models, you might want to account for the prevalence of zero counts in the data.

For instance, you might count how many fish each visitor to a park catches. Many visitors may catch zero, because they do not fish (as opposed to being unsuccessful). You may be able to model whether a person fishes depending on several covariates related to fishing activity and model how many fish a person catches depending on several covariates having to do with the success of catching fish (type of lure/bait, time of day, temperature, season, etc.). This is the type of data for which the `zip` command is useful.

The zero-inflated (or zero-altered) Poisson model allows overdispersion through the splitting process that models the outcomes as zero or nonzero.

## ► Example 1

We have data on the number of fish caught by visitors to a national park. Some of the visitors do not fish, but we do not have the data on whether a person fished; we merely have data on how many fish were caught together with several covariates. Because our data have a preponderance of zeros (142 of 250), we use the `zip` command to model the outcome.

```
. use http://www.stata-press.com/data/r11/fish
. zip count persons livebait, inf(child camper) vuong
Fitting constant-only model:
Iteration 0: log likelihood = -1347.807
Iteration 1: log likelihood = -1305.3245
 (output omitted)
Iteration 4: log likelihood = -1103.9425
Fitting full model:
Iteration 0: log likelihood = -1103.9425
Iteration 1: log likelihood = -896.2346
 (output omitted)
Iteration 5: log likelihood = -850.70142
```



|                                  |               |   |        |
|----------------------------------|---------------|---|--------|
| Zero-inflated Poisson regression | Number of obs | = | 250    |
|                                  | Nonzero obs   | = | 108    |
|                                  | Zero obs      | = | 142    |
| Inflation model = logit          | LR chi2(2)    | = | 506.48 |
| Log likelihood = -850.7014       | Prob > chi2   | = | 0.0000 |

|          | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|-----------|-------|-------|----------------------|-----------|
| count    |           |           |       |       |                      |           |
| persons  | .8068853  | .0453288  | 17.80 | 0.000 | .7180424             | .8957281  |
| livebait | 1.757289  | .2446082  | 7.18  | 0.000 | 1.277866             | 2.236713  |
| _cons    | -2.178472 | .2860289  | -7.62 | 0.000 | -2.739078            | -1.617865 |
| inflate  |           |           |       |       |                      |           |
| child    | 1.602571  | .2797719  | 5.73  | 0.000 | 1.054228             | 2.150913  |
| camper   | -1.015698 | .365259   | -2.78 | 0.005 | -1.731593            | -.2998038 |
| _cons    | -.4922872 | .3114562  | -1.58 | 0.114 | -1.10273             | .1181558  |

Vuong test of zip vs. standard Poisson:                      z =        3.95    Pr>z = 0.0000

In general, Vuong test statistics that are significantly positive favor the zero-inflated models, while those that are significantly negative favor the non-zero-inflated models. Thus, in the above model, the zero inflation is significant.



## Saved results

zip saves the following in e():

### Scalars

|               |                                                |
|---------------|------------------------------------------------|
| e(N)          | number of observations                         |
| e(N_zero)     | number of zero observations                    |
| e(k)          | number of parameters                           |
| e(k_eq)       | number of equations                            |
| e(k_eq_model) | number of equations in model Wald test         |
| e(k_dv)       | number of dependent variables                  |
| e(k_autoCns)  | number of base, empty, and omitted constraints |
| e(df_m)       | model degrees of freedom                       |
| e(ll)         | log likelihood                                 |
| e(ll_0)       | log likelihood, constant-only model            |
| e(ll_c)       | log likelihood, comparison model               |
| e(df_c)       | degrees of freedom for comparison test         |
| e(N_clust)    | number of clusters                             |
| e(chi2)       | $\chi^2$                                       |
| e(p)          | significance of model test                     |
| e(vuong)      | Vuong test statistic                           |
| e(rank)       | rank of e(V)                                   |
| e(ic)         | number of iterations                           |
| e(rc)         | return code                                    |
| e(converged)  | 1 if converged, 0 otherwise                    |

## Macros

|                                 |                                                                          |
|---------------------------------|--------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>zip</code>                                                         |
| <code>e(cmdline)</code>         | command as typed                                                         |
| <code>e(depvar)</code>          | name of dependent variable                                               |
| <code>e(inflate)</code>         | logit or probit                                                          |
| <code>e(wtype)</code>           | weight type                                                              |
| <code>e(wexp)</code>            | weight expression                                                        |
| <code>e(title)</code>           | title in estimation output                                               |
| <code>e(clustvar)</code>        | name of cluster variable                                                 |
| <code>e(offset1)</code>         | offset                                                                   |
| <code>e(offset2)</code>         | offset for <code>inflate()</code>                                        |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                  |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                           |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                            |
| <code>e(opt)</code>             | type of optimization                                                     |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>       | type of ml method                                                        |
| <code>e(user)</code>            | name of likelihood-evaluator program                                     |
| <code>e(technique)</code>       | maximization technique                                                   |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular              |
| <code>e(crittype)</code>        | optimization criterion                                                   |
| <code>e(properties)</code>      | b V                                                                      |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                           |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>           |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>           |

## Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(gradient)</code>     | gradient vector                              |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`zip` is implemented as an ado-file.

Several models in the literature are (correctly) described as zero inflated. The `zip` command maximizes the log-likelihood  $\ln L$ , defined by

$$\begin{aligned}
 \xi_j^\beta &= \mathbf{x}_j \boldsymbol{\beta} + \text{offset}_j^\beta \\
 \xi_j^\gamma &= \mathbf{z}_j \boldsymbol{\gamma} + \text{offset}_j^\gamma \\
 \ln L &= \sum_{i \in S} w_j \ln [F(\xi_j^\gamma) + \{1 - F(\xi_j^\gamma)\} \exp(-\lambda_j)] + \\
 &\quad \sum_{i \notin S} w_j [\ln \{1 - F(\xi_j^\gamma)\} - \lambda_j + \xi_j^\beta y_j - \ln(y_j!)]
 \end{aligned}$$

where  $w_j$  are the weights,  $F$  is the logit link (or probit link if `probit` was specified), and  $S$  is the set of observations for which the outcome  $y_j = 0$ .

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

`zip` also supports estimation with survey data. For details on VCEs with survey data, see [SVY] [variance estimation](#).

## References

- Greene, W. H. 1994. Accounting for excess zeros and sample selection in Poisson and negative binomial regression models. Working paper EC-94-10, Department of Economics, Stern School of Business, New York University. <http://ideas.repec.org/p/ste/nystbu/94-10.html>.
- . 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice-Hall.
- Lambert, D. 1992. Zero-inflated Poisson regression, with an application to defects in manufacturing. *Technometrics* 34: 1–14.
- Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.
- Long, J. S., and J. Freese. 2001. Predicted probabilities for count models. *Stata Journal* 1: 51–57.
- . 2006. *Regression Models for Categorical Dependent Variables Using Stata*. 2nd ed. College Station, TX: Stata Press.
- Mullahy, J. 1986. Specification and testing of some modified count data models. *Journal of Econometrics* 33: 341–365.
- Vuong, Q. H. 1989. Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica* 57: 307–333.

## Also see

- [R] [zip postestimation](#) — Postestimation tools for `zip`
- [R] [zinb](#) — Zero-inflated negative binomial regression
- [R] [ztnb](#) — Zero-truncated negative binomial regression
- [R] [ztp](#) — Zero-truncated Poisson regression
- [R] [nbreg](#) — Negative binomial regression
- [R] [poisson](#) — Poisson regression
- [XT] [xtnbreg](#) — Fixed-effects, random-effects, & population-averaged negative binomial models
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `zip`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic noffset]

predict [type] { stub* | newvarreg newvarinflate } [if] [in] , scores
```

| statistic         | description                             |
|-------------------|-----------------------------------------|
| Main              |                                         |
| <code>n</code>    | number of events; the default           |
| <code>ir</code>   | incidence rate                          |
| <code>pr</code>   | probability of a zero outcome           |
| <code>xb</code>   | linear prediction                       |
| <code>stdp</code> | standard error of the linear prediction |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

**n**, the default, calculates the predicted number of events, which is  $(1 - p_j) \exp(\mathbf{x}_j\beta)$  if neither `offset()` nor `exposure()` was specified when the model was fit, where  $p_j$  is the predicted probability of a zero outcome;  $(1 - p_j) \exp((\mathbf{x}_j\beta) + \text{offset}_j)$  if `offset()` was specified; or  $(1 - p_j)(\exp(\mathbf{x}_j\beta) \times \text{exposure}_j)$  if `exposure()` was specified.

**ir** calculates the incidence rate  $\exp(\mathbf{x}_j\beta)$ , which is the predicted number of events when exposure is 1. This is equivalent to specifying both **n** and **nooffset** options.

**pr** calculates the probability of a zero outcome  $F(\mathbf{z}_j\gamma)$ , where  $F()$  is the logit or probit link function. If `offset()` was specified within the `inflate()` option, then  $F(\mathbf{z}_j\gamma + \text{offset}_j^\gamma)$  is calculated.

**xb** calculates the linear prediction, which is  $\mathbf{x}_j\beta$  if neither `offset()` nor `exposure()` was specified;  $\mathbf{x}_j\beta + \text{offset}_j$  if `offset()` was specified; or  $\mathbf{x}_j\beta + \ln(\text{exposure}_j)$  if `exposure()` was specified; see **nooffset** below.

**stdp** calculates the standard error of the linear prediction.

**nooffset** is relevant only if you specified `offset()` or `exposure()` when you fit the model. It modifies the calculations made by **predict** so that they ignore the offset or exposure variable; the linear prediction is treated as  $\mathbf{x}_j\beta$  rather than as  $\mathbf{x}_j\beta + \text{offset}_j$  or  $\mathbf{x}_j\beta + \ln(\text{exposure}_j)$ . Specifying `predict ... , nooffset` is equivalent to specifying `predict ... , ir`.

**scores** calculates equation-level score variables.

The first new variable will contain  $\partial \ln L / \partial (\mathbf{x}_j\beta)$ .

The second new variable will contain  $\partial \ln L / \partial (\mathbf{z}_j\gamma)$ .

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

## Also see

[R] [zip](#) — Zero-inflated Poisson regression

[U] [20 Estimation and postestimation commands](#)

Syntax

ztnb *depvar* [*indepvars*] [*if*] [*in*] [*weight*] [, *options*]

| <i>options</i>                                  | description                                                                                                             |
|-------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| Model                                           |                                                                                                                         |
| <u>no</u> constant                              | suppress constant term                                                                                                  |
| <u>dispersion</u> ( <u>mean</u> )               | parameterization of dispersion; <u>dispersion</u> ( <u>mean</u> ) is the default                                        |
| <u>dispersion</u> ( <u>constant</u> )           | constant dispersion for all observations                                                                                |
| <u>exposure</u> ( <i>varname</i> <sub>e</sub> ) | include ln( <i>varname</i> <sub>e</sub> ) in model with coefficient constrained to 1                                    |
| <u>offset</u> ( <i>varname</i> <sub>o</sub> )   | include <i>varname</i> <sub>o</sub> in model with coefficient constrained to 1                                          |
| <u>constraints</u> ( <i>constraints</i> )       | apply specified linear constraints                                                                                      |
| <u>collinear</u>                                | keep collinear variables                                                                                                |
| SE/Robust                                       |                                                                                                                         |
| <u>vce</u> ( <i>vcetype</i> )                   | <i>vcetype</i> may be oim, <u>robust</u> , <u>cluster</u> <i>clustvar</i> , opg, <u>bootstrap</u> , or <u>jackknife</u> |
| Reporting                                       |                                                                                                                         |
| <u>level</u> (#)                                | set confidence level; default is level(95)                                                                              |
| <u>nolrtest</u>                                 | suppress likelihood-ratio test                                                                                          |
| <u>irr</u>                                      | report incidence-rate ratios                                                                                            |
| <u>nocnsreport</u>                              | do not display constraints                                                                                              |
| <i>display_options</i>                          | control spacing and display of omitted variables and base and empty cells                                               |
| Maximization                                    |                                                                                                                         |
| <i>maximize_options</i>                         | control the maximization process; seldom used                                                                           |
| <sup>†</sup> <u>coeflegend</u>                  | display coefficients' legend instead of coefficient table                                                               |

<sup>†</sup>coeflegend does not appear in the dialog box.  
*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.  
*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.  
bootstrap, by, jackknife, rolling, statsby, and svy are allowed; see [U] 11.1.10 Prefix commands.  
Weights are not allowed with the bootstrap prefix.  
vce() and weights are not allowed with the svy prefix.  
fweights, iweights, and pweights are allowed; see [U] 11.1.6 weight.  
See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Count outcomes > Zero-truncated negative binomial regression

## Description

`ztnb` fits a zero-truncated negative binomial (ZTNB) regression model of *depvar* on *indepvars*, where *depvar* is a positive count variable.

## Options

### Model

`noconstant`; see [R] [estimation options](#).

`dispersion(mean | constant)` specifies the parameterization of the model. `dispersion(mean)`, the default, yields a model with dispersion equal to  $1 + \alpha \exp(\mathbf{x}_j\beta + \text{offset}_j)$ ; that is, the dispersion is a function of the expected mean:  $\exp(\mathbf{x}_j\beta + \text{offset}_j)$ . `dispersion(constant)` has dispersion equal to  $1 + \delta$ ; that is, it is a constant for all observations.

`exposure(varnamee)`, `offset(varnameo)`, `constraints(constraints)`, `collinear`; see [R] [estimation options](#).

### SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

### Reporting

`level(#)`; see [R] [estimation options](#).

`nolrtest` suppresses fitting the Poisson model. Without this option, a comparison Poisson model is fit, and the likelihood is used in a likelihood-ratio test of the null hypothesis that the dispersion parameter is zero.

`irr` reports estimated coefficients transformed to incidence-rate ratios, i.e.,  $e^b$  rather than  $b$ . Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated or stored. You can specify `irr` at estimation or when you replay previously estimated results.

`nocnsreport`; see [R] [estimation options](#).

`display_options`: `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`; see [R] [estimation options](#).

### Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `from(init_specs)`; see [R] [maximize](#). These options are seldom used.

Setting the optimization type to `technique(bhhh)` resets the default `vcetype` to `vce(opg)`.

The following option is available with `ztnb` but is not shown in the dialog box:

`coeflegend`; see [R] [estimation options](#).

Remarks

Grogger and Carson (1991) showed that overdispersion causes inconsistent estimation of the mean in the truncated Poisson model, so they proposed the ZTNB model as an alternative. If the data exhibit zero truncation but not overdispersion, the zero-truncated Poisson model is appropriate; see [R] ztp. See Long (1997, chap. 8) for an introduction to negative binomial and ZTNB regression.

ztnb fits two different parameterizations of ZTNB models, namely, the mean-dispersion and constant-dispersion models. They are equivalent to those modeled by nbreg; see [R] nbreg.

► Example 1

We illustrate the ZTNB model by using the HCFA’s 1997 MedPar dataset (Hilbe 1999). The data represent 475 patients in Arizona who were assigned a diagnostic-related group (DRG) of patients having a ventilator. Length of stay, the dependent variable, is a positive integer; it cannot have zero values. The variables used in the study include

|         |                                        |
|---------|----------------------------------------|
| provnum | provider number (Hospital identifier)  |
| died    | patient died while in hospital         |
| hmo     | patient is a member of an HMO          |
| los     | length of stay                         |
| type1   | emergency admission                    |
| type2   | urgent admission (first available bed) |
| type3   | elective admission                     |

```
. use http://www.stata-press.com/data/r11/medpar
. ztnb los died hmo type2-type3, vce(cluster provnum) nolog
Zero-truncated negative binomial regression Number of obs = 1495
Dispersion = mean Wald chi2(4) = 36.01
Log likelihood = -4737.535 Prob > chi2 = 0.0000
 (Std. Err. adjusted for 54 clusters in provnum)
```

| los      | Coef.     | Robust Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|-----------|------------------|-------|-------|----------------------|-----------|
| died     | -.2521884 | .061533          | -4.10 | 0.000 | -.3727908            | -.1315859 |
| hmo      | -.0754173 | .0533132         | -1.41 | 0.157 | -.1799091            | .0290746  |
| type2    | .2685095  | .0666474         | 4.03  | 0.000 | .137883              | .3991359  |
| type3    | .7668101  | .2183505         | 3.51  | 0.000 | .338851              | 1.194769  |
| _cons    | 2.224028  | .034727          | 64.04 | 0.000 | 2.155964             | 2.292091  |
| /lnalpha | -.630108  | .0764019         |       |       | -.779853             | -.480363  |
| alpha    | .5325343  | .0406866         |       |       | .4584734             | .6185588  |

Because observations within the same hospital (provnum) are likely to be correlated, we specified the vce(cluster provnum) option. The results show that whether the patient died in the hospital and the type of admission have significant effects on the patient’s length of stay.



➤ Example 2

Here we compare the zero-truncated Poisson model with the ZTNB model. The following data appeared in [Rodríguez \(1993\)](#).

```
. use http://www.stata-press.com/data/r11/rod93
. ztp deaths i.cohort, exp(exposure) nolog
Zero-truncated Poisson regression
Log likelihood = -2159.5098
Number of obs = 21
LR chi2(2) = 49.16
Prob > chi2 = 0.0000
Pseudo R2 = 0.0113
```

| deaths   | Coef.      | Std. Err. | z      | P> z  | [95% Conf. Interval] |           |
|----------|------------|-----------|--------|-------|----------------------|-----------|
| cohort   |            |           |        |       |                      |           |
| 2        | -.3020296  | .0573367  | -5.27  | 0.000 | -.4144076            | -.1896517 |
| 3        | .074248    | .0589754  | 1.26   | 0.208 | -.0413417            | .1898377  |
| _cons    | -3.899522  | .0411385  | -94.79 | 0.000 | -3.980152            | -3.818892 |
| exposure | (exposure) |           |        |       |                      |           |

```
. ztnb deaths i.cohort, exp(exposure) nolog
Zero-truncated negative binomial regression
Dispersion = mean
Log likelihood = -130.05845
Number of obs = 21
LR chi2(2) = 0.32
Prob > chi2 = 0.8535
Pseudo R2 = 0.0012
```

| deaths   | Coef.      | Std. Err. | z     | P> z  | [95% Conf. Interval] |           |
|----------|------------|-----------|-------|-------|----------------------|-----------|
| cohort   |            |           |       |       |                      |           |
| 2        | -.2628645  | .817489   | -0.32 | 0.748 | -1.865114            | 1.339384  |
| 3        | -.4587405  | .8167563  | -0.56 | 0.574 | -2.059553            | 1.142072  |
| _cons    | -2.181593  | .5831368  | -3.74 | 0.000 | -3.32452             | -1.038666 |
| exposure | (exposure) |           |       |       |                      |           |
| /lnalpha | .8916014   | .4062167  |       |       | .0954313             | 1.687772  |
| alpha    | 2.439032   | .9907757  |       |       | 1.100133             | 5.407417  |

Likelihood-ratio test of alpha=0:  chibar2(01) = 4058.90 Prob>=chibar2 = 0.000

Comparing the zero-truncated Poisson and ZTNB models, we see that `ztnb` has a much larger log likelihood. To test  $\alpha = 0$ , `ztnb` performs a likelihood-ratio test. The large  $\chi^2$  value of 4058.90 strongly suggests that the ZTNB model is more appropriate.



□ Technical note

The usual Gaussian test of  $\alpha = 0$  is omitted because this test occurs on the boundary, invalidating the usual theory associated with such tests. However, the likelihood-ratio test of  $\alpha = 0$  has been modified to be valid on the boundary. In particular, the null distribution of the likelihood-ratio test statistic is not the usual  $\chi^2_1$ , but rather a 50:50 mixture of a  $\chi^2_0$  (point mass at zero) and a  $\chi^2_1$ , denoted as  $\bar{\chi}^2_{01}$ . See [Gutierrez, Carter, and Drukker \(2001\)](#) for more details.



## Saved results

ztnb saves the following in `e()`:

|                            |                                                   |
|----------------------------|---------------------------------------------------|
| Scalars                    |                                                   |
| <code>e(N)</code>          | number of observations                            |
| <code>e(k)</code>          | number of parameters                              |
| <code>e(k_eq)</code>       | number of equations                               |
| <code>e(k_eq_model)</code> | number of equations in model Wald test            |
| <code>e(k_aux)</code>      | number of auxiliary parameters                    |
| <code>e(k_dv)</code>       | number of dependent variables                     |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints    |
| <code>e(df_m)</code>       | model degrees of freedom                          |
| <code>e(r2_p)</code>       | pseudo- <i>R</i> -squared                         |
| <code>e(l1)</code>         | log likelihood                                    |
| <code>e(l1_0)</code>       | log likelihood, constant-only model               |
| <code>e(l1_c)</code>       | log likelihood, comparison model                  |
| <code>e(alpha)</code>      | the value of alpha                                |
| <code>e(N_clust)</code>    | number of clusters                                |
| <code>e(chi2)</code>       | $\chi^2$                                          |
| <code>e(chi2_c)</code>     | $\chi^2$ for comparison test                      |
| <code>e(p)</code>          | significance                                      |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                         |
| <code>e(rank0)</code>      | rank of <code>e(V)</code> for constant-only model |
| <code>e(ic)</code>         | number of iterations                              |
| <code>e(rc)</code>         | return code                                       |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                       |

## Macros

|                                 |                                                                                 |
|---------------------------------|---------------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <code>ztnb</code>                                                               |
| <code>e(cmdline)</code>         | command as typed                                                                |
| <code>e(depvar)</code>          | name of dependent variable                                                      |
| <code>e(wtype)</code>           | weight type                                                                     |
| <code>e(wexp)</code>            | weight expression                                                               |
| <code>e(title)</code>           | title in estimation output                                                      |
| <code>e(clustvar)</code>        | name of cluster variable                                                        |
| <code>e(offset)</code>          | offset                                                                          |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                         |
| <code>e(dispers)</code>         | mean or constant                                                                |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                                  |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                                   |
| <code>e(chi2_ct)</code>         | Wald or LR; type of model $\chi^2$ test corresponding to <code>e(chi2_c)</code> |
| <code>e(diparm#)</code>         | display transformed parameter #                                                 |
| <code>e(diparm_opt2)</code>     | options for displaying transformed parameters                                   |
| <code>e(opt)</code>             | type of optimization                                                            |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization        |
| <code>e(ml_method)</code>       | type of ml method                                                               |
| <code>e(user)</code>            | name of likelihood-evaluator program                                            |
| <code>e(technique)</code>       | maximization technique                                                          |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular                     |
| <code>e(crittype)</code>        | optimization criterion                                                          |
| <code>e(properties)</code>      | b V                                                                             |
| <code>e(predict)</code>         | program used to implement predict                                               |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>                  |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>                  |

## Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(gradient)</code>     | gradient vector                              |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`ztnb` is implemented as an ado-file.

Methods and formulas are presented under the following headings:

*Mean-dispersion model*

*Constant-dispersion model*

## Mean-dispersion model

A negative binomial distribution can be regarded as a gamma mixture of Poisson random variables. The number of times something occurs,  $y_j$ , is distributed as  $\text{Poisson}(\nu_j \mu_j)$ . That is, its conditional likelihood is

$$f(y_j | \nu_j) = \frac{(\nu_j \mu_j)^{y_j} e^{-\nu_j \mu_j}}{\Gamma(y_j + 1)}$$

where  $\mu_j = \exp(\mathbf{x}_j \boldsymbol{\beta} + \text{offset}_j)$  and  $\nu_j$  is an unobserved parameter with a  $\text{gamma}(1/\alpha, \alpha)$  density:

$$g(\nu) = \frac{\nu^{(1-\alpha)/\alpha} e^{-\nu/\alpha}}{\alpha^{1/\alpha} \Gamma(1/\alpha)}$$

This gamma distribution has mean 1 and variance  $\alpha$ , where  $\alpha$  is the ancillary parameter.

The unconditional likelihood for the  $j$ th observation is therefore

$$f(y_j) = \int_0^\infty f(y_j | \nu) g(\nu) d\nu = \frac{\Gamma(m + y_j)}{\Gamma(y_j + 1) \Gamma(m)} p_j^m (1 - p_j)^{y_j}$$

where  $p_j = 1/(1 + \alpha \mu_j)$  and  $m = 1/\alpha$ . Solutions for  $\alpha$  are handled by searching for  $\ln \alpha$  because  $\alpha$  must be greater than zero. The conditional probability of observing  $y_j$  events given that  $y_j > 0$  is

$$\Pr(Y = y_j | y_j > 0, \mathbf{x}_j) = \frac{f(y_j)}{1 - (1 + \alpha \mu_j)^{-\alpha-1}}$$

The log likelihood (with weights  $w_j$  and offsets) is given by

$$\begin{aligned} m &= 1/\alpha & p_j &= 1/(1 + \alpha \mu_j) & \mu_j &= \exp(\mathbf{x}_j \boldsymbol{\beta} + \text{offset}_j) \\ \ln L &= \sum_{j=1}^n w_j \left[ \ln\{\Gamma(m + y_j)\} - \ln\{\Gamma(y_j + 1)\} \right. \\ &\quad \left. - \ln\{\Gamma(m)\} + m \ln(p_j) + y_j \ln(1 - p_j) - \ln(1 - p_j^m) \right] \end{aligned}$$

## Constant-dispersion model

The constant-dispersion model assumes that  $y_j$  is conditionally distributed as  $\text{Poisson}(\mu_j^*)$ , where  $\mu_j^* \sim \text{gamma}(\mu_j/\delta, \delta)$  for some dispersion parameter  $\delta$  (by contrast, the mean-dispersion model assumes  $\mu_j^* \sim \text{gamma}(1/\alpha, \alpha \mu_j)$ ). The log likelihood is given by

$$\begin{aligned} m_j &= \mu_j/\delta & p &= 1/(1 + \delta) \\ \ln L &= \sum_{j=1}^n w_j \left[ \ln\{\Gamma(m_j + y_j)\} - \ln\{\Gamma(y_j + 1)\} \right. \\ &\quad \left. - \ln\{\Gamma(m_j)\} + m_j \ln(p) + y_j \ln(1 - p) - \ln(1 - p^{m_j}) \right] \end{aligned}$$

with everything else defined as before in the calculations for the mean-dispersion model.

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

`ztnb` also supports estimation with survey data. For details on VCEs with survey data, see [SVY] [variance estimation](#).

## Acknowledgment

We gratefully acknowledge the previous work by Joseph Hilbe, Arizona State University, USA; see [Hilbe \(1999\)](#).

## References

- Cameron, A. C., and P. K. Trivedi. 1998. *Regression Analysis of Count Data*. Cambridge: Cambridge University Press.
- Grogger, J. T., and R. T. Carson. 1991. Models for truncated counts. *Journal of Applied Econometrics* 6: 225–238.
- Gutierrez, R. G., S. Carter, and D. M. Drukker. 2001. [sg160: On boundary-value likelihood-ratio tests](#). *Stata Technical Bulletin* 60: 15–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 269–273. College Station, TX: Stata Press.
- Hilbe, J. M. 1998. [sg91: Robust variance estimators for MLE Poisson and negative binomial regression](#). *Stata Technical Bulletin* 45: 26–28. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 177–180. College Station, TX: Stata Press.
- . 1999. [sg102: Zero-truncated Poisson and negative binomial regression](#). *Stata Technical Bulletin* 47: 37–40. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 233–236. College Station, TX: Stata Press.
- Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.
- Rodríguez, G. 1993. [sbe10: An improvement to poisson](#). *Stata Technical Bulletin* 11: 11–14. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 94–98. College Station, TX: Stata Press.
- Simonoff, J. S. 2003. *Analyzing Categorical Data*. New York: Springer.

## Also see

- [R] [ztnb postestimation](#) — Postestimation tools for `ztnb`
- [R] [nbreg](#) — Negative binomial regression
- [R] [poisson](#) — Poisson regression
- [R] [zinb](#) — Zero-inflated negative binomial regression
- [R] [zip](#) — Zero-inflated Poisson regression
- [R] [ztp](#) — Zero-truncated Poisson regression
- [SVY] [svy estimation](#) — Estimation commands for survey data
- [XT] [xtnbreg](#) — Fixed-effects, random-effects, & population-averaged negative binomial models
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `ztnb`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic nooffset]

predict [type] { stub* | newvarreg newvardisp } [if] [in] , scores
```

| statistic         | description                                             |
|-------------------|---------------------------------------------------------|
| Main              |                                                         |
| <code>n</code>    | number of events; the default                           |
| <code>ir</code>   | incidence rate                                          |
| <code>cm</code>   | estimate of the conditional mean, $E(y_j \mid y_j > 0)$ |
| <code>xb</code>   | linear prediction                                       |
| <code>stdp</code> | standard error of the linear prediction                 |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

**n**, the default, calculates the predicted number of events, which is  $\exp(\mathbf{x}_j\beta)$  if neither `offset()` nor `exposure()` was specified when the model was fit;  $\exp(\mathbf{x}_j\beta + \text{offset}_j)$  if `offset()` was specified; or  $\exp(\mathbf{x}_j\beta) \times \text{exposure}_j$  if `exposure()` was specified.

**ir** calculates the incidence rate  $\exp(\mathbf{x}_j\beta)$ , which is the predicted number of events when exposure is 1. This is equivalent to specifying both **n** and **nooffset** options.

**cm** calculates the estimate of the conditional mean.

$$E(y_j \mid y_j > 0) = \frac{E(y_j)}{\Pr(y_j > 0)}$$

**xb** calculates the linear prediction, which is  $\mathbf{x}_j\beta$  if neither `offset()` nor `exposure()` was specified;  $\mathbf{x}_j\beta + \text{offset}_j$  if `offset()` was specified; or  $\mathbf{x}_j\beta + \ln(\text{exposure}_j)$  if `exposure()` was specified; see **nooffset** below.

**stdp** calculates the standard error of the linear prediction.

**nooffset** is relevant only if you specified `offset()` or `exposure()` when you fit the model. It modifies the calculations made by **predict** so that they ignore the offset or exposure variable; the linear prediction is treated as  $\mathbf{x}_j\beta$  rather than as  $\mathbf{x}_j\beta + \text{offset}_j$  or  $\mathbf{x}_j\beta + \ln(\text{exposure}_j)$ . Specifying **predict ... , nooffset** is equivalent to specifying **predict ... , ir**.

**scores** calculates equation-level score variables.

The first new variable will contain  $\partial \ln L / \partial (\mathbf{x}_j\beta)$ .

The second new variable will contain  $\partial \ln L / \partial (\ln \alpha)$  for **dispersion(mean)**.

The second new variable will contain  $\partial \ln L / \partial (\ln \delta)$  for **dispersion(constant)**.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

In the following, we use the same notation as in [R] [ztnb](#).

Methods and formulas are presented under the following headings:

*Mean-dispersion model*

*Constant-dispersion model*

(Continued on next page)

## Mean-dispersion model

The equation-level scores are given by

$$\begin{aligned}\text{score}(\mathbf{x}\beta)_j &= p_j(y_j - \mu_j) - \frac{p_j^{(m+1)}\mu_j}{1 - p_j^m} \\ \text{score}(\tau)_j &= -m \left\{ \frac{\alpha(\mu_j - y_j)}{1 + \alpha\mu_j} - \ln(1 + \alpha\mu_j) + \psi(y_j + m) - \psi(m) \right\} \\ &\quad - \frac{p_j^m}{1 - p_j^m} \{m \ln(p_j) + \mu_j p_j\}\end{aligned}$$

where  $\tau_j = \ln\alpha_j$  and  $\psi(z)$  is the digamma function.

## Constant-dispersion model

The equation-level scores are given by

$$\begin{aligned}\text{score}(\mathbf{x}\beta)_j &= m_j \left\{ \psi(y_j + m_j) - \psi(m_j) + \ln(p) + \frac{p^{m_j} \ln(p)}{1 - p^{m_j}} \right\} \\ \text{score}(\tau)_j &= y_j - (y_j + m_j)(1 - p) - \text{score}(\mathbf{x}\beta)_j - \frac{\mu_j p}{1 - p^{m_j}}\end{aligned}$$

where  $\tau_j = \ln\delta_j$ .

## Also see

[R] **ztnb** — Zero-truncated negative binomial regression

[U] **20 Estimation and postestimation commands**



Syntax

ztp *depvar* [*indepvars*] [*if*] [*in*] [*weight*] [, *options*]

| <i>options</i>                                 | description                                                                                                               |
|------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| Model                                          |                                                                                                                           |
| <u>n</u> o <u>c</u> onstant                    | suppress constant term                                                                                                    |
| <u>e</u> xposure( <i>varname<sub>e</sub></i> ) | include ln( <i>varname<sub>e</sub></i> ) in model with coefficient constrained to 1                                       |
| <u>o</u> ffset( <i>varname<sub>o</sub></i> )   | include <i>varname<sub>o</sub></i> in model with coefficient constrained to 1                                             |
| <u>c</u> onstraints( <i>constraints</i> )      | apply specified linear constraints                                                                                        |
| <u>c</u> ollinear                              | keep collinear variables                                                                                                  |
| SE/Robust                                      |                                                                                                                           |
| vce( <i>vcetype</i> )                          | <i>vcetype</i> may be oim, <u>r</u> obust, <u>c</u> luster <i>clustvar</i> , opg, <u>b</u> ootstrap, or <u>j</u> ackknife |
| Reporting                                      |                                                                                                                           |
| <u>l</u> evel(#)                               | set confidence level; default is level(95)                                                                                |
| <u>i</u> rr                                    | report incidence-rate ratios                                                                                              |
| <u>n</u> o <u>c</u> ns <u>r</u> eport          | do not display constraints                                                                                                |
| <i>display_options</i>                         | control spacing and display of omitted variables and base and empty cells                                                 |
| Maximization                                   |                                                                                                                           |
| <i>maximize_options</i>                        | control the maximization process; seldom used                                                                             |
| <sup>†</sup> <u>c</u> oef <u>l</u> egend       | display coefficients' legend instead of coefficient table                                                                 |

<sup>†</sup>coeflegend does not appear in the dialog box.  
*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.  
*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.  
*bootstrap*, *by*, *jackknife*, *rolling*, *statsby*, and *svy* are allowed; see [U] 11.1.10 Prefix commands.  
Weights are not allowed with the *bootstrap* prefix.  
*vce()* and weights are not allowed with the *svy* prefix.  
*fweights*, *iweights*, and *pweights* are allowed; see [U] 11.1.6 weight.  
See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Menu

Statistics > Count outcomes > Zero-truncated Poisson regression

## Description

**ztp** fits a zero-truncated Poisson (ZTP) regression of *depvar* on *indepvars*, where *depvar* is a positive count variable.

## Options for ztp

### Model

**noconstant**, **exposure**(*varname<sub>e</sub>*), **offset**(*varname<sub>o</sub>*), **constraints**(*constraints*), **collinear**; see [R] [estimation options](#).

### SE/Robust

**vce**(*vcetype*) specifies the type of standard error reported, which includes types that are derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods; see [R] [vce\\_option](#).

### Reporting

**level**(#); see [R] [estimation options](#).

**irr** reports estimated coefficients transformed to incidence-rate ratios, that is,  $e^{\beta_i}$  rather than  $\beta_i$ . Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated or stored. **irr** may be specified at estimation or when replaying previously estimated results.

**nocnsreport**; see [R] [estimation options](#).

**display\_options**: **noomitted**, **vsquish**, **noemptycells**, **baselevels**, **allbaselevels**; see [R] [estimation options](#).

### Maximization

**maximize\_options**: **difficult**, **technique**(*algorithm\_spec*), **iterate**(#), **[no]log**, **trace**, **gradient**, **showstep**, **hessian**, **showtolerance**, **tolerance**(#), **ltolerance**(#), **nrtolerance**(#), **nonrtolerance**, **from**(*init\_specs*); see [R] [maximize](#). These options are seldom used.

Setting the optimization type to **technique(bhhh)** resets the default *vcetype* to **vce(opg)**.

The following option is available with **ztp** but is not shown in the dialog box:

**coeflegend**; see [R] [estimation options](#).

## Remarks

ZTP regression is used to model the number of occurrences of an event when that number is restricted to be positive. If zero is an admissible value for the dependent variable, then standard Poisson regression may be more appropriate; see [R] [poisson](#). ZTP regression was first proposed by [Grogger and Carson \(1991\)](#). See [Long \(1997, chap. 8\)](#) for an introduction to Poisson and ZTP regression.

Suppose that you counted the number of times a patient has medical treatments, and those patients who have at least one treatment are randomly selected into the sample. You might examine the length of hospital stay of patients, which has to be positive. Then the `ztp` command should be used instead of `poisson`.

## ► Example 1

We will illustrate the ZTP model by using the dataset of running shoes for a sample of runners who registered an online running log, which is used by [Simonoff \(2003\)](#). A running-shoe marketing executive is interested in knowing how the number of running shoes purchased related to other factors such as gender, marital status, age, education, income, typical number of runs per week, average miles run per week, and the preferred type of running. These data are naturally truncated. We fit a ZTP regression model based on distance running, runs per week, miles run per week, gender, and age.

```
. use http://www.stata-press.com/data/r11/runshoes
. ztp shoes distance rpweek mpweek male age

Iteration 0: log likelihood = -86.267921
Iteration 1: log likelihood = -84.096471
Iteration 2: log likelihood = -84.079524
Iteration 3: log likelihood = -84.079517

Zero-truncated Poisson regression Number of obs = 60
 LR chi2(5) = 27.10
 Prob > chi2 = 0.0001
Log likelihood = -84.079517 Pseudo R2 = 0.1388
```

| shoes    | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |          |
|----------|-----------|-----------|-------|-------|----------------------|----------|
| distance | .5101724  | .2336767  | 2.18  | 0.029 | .0521746             | .9681703 |
| rpweek   | .1822472  | .1146021  | 1.59  | 0.112 | -.0423688            | .4068631 |
| mpweek   | .0143156  | .0099102  | 1.44  | 0.149 | -.0051081            | .0337393 |
| male     | .1314745  | .2480547  | 0.53  | 0.596 | -.3547039            | .6176528 |
| age      | .011772   | .010266   | 1.15  | 0.252 | -.0083489            | .0318929 |
| _cons    | -1.174196 | .6524461  | -1.80 | 0.072 | -2.452967            | .1045745 |

◀

(Continued on next page)

## Saved results

**ztp** saves the following in `e()`:

### Scalars

|                            |                                                |
|----------------------------|------------------------------------------------|
| <code>e(N)</code>          | number of observations                         |
| <code>e(k)</code>          | number of parameters                           |
| <code>e(k_eq)</code>       | number of equations                            |
| <code>e(k_eq_model)</code> | number of equations in model Wald test         |
| <code>e(k_dv)</code>       | number of dependent variables                  |
| <code>e(k_autoCns)</code>  | number of base, empty, and omitted constraints |
| <code>e(df_m)</code>       | model degrees of freedom                       |
| <code>e(r2_p)</code>       | pseudo- <i>R</i> -squared                      |
| <code>e(ll)</code>         | log likelihood                                 |
| <code>e(ll_0)</code>       | log likelihood, constant-only model            |
| <code>e(N_clust)</code>    | number of clusters                             |
| <code>e(chi2)</code>       | $\chi^2$                                       |
| <code>e(p)</code>          | significance                                   |
| <code>e(rank)</code>       | rank of <code>e(V)</code>                      |
| <code>e(ic)</code>         | number of iterations                           |
| <code>e(rc)</code>         | return code                                    |
| <code>e(converged)</code>  | 1 if converged, 0 otherwise                    |

### Macros

|                                 |                                                                          |
|---------------------------------|--------------------------------------------------------------------------|
| <code>e(cmd)</code>             | <b>ztp</b>                                                               |
| <code>e(cmdline)</code>         | command as typed                                                         |
| <code>e(depvar)</code>          | name of dependent variable                                               |
| <code>e(wtype)</code>           | weight type                                                              |
| <code>e(wexp)</code>            | weight expression                                                        |
| <code>e(title)</code>           | title in estimation output                                               |
| <code>e(clustvar)</code>        | name of cluster variable                                                 |
| <code>e(offset)</code>          | offset                                                                   |
| <code>e(chi2type)</code>        | Wald or LR; type of model $\chi^2$ test                                  |
| <code>e(vce)</code>             | <i>vcetype</i> specified in <code>vce()</code>                           |
| <code>e(vcetype)</code>         | title used to label Std. Err.                                            |
| <code>e(opt)</code>             | type of optimization                                                     |
| <code>e(which)</code>           | max or min; whether optimizer is to perform maximization or minimization |
| <code>e(ml_method)</code>       | type of <code>ml</code> method                                           |
| <code>e(user)</code>            | name of likelihood-evaluator program                                     |
| <code>e(technique)</code>       | maximization technique                                                   |
| <code>e(singularHmethod)</code> | m-marquardt or hybrid; method used when Hessian is singular              |
| <code>e(crittype)</code>        | optimization criterion                                                   |
| <code>e(properties)</code>      | <code>b V</code>                                                         |
| <code>e(predict)</code>         | program used to implement <code>predict</code>                           |
| <code>e(asbalanced)</code>      | factor variables <code>fvset</code> as <code>asbalanced</code>           |
| <code>e(asobserved)</code>      | factor variables <code>fvset</code> as <code>asobserved</code>           |

## Matrices

|                              |                                              |
|------------------------------|----------------------------------------------|
| <code>e(b)</code>            | coefficient vector                           |
| <code>e(Cns)</code>          | constraints matrix                           |
| <code>e(ilog)</code>         | iteration log (up to 20 iterations)          |
| <code>e(gradient)</code>     | gradient vector                              |
| <code>e(V)</code>            | variance–covariance matrix of the estimators |
| <code>e(V_modelbased)</code> | model-based variance                         |

## Functions

|                        |                         |
|------------------------|-------------------------|
| <code>e(sample)</code> | marks estimation sample |
|------------------------|-------------------------|

## Methods and formulas

`ztp` is implemented as an ado-file.

The conditional probability of observing  $y_j$  events given that  $y_j > 0$  is

$$\Pr(Y = y_j \mid y_j > 0, \mathbf{x}_j) = \frac{e^{-\lambda} \lambda^{y_j}}{y_j! (1 - e^{-\lambda})}$$

The log likelihood (with weights  $w_j$  and offsets) is given by

$$\begin{aligned} \xi_j &= \mathbf{x}_j \boldsymbol{\beta} + \text{offset}_j \\ f(y_j) &= \frac{e^{-\exp(\xi_j)} e^{\xi_j y_j}}{y_j! \{1 - e^{-\exp(\xi_j)}\}} \\ \ln L &= \sum_{j=1}^n w_j \left[ -e^{\xi_j} + \xi_j y_j - \ln(y_j!) - \ln\{1 - e^{-\exp(\xi_j)}\} \right] \end{aligned}$$

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] [\\_robust](#), in particular, in [Maximum likelihood estimators](#) and [Methods and formulas](#).

`ztp` also supports estimation with survey data. For details on VCEs with survey data, see [SVY] [variance estimation](#).

## Acknowledgment

We gratefully acknowledge the previous work by Joseph Hilbe, Arizona State University, USA; see [Hilbe \(1999\)](#).

## References

- Cameron, A. C., and P. K. Trivedi. 1998. *Regression Analysis of Count Data*. Cambridge: Cambridge University Press.
- Grogger, J. T., and R. T. Carson. 1991. Models for truncated counts. *Journal of Applied Econometrics* 6: 225–238.
- Hilbe, J. M. 1998. [sg91: Robust variance estimators for MLE Poisson and negative binomial regression](#). *Stata Technical Bulletin* 45: 26–28. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 177–180. College Station, TX: Stata Press.

- . 1999. [sg102: Zero-truncated Poisson and negative binomial regression](#). *Stata Technical Bulletin* 47: 37–40. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 233–236. College Station, TX: Stata Press.
- Hilbe, J. M., and D. H. Judson. 1998. [sg94: Right, left, and uncensored Poisson regression](#). *Stata Technical Bulletin* 46: 18–20. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 186–189. College Station, TX: Stata Press.
- Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.
- Simonoff, J. S. 2003. *Analyzing Categorical Data*. New York: Springer.

## Also see

- [R] [ztp postestimation](#) — Postestimation tools for ztp
- [R] [nbreg](#) — Negative binomial regression
- [R] [poisson](#) — Poisson regression
- [R] [zinb](#) — Zero-inflated negative binomial regression
- [R] [zip](#) — Zero-inflated Poisson regression
- [R] [ztnb](#) — Zero-truncated negative binomial regression
- [XT] [xtpoisson](#) — Fixed-effects, random-effects, and population-averaged Poisson models
- [U] [20 Estimation and postestimation commands](#)

Description

The following postestimation commands are available for `ztp`:

| command                          | description                                                                                         |
|----------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>estat</code>               | AIC, BIC, VCE, and estimation sample summary                                                        |
| <code>estat (svy)</code>         | postestimation statistics for survey data                                                           |
| <code>estimates</code>           | cataloging estimation results                                                                       |
| <code>lincom</code>              | point estimates, standard errors, testing, and inference for linear combinations of coefficients    |
| <code>lrtest</code> <sup>1</sup> | likelihood-ratio test                                                                               |
| <code>margins</code>             | marginal means, predictive margins, marginal effects, and average marginal effects                  |
| <code>nlcom</code>               | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| <code>predict</code>             | predictions, residuals, influence statistics, and other diagnostic measures                         |
| <code>predictnl</code>           | point estimates, standard errors, testing, and inference for generalized predictions                |
| <code>suest</code>               | seemingly unrelated estimation                                                                      |
| <code>test</code>                | Wald tests of simple and composite linear hypotheses                                                |
| <code>testnl</code>              | Wald tests of nonlinear hypotheses                                                                  |

<sup>1</sup> `lrtest` is not appropriate with `svy` estimation results.

See the corresponding entries in the *Base Reference Manual* for details, but see [\[SVY\] estat](#) for details about `estat (svy)`.

Syntax for predict

```
predict [type] newvar [if] [in] [, statistic nooffset]
```

| statistic           | description                                                                |
|---------------------|----------------------------------------------------------------------------|
| Main                |                                                                            |
| <code>n</code>      | number of events; the default                                              |
| <code>ir</code>     | incidence rate                                                             |
| <code>cm</code>     | estimate of the conditional mean, $E(y_j \mid y_j > 0)$                    |
| <code>xb</code>     | linear prediction                                                          |
| <code>stdp</code>   | standard error of the linear prediction                                    |
| <code>scores</code> | first derivative of the log likelihood with respect to $\mathbf{x}_j\beta$ |

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

## Menu

Statistics > Postestimation > Predictions, residuals, etc.

## Options for predict

Main

**n**, the default, calculates the predicted number of events, which is  $\exp(\mathbf{x}_j\beta)$  if neither `offset()` nor `exposure()` was specified when the model was fit;  $\exp(\mathbf{x}_j\beta + \text{offset}_j)$  if `offset()` was specified; or  $\exp(\mathbf{x}_j\beta) \times \text{exposure}_j$  if `exposure()` was specified.

**ir** calculates the incidence rate  $\exp(\mathbf{x}_j\beta)$ , which is the predicted number of events when exposure is 1. This is equivalent to specifying both **n** and **nooffset** options.

**cm** calculates the estimate of the conditional mean.

$$E(y_j \mid y_j > 0) = \frac{E(y_j)}{\Pr(y_j > 0)}$$

**xb** calculates the linear prediction, which is  $\mathbf{x}_j\beta$  if neither `offset()` nor `exposure()` was specified;  $\mathbf{x}_j\beta + \text{offset}_j$  if `offset()` was specified; or  $\mathbf{x}_j\beta + \ln(\text{exposure}_j)$  if `exposure()` was specified; see **nooffset** below.

**stdp** calculates the standard error of the linear prediction.

**score** calculates the equation-level score,  $\partial \ln L / \partial (\mathbf{x}_j\beta)$ .

**nooffset** is relevant only if you specified `offset()` or `exposure()` when you fit the model. It modifies the calculations made by **predict** so that they ignore the offset or exposure variable; the linear prediction is treated as  $\mathbf{x}_j\beta$  rather than as  $\mathbf{x}_j\beta + \text{offset}_j$  or  $\mathbf{x}_j\beta + \ln(\text{exposure}_j)$ . Specifying **predict** ..., **nooffset** is equivalent to specifying **predict** ..., **ir**.

## Methods and formulas

All postestimation commands listed above are implemented as ado-files.

In the following, we use the same notation as in [\[R\] ztp](#).

The equation-level scores are given by

$$\text{score}(\mathbf{x}\beta)_j = y_j - e^{\xi_j} - \frac{e^{-\exp(\xi_j)} e^{\xi_j}}{1 - e^{-\exp(\xi_j)}}$$

## Also see

[\[R\] ztp](#) — Zero-truncated Poisson regression

[\[U\] 20 Estimation and postestimation commands](#)



# Author index

This is the author index for the 3-volume *Stata Base Reference Manual*.

## A

Abramowitz, M., [R] [orthog](#)  
 Abrams, K. R., [R] [meta](#)  
 Abramson, J. H., [R] [kappa](#)  
 Abramson, Z. H., [R] [kappa](#)  
 Achen, C. H., [R] [scobit](#)  
 Acock, A. C., [R] [alpha](#), [R] [anova](#), [R] [correlate](#),  
     [R] [nestreg](#), [R] [oneway](#), [R] [prtest](#),  
     [R] [ranksum](#), [R] [ttest](#)  
 Adkins, L. C., [R] [heckman](#), [R] [regress](#), [R] [regress](#)  
     [postestimation](#)  
 Afifi, A. A., [R] [anova](#), [R] [stepwise](#)  
 Agresti, A., [R] [ci](#), [R] [expoisson](#), [R] [tabulate twoway](#)  
 Aigner, D., [R] [frontier](#)  
 Aiken, L. S., [R] [pcorr](#)  
 Aitchison, J., [R] [ologit](#), [R] [oprobit](#)  
 Aitken, A. C., [R] [reg3](#)  
 Aivazian, S. A., [R] [ksmirnov](#)  
 Akaike, H., [R] [BIC note](#), [R] [estat](#), [R] [glm](#)  
 Aldrich, J. H., [R] [logit](#), [R] [probit](#)  
 Alexandersson, A., [R] [regress](#)  
 Alf Jr., E., [R] [rocfit](#)  
 Alldredge, J. R., [R] [pk](#), [R] [pkcross](#)  
 Allen, M. J., [R] [alpha](#)  
 Allison, P. D., [R] [rologit](#), [R] [testnl](#)  
 Altman, D. G., [R] [anova](#), [R] [fracpoly](#), [R] [kappa](#),  
     [R] [kwallis](#), [R] [meta](#), [R] [mfp](#), [R] [nptrend](#),  
     [R] [oneway](#)  
 Ambler, G., [R] [fracpoly](#), [R] [mfp](#), [R] [regress](#)  
 Amemiya, T., [R] [glogit](#), [R] [intreg](#), [R] [ivprobit](#),  
     [R] [nlogit](#), [R] [tobit](#)  
 Andersen, E. B., [R] [clogit](#)  
 Anderson, J. A., [R] [ologit](#), [R] [slogit](#)  
 Anderson, R. E., [R] [rologit](#)  
 Anderson, R. L., [R] [anova](#)  
 Anderson, S., [R] [pkequiv](#)  
 Anderson, T. W., [R] [ivregress postestimation](#)  
 Andrews, D. F., [R] [rreg](#)  
 Andrews, D. W. K., [R] [ivregress](#)  
 Angrist, J. D., [R] [ivregress](#), [R] [ivregress](#)  
     [postestimation](#), [R] [qreg](#), [R] [regress](#)  
 Ansccombe, F. J., [R] [binreg postestimation](#), [R] [glm](#),  
     [R] [glm postestimation](#)  
 Arbuthnott, J., [R] [signrank](#)  
 Archer, K. J., [R] [logistic](#), [R] [logistic postestimation](#),  
     [R] [logit](#), [R] [logit postestimation](#)  
 Arellano, M., [R] [gmm](#)  
 Arminger, G., [R] [suest](#)  
 Armitage, P., [R] [ameans](#), [R] [expoisson](#), [R] [pkcross](#),  
     [R] [sdtest](#)

Armstrong, R. D., [R] [qreg](#)  
 Arthur, M., [R] [symmetry](#)  
 Atkinson, A. C., [R] [boxcox](#), [R] [nl](#)  
 Azen, S. P., [R] [anova](#)

## B

Babiker, A., [R] [sampsi](#)  
 Babin, B. J., [R] [rologit](#)  
 Baker, R. J., [R] [glm](#)  
 Baker, R. M., [R] [ivregress postestimation](#)  
 Bakker, A., [R] [mean](#)  
 Balaam, L. N., [R] [pkcross](#)  
 Baltagi, B. H., [R] [hausman](#)  
 Bamber, D., [R] [roc](#), [R] [rocfit](#)  
 Bancroft, T. A., [R] [stepwise](#)  
 Barnard, G. A., [R] [spearman](#), [R] [ttest](#)  
 Barnett, A. G., [R] [glm](#)  
 Barnow, B. S., [R] [treatreg](#)  
 Barrison, I. G., [R] [binreg](#)  
 Bartlett, M. S., [R] [oneway](#)  
 Bartus, T., [R] [margins](#)  
 Basmann, R. L., [R] [ivregress](#), [R] [ivregress](#)  
     [postestimation](#)  
 Bassett Jr., G., [R] [qreg](#)  
 Basu, A., [R] [glm](#)  
 Baum, C. F., [R] [gmm](#), [R] [heckman](#), [R] [heckprob](#),  
     [R] [ivregress](#), [R] [ivregress postestimation](#),  
     [R] [net](#), [R] [net search](#), [R] [regress](#)  
     [postestimation](#), [R] [regress postestimation time](#)  
     [series](#), [R] [ssc](#)  
 Bayart, D., [R] [qc](#)  
 Beale, E. M. L., [R] [stepwise](#), [R] [test](#)  
 Beaton, A. E., [R] [rreg](#)  
 Beckett, S., [R] [fracpoly](#), [R] [runtest](#), [R] [spearman](#)  
 Beggs, S., [R] [rologit](#)  
 Belanger, A. J., [R] [sktest](#)  
 Belsley, D. A., [R] [estat](#), [R] [regress postestimation](#)  
 Bendel, R. B., [R] [stepwise](#)  
 Benedetti, J. K., [R] [tetrachoric](#)  
 Beniger, J. R., [R] [cumul](#)  
 Bera, A. K., [R] [sktest](#)  
 Beran, R. J., [R] [regress postestimation time series](#)  
 Berk, K. N., [R] [stepwise](#)  
 Berk, R. A., [R] [rreg](#)  
 Berkson, J., [R] [logit](#), [R] [probit](#)  
 Bern, P. H., [R] [nestreg](#)  
 Bernasco, W., [R] [tetrachoric](#)  
 Berndt, E. K., [R] [glm](#)  
 Berndt, E. R., [R] [treatreg](#), [R] [truncreg](#)  
 Bernstein, I. H., [R] [alpha](#)  
 Berry, G., [R] [ameans](#), [R] [expoisson](#), [R] [sdtest](#)  
 Bewley, R., [R] [reg3](#)  
 Beyer, W. H., [R] [qc](#)  
 Bickeböller, H., [R] [symmetry](#)  
 Bickel, P. J., [R] [rreg](#)  
 Birdsall, T. G., [R] [logistic postestimation](#)  
 Black, W. C., [R] [rologit](#)

Blackwell III, J. L., [R] **areg**  
 Bland, M., [R] **ranksum**, [R] **sdtest**, [R] **signrank**,  
     [R] **spearman**  
 Bleda, M.-J., [R] **alpha**  
 Bliss, C. I., [R] **probit**  
 Bloch, D. A., [R] **brier**  
 Bloomfield, P., [R] **qreg**  
 Blundell, R., [R] **gmm**, [R] **ivprobit**  
 BMDP, [R] **symmetry**  
 Boice Jr., J. D., [R] **bitest**  
 Boland, P. J., [R] **ttest**  
 Bolduc, D., [R] **asmprobit**  
 Bollen, K. A., [R] **regress postestimation**  
 Bond, S., [R] **gmm**  
 Borenstein, M., [R] **meta**  
 Bound, J., [R] **ivregress postestimation**  
 Bowker, A. H., [R] **symmetry**  
 Box, G. E. P., [R] **anova**, [R] **boxcox**, [R] **lnskew0**  
 Box, J. F., [R] **anova**  
 Boyd, N. F., [R] **kappa**  
 Bradley, R. A., [R] **signrank**  
 Brady, A. R., [R] **logistic**, [R] **spikeplot**  
 Brant, R., [R] **ologit**  
 Breslow, N. E., [R] **clogit**, [R] **dstdize**, [R] **symmetry**  
 Breusch, T. S., [R] **mvreg**, [R] **regress postestimation**,  
     [R] **regress postestimation time series**, [R] **sureg**  
 Brier, G. W., [R] **brier**  
 Brillinger, D. R., [R] **jackknife**  
 Brook, R. H., [R] **brier**  
 Brown, D. R., [R] **anova**, [R] **loneway**, [R] **oneway**  
 Brown, L. D., [R] **ci**  
 Brown, M. B., [R] **sdtest**, [R] **tetrachoric**  
 Brown, S. E., [R] **symmetry**  
 Bru, B., [R] **poisson**  
 Buchner, D. M., [R] **ladder**  
 Bunch, D. S., [R] **asmprobit**  
 Burnam, M. A., [R] **lincom**, [R] **mlogit**, [R] **mprobit**,  
     [R] **mprobit postestimation**, [R] **predictnl**,  
     [R] **slogit**  
 Burr, I. W., [R] **qc**  
 Buskens, V., [R] **tabstat**

## C

Cai, T. T., [R] **ci**  
 Cain, G. G., [R] **treatreg**  
 Cameron, A. C., [R] **asclogit**, [R] **asmprobit**,  
     [R] **bootstrap**, [R] **gmm**, [R] **heckman**,  
     [R] **intreg**, [R] **ivregress**, [R] **ivregress**  
     **postestimation**, [R] **logit**, [R] **mprobit**,  
     [R] **nbreg**, [R] **ologit**, [R] **oprobit**, [R] **poisson**,  
     [R] **probit**, [R] **qreg**, [R] **regress**, [R] **regress**  
     **postestimation**, [R] **simulate**, [R] **sureg**,  
     [R] **tobit**, [R] **ztnb**, [R] **ztp**  
 Campbell, M. J., [R] **ci**, [R] **kappa**, [R] **logistic**  
     **postestimation**, [R] **poisson**, [R] **tabulate**  
     **twoway**  
 Cappellari, L., [R] **asmprobit**

Cardell, S., [R] **rologit**  
 Carlile, T., [R] **kappa**  
 Carlin, J., [R] **ameans**  
 Carroll, R. J., [R] **boxcox**, [R] **rreg**, [R] **sdtest**  
 Carson, R. T., [R] **ztnb**, [R] **ztp**  
 Carter, S., [R] **frontier**, [R] **lrtest**, [R] **nbreg**, [R] **ztnb**  
 Caudill, S. B., [R] **frontier**  
 Caulcutt, R., [R] **qc**  
 Chadwick, J., [R] **poisson**  
 Chamberlain, G., [R] **clogit**, [R] **gmm**  
 Chambers, J. M., [R] **diagnostic plots**, [R] **grmeanby**,  
     [R] **lowess**  
 Chang, I. M., [R] **margins**  
 Charlett, A., [R] **fracpoly**  
 Chatterjee, S., [R] **poisson**, [R] **regress**, [R] **regress**  
     **postestimation**  
 Chen, X., [R] **logistic**, [R] **logistic postestimation**,  
     [R] **logit**  
 Chiburis, R., [R] **heckman**, [R] **heckprob**, [R] **oprobit**  
 Choi, B. C. K., [R] **roc**, [R] **rocfit**  
 Chow, S. C., [R] **pk**, [R] **pkcross**, [R] **pkequiv**,  
     [R] **pkexamine**, [R] **pkshape**  
 Christakis, N., [R] **rologit**  
 Clarke, R. D., [R] **poisson**  
 Clarke-Pearson, D. L., [R] **roc**  
 Clarkson, D. B., [R] **tabulate twoway**  
 Clayton, D. G., [R] **cloglog**, [R] **cumul**  
 Clerget-Darpoux, F., [R] **symmetry**  
 Cleveland, W. S., [R] **diagnostic plots**, [R] **lowess**,  
     [R] **lpoly**, [R] **sunflower**  
 Cleves, M. A., [R] **binreg**, [R] **dstdize**, [R] **logistic**,  
     [R] **logit**, [R] **roc**, [R] **rocfit**, [R] **sdtest**,  
     [R] **symmetry**  
 Clogg, C. C., [R] **suest**  
 Clopper, C. J., [R] **ci**  
 Cobb, G. W., [R] **anova**  
 Cochran, W. G., [R] **ameans**, [R] **anova**, [R] **correlate**,  
     [R] **dstdize**, [R] **mean**, [R] **oneway**, [R] **poisson**,  
     [R] **probit**, [R] **proportion**, [R] **ranksum**,  
     [R] **ratio**, [R] **signrank**, [R] **total**  
 Coelli, T. J., [R] **frontier**  
 Cohen, J., [R] **kappa**, [R] **pcorr**  
 Cohen, P., [R] **pcorr**  
 Coleman, J. S., [R] **poisson**  
 Collett, D., [R] **clogit**, [R] **logistic**, [R] **logistic**  
     **postestimation**  
 Cong, R., [R] **tobit**, [R] **tobit postestimation**,  
     [R] **treatreg**, [R] **truncreg**  
 Conover, W. J., [R] **centile**, [R] **ksmirnov**, [R] **kwallis**,  
     [R] **nptrend**, [R] **sdtest**, [R] **spearman**,  
     [R] **tabulate twoway**  
 Conroy, R. M., [R] **intreg**  
 Cook, A., [R] **ci**  
 Cook, R. D., [R] **boxcox**, [R] **regress postestimation**  
 Coull, B. A., [R] **ci**  
 Cox, D. R., [R] **boxcox**, [R] **xlogistic**, [R] **expoisson**,  
     [R] **lnskew0**

Cox, N. J., [R] **ci**, [R] **cumul**, [R] **diagnostic plots**,  
[R] **histogram**, [R] **inequality**, [R] **kappa**,  
[R] **kdensity**, [R] **ladder**, [R] **lowess**,  
[R] **lpoly**, [R] **net**, [R] **net search**, [R] **regress**  
**postestimation**, [R] **search**, [R] **serrbar**,  
[R] **sktest**, [R] **smooth**, [R] **spikeplot**, [R] **ssc**,  
[R] **stem**, [R] **sunflower**, [R] **tabulate twoway**

Cragg, J. G., [R] **ivregress postestimation**

Cramér, H., [R] **tabulate twoway**

Cramer, J. S., [R] **logit**

Cronbach, L. J., [R] **alpha**

Cui, J., [R] **symmetry**

Curts-García, J., [R] **smooth**

Cuzick, J., [R] **kappa**, [R] **nptrend**

## D

D'Agostino Jr., R. B., [R] **sktest**

D'Agostino, R. B., [R] **sktest**

Daniel, C., [R] **diagnostic plots**, [R] **oneway**

Danuso, F., [R] **nl**

DasGupta, A., [R] **ci**

Davey Smith, G., [R] **meta**

David, H. A., [R] **spearman**, [R] **summarize**

Davidson, R., [R] **boxcox**, [R] **cnsreg**, [R] **gmm**,  
[R] **intreg**, [R] **ivregress**, [R] **ivregress**  
**postestimation**, [R] **mlogit**, [R] **nl**, [R] **nlstur**,  
[R] **reg3**, [R] **regress**, [R] **regress postestimation**  
**time series**, [R] **tobit**, [R] **truncreg**

Davison, A. C., [R] **bootstrap**

Day, N. E., [R] **clogit**, [R] **dstdize**, [R] **symmetry**

de Irala-Estévez, J., [R] **logistic**

De Luca, G., [R] **biprobit**, [R] **heckprob**, [R] **probit**

de Wolf, I., [R] **rologit**

Deaton, A., [R] **nlstur**

Deb, P., [R] **nbreg**

DeLong, D. M., [R] **roc**

DeLong, E. R., [R] **roc**

DeMaris, A., [R] **regress postestimation**

Dewey, M. E., [R] **correlate**

Digby, P. G. N., [R] **tetrachoric**

Dixon, W. J., [R] **tttest**

Dobson, A. J., [R] **glm**

Dohoo, I., [R] **regress**

Doll, R., [R] **poisson**

Donald, S. G., [R] **ivregress postestimation**

Donner, A., [R] **loneway**

Donoho, D. L., [R] **lpoly**

Dore, C. J., [R] **fracpoly**

Dorfman, D. D., [R] **rocfit**

Draper, N., [R] **eivreg**, [R] **oneway**, [R] **regress**,  
[R] **stepwise**

Drukker, D. M., [R] **asmpobit**, [R] **boxcox**,  
[R] **frontier**, [R] **lrtest**, [R] **nbreg**, [R] **tobit**,  
[R] **treatreg**, [R] **ztnb**

Duan, N., [R] **heckman**

Duncan, A. J., [R] **qc**

Dunlop, D. D., [R] **samps**

Dunn, G., [R] **kappa**

Dunnett, C. W., [R] **mprobit**

Dunnington, G. W., [R] **regress**

Dupont, W. D., [R] **logistic**, [R] **mkspline**,  
[R] **sunflower**

Durbin, J., [R] **ivregress postestimation**, [R] **regress**  
**postestimation time series**

Duval, R. D., [R] **bootstrap**, [R] **jackknife**

## E

Edgington, E. S., [R] **runtest**

Edwards, A. L., [R] **anova**

Edwards, A. W. F., [R] **tetrachoric**

Edwards, J. H., [R] **tetrachoric**

Efron, B., [R] **bootstrap**, [R] **qreg**

Efroymson, M. A., [R] **stepwise**

Egger, M., [R] **meta**

Eisenhart, C., [R] **correlate**, [R] **runtest**

Ellis, C. D., [R] **poisson**

Eltinge, J. L., [R] **test**

Emerson, J. D., [R] **lv**, [R] **stem**

Engle, R. F., [R] **regress postestimation time series**

Erdreich, L. S., [R] **roc**, [R] **rocfit**

Eubank, R. L., [R] **lpoly**

Evans, M. A., [R] **pk**, [R] **pkcross**

Everitt, B. S., [R] **anova**, [R] **gllamm**, [R] **glm**

Ewens, W. J., [R] **symmetry**

Ezekiel, M., [R] **regress postestimation**

## F

Fan, J., [R] **lpoly**

Fan, Y.-A., [R] **tabulate twoway**

Fang, K.-T., [R] **asmpobit**

Feiveson, A. H., [R] **nlcom**, [R] **ranksum**

Feldt, L. S., [R] **anova**

Ferri, H. A., [R] **kappa**

Fieller, E. C., [R] **pkequiv**

Fienberg, S. E., [R] **kwallis**, [R] **tabulate twoway**

Filon, L. N. G., [R] **correlate**

Findley, D. F., [R] **estat**

Findley, T. W., [R] **ladder**

Finney, D. J., [R] **probit**, [R] **tabulate twoway**

Fiorio, C. V., [R] **kdensity**

Fiser, D. H., [R] **logistic postestimation**

Fishell, E., [R] **kappa**

Fisher, L. D., [R] **anova**, [R] **dstdize**, [R] **oneway**

Fisher, N. I., [R] **regress postestimation time series**

Fisher, R. A., [R] **anova**, [R] **ranksum**, [R] **signrank**,  
[R] **tabulate twoway**

Flannery, B. P., [R] **dydx**, [R] **vwls**

Fleiss, J. L., [R] **dstdize**, [R] **kappa**, [R] **samps**

Ford, J. M., [R] **frontier**

Forsythe, A. B., [R] **sctest**

Forthofer, R. N., [R] **dstdize**

Foster, A., [R] **regress**

Fourier, J. B. J., [R] **cumul**  
 Fox, J., [R] **kdensity**, [R] **lv**  
 Fox, W. C., [R] **logistic postestimation**  
 Francia, R. S., [R] **swilk**  
 Freese, J., [R] **asprobit**, [R] **clogit**, [R] **cloglog**,  
 [R] **logistic**, [R] **logit**, [R] **mlogit**, [R] **mprobit**,  
 [R] **nbreg**, [R] **ologit**, [R] **oprobit**, [R] **poisson**,  
 [R] **probit**, [R] **regress**, [R] **regress**  
**postestimation**, [R] **zibn**, [R] **zip**  
 Frison, L., [R] **sampsi**  
 Frome, E. L., [R] **qreg**  
 Fu, V. K., [R] **ologit**  
 Fuller, W. A., [R] **regress**, [R] **spearman**

## G

Gall, J.-R. L., [R] **logistic**, [R] **logistic postestimation**  
 Gallant, A. R., [R] **ivregress**, [R] **nl**  
 Galton, F., [R] **correlate**, [R] **cumul**, [R] **summarize**  
 Gan, F. F., [R] **diagnostic plots**  
 Garrett, J. M., [R] **fracpoly**, [R] **logistic**, [R] **logistic**  
**postestimation**, [R] **regress postestimation**  
 Garsd, A., [R] **exlogistic**  
 Gasser, T., [R] **lpoly**  
 Gates, R., [R] **asmpobit**  
 Gauvreau, K., [R] **dstdize**, [R] **logistic**, [R] **sampsi**  
 Geisser, S., [R] **anova**  
 Gelbach, J., [R] **ivprobit**, [R] **ivtobit**  
 Gelman, R., [R] **margins**  
 Gentle, J. E., [R] **anova**, [R] **nl**  
 Genz, A., [R] **asmpobit**  
 Gerkins, V. R., [R] **symmetry**  
 Geweke, J., [R] **asmpobit**  
 Gibbons, J. D., [R] **ksmirnov**, [R] **spearman**  
 Giesen, D., [R] **tetrachoric**  
 Gijbels, I., [R] **lpoly**  
 Gillham, N. W., [R] **regress**  
 Gillispie, C. C., [R] **regress**  
 Gini, R., [R] **vwls**  
 Gleason, J. R., [R] **anova**, [R] **bootstrap**, [R] **ci**,  
 [R] **correlate**, [R] **loneway**, [R] **summarize**,  
 [R] **ttest**  
 Gleser, G., [R] **alpha**  
 Glidden, D. V., [R] **logistic**  
 Gnanadesikan, R., [R] **cumul**, [R] **diagnostic plots**  
 Godfrey, L. G., [R] **regress postestimation time series**  
 Goeden, G. B., [R] **kdensity**  
 Goldberger, A. S., [R] **intreg**, [R] **tobit**, [R] **treatreg**  
 Goldstein, R., [R] **brier**, [R] **correlate**, [R] **inequality**,  
 [R] **nl**, [R] **ologit**, [R] **oprobit**, [R] **ranksum**,  
 [R] **regress postestimation**  
 Golub, G. H., [R] **orthog**, [R] **tetrachoric**  
 Good, P. I., [R] **permute**, [R] **symmetry**, [R] **tabulate**  
**twoway**  
 Goodall, C., [R] **lowess**, [R] **rreg**  
 Goodman, L. A., [R] **tabulate twoway**  
 Gordon, M. G., [R] **binreg**  
 Gorman, J. W., [R] **stepwise**

Gosset [Student, pseud.], W. S., [R] **ttest**  
 Gosset, W. S., [R] **ttest**  
 Gould, W. W., [R] **bootstrap**, [R] **dydx**, [R] **frontier**,  
 [R] **grmeanby**, [R] **jackknife**, [R] **kappa**,  
 [R] **logistic**, [R] **margins**, [R] **maximize**,  
 [R] **mkspline**, [R] **ml**, [R] **net search**, [R] **nlcom**,  
 [R] **ologit**, [R] **oprobit**, [R] **predictnl**, [R] **qreg**,  
 [R] **rreg**, [R] **simulate**, [R] **sktest**, [R] **smooth**,  
 [R] **swilk**, [R] **testnl**  
 Gourieroux, C., [R] **hausman**, [R] **suest**, [R] **test**  
 Graubard, B. I., [R] **margins**, [R] **ml**, [R] **test**  
 Graybill, F. A., [R] **centile**  
 Green, D. M., [R] **logistic postestimation**  
 Greene, W. H., [R] **asclogit**, [R] **asmpobit**,  
 [R] **biprobit**, [R] **clogit**, [R] **cnsreg**, [R] **frontier**,  
 [R] **gmm**, [R] **heckman**, [R] **heckprob**,  
 [R] **hetprob**, [R] **ivregress**, [R] **logit**, [R] **lrtest**,  
 [R] **margins**, [R] **mkspline**, [R] **mlogit**,  
 [R] **nlogit**, [R] **nlstur**, [R] **pcorr**, [R] **probit**,  
 [R] **reg3**, [R] **regress**, [R] **regress postestimation**  
**time series**, [R] **sureg**, [R] **testnl**, [R] **treatreg**,  
 [R] **truncreg**, [R] **zibn**, [R] **zip**  
 Greenfield, S., [R] **alpha**, [R] **lincom**, [R] **mlogit**,  
 [R] **mprobit**, [R] **mprobit postestimation**,  
 [R] **predictnl**, [R] **slogit**  
 Greenhouse, S. W., [R] **anova**  
 Greenland, S., [R] **ci**, [R] **glogit**, [R] **ologit**, [R] **poisson**  
 Gregoire, A., [R] **kappa**  
 Griffith, J. L., [R] **brier**  
 Griffith, R., [R] **gmm**  
 Griffiths, W. E., [R] **cnsreg**, [R] **estat**, [R] **glogit**,  
 [R] **ivregress**, [R] **ivregress postestimation**,  
 [R] **logit**, [R] **probit**, [R] **regress**, [R] **regress**  
**postestimation**, [R] **test**  
 Grizzle, J. E., [R] **vwls**  
 Grogger, J. T., [R] **ztnb**, [R] **ztp**  
 Gronau, R., [R] **heckman**  
 Gropper, D. M., [R] **frontier**  
 Guan, W., [R] **bootstrap**  
 Gutierrez, R. G., [R] **frontier**, [R] **lpoly**, [R] **lrtest**,  
 [R] **nbreg**, [R] **ztnb**

## H

Haan, P., [R] **asmpobit**, [R] **mlogit**, [R] **mprobit**  
 Hadi, A. S., [R] **poisson**, [R] **regress**, [R] **regress**  
**postestimation**  
 Hadorn, D. C., [R] **brier**  
 Hahn, J., [R] **ivregress postestimation**  
 Hair Jr., J. F., [R] **ologit**  
 Hajivassiliou, V. A., [R] **asmpobit**  
 Hald, A., [R] **qreg**, [R] **regress**, [R] **signrank**,  
 [R] **summarize**  
 Hall, A. D., [R] **frontier**  
 Hall, A. R., [R] **gmm**, [R] **gmm postestimation**,  
 [R] **ivregress**, [R] **ivregress postestimation**  
 Hall, B. H., [R] **glm**  
 Hall, P., [R] **bootstrap**, [R] **regress postestimation time**  
**series**

Hall, R. E., [R] **glm**  
Hall, W. J., [R] **roc**, [R] **rocf**  
Hallock, K., [R] **qreg**  
Halvorsen, K. T., [R] **tabulate twoway**  
Hamerle, A., [R] **clogit**  
Hamilton, J. D., [R] **gmm**  
Hamilton, L. C., [R] **bootstrap**, [R] **diagnostic plots**,  
[R] **ladder**, [R] **lv**, [R] **mlogit**, [R] **regress**,  
[R] **regress postestimation**, [R] **rreg**,  
[R] **simulate**, [R] **summarize**, [R] **ttest**  
Hampel, F. R., [R] **rreg**  
Hanley, J. A., [R] **roc**, [R] **rocf**  
Hansen, L. P., [R] **gmm**, [R] **ivregress**, [R] **ivregress**  
**postestimation**  
Hao, L., [R] **qreg**  
Hardin, J. W., [R] **binreg**, [R] **biprobit**, [R] **estat**,  
[R] **glm**, [R] **glm postestimation**, [R] **regress**  
**postestimation**  
Haritou, A., [R] **suest**  
Harkness, J., [R] **ivprobit**, [R] **ivtobit**  
Harrell Jr., F. E., [R] **mkspline**, [R] **ologit**  
Harris, R. L., [R] **qc**  
Harris, T., [R] **qreg**  
Harrison, D. A., [R] **histogram**, [R] **tabulate oneway**,  
[R] **tabulate twoway**  
Harrison, J. A., [R] **dstdize**  
Harvey, A. C., [R] **hetprob**  
Hastie, T. J., [R] **grmeanby**, [R] **slogit**  
Hauck, W. W., [R] **pkequiv**  
Hausman, J. A., [R] **glm**, [R] **hausman**, [R] **ivregress**  
**postestimation**, [R] **nlogit**, [R] **rologit**, [R] **suest**  
Hayashi, F., [R] **gmm**, [R] **ivregress**, [R] **ivregress**  
**postestimation**  
Hayes, R. J., [R] **permute**  
Hays, R. D., [R] **lincom**, [R] **mlogit**, [R] **mprobit**,  
[R] **mprobit postestimation**, [R] **predictnl**,  
[R] **slogit**  
Heagerty, P. J., [R] **anova**, [R] **dstdize**, [R] **oneway**  
Heckman, J., [R] **biprobit**, [R] **heckman**, [R] **heckman**  
**postestimation**, [R] **heckprob**  
Hedges, L. V., [R] **meta**  
Heiss, F., [R] **nlogit**  
Henderson, B. E., [R] **symmetry**  
Hendrickx, J., [R] **mlogit**, [R] **xi**  
Hensher, D. A., [R] **nlogit**  
Hickam, D. H., [R] **brier**  
Higgins, J. E., [R] **anova**  
Higgins, J. P. T., [R] **meta**  
Hilbe, J. M., [R] **cloglog**, [R] **estat**, [R] **glm**, [R] **glm**  
**postestimation**, [R] **logistic**, [R] **logit**, [R] **nbreg**,  
[R] **poisson**, [R] **probit**, [R] **sampsi**, [R] **ztnb**,  
[R] **ztp**  
Hill, A. B., [R] **poisson**  
Hill, R. C., [R] **cnsreg**, [R] **estat**, [R] **glogit**,  
[R] **heckman**, [R] **ivregress**, [R] **ivregress**  
**postestimation**, [R] **logit**, [R] **probit**,  
[R] **regress**, [R] **regress postestimation**, [R] **test**  
Hills, M., [R] **cloglog**, [R] **cumul**

Hinkley, D. V., [R] **bootstrap**  
Hirji, K. F., [R] **exlogistic**, [R] **expoisson**  
Hoaglin, D. C., [R] **diagnostic plots**, [R] **lv**, [R] **regress**  
**postestimation**, [R] **smooth**, [R] **stem**  
Hochberg, Y., [R] **oneway**  
Hocking, R. R., [R] **stepwise**  
Hoel, P. G., [R] **bitest**, [R] **ttest**  
Hoffmann, J. P., [R] **glm**  
Hole, A. R., [R] **asmprobit**, [R] **clogit**, [R] **mlogit**,  
[R] **mprobit**  
Holloway, L., [R] **brier**  
Holm, S., [R] **test**  
Holmes, S., [R] **bootstrap**  
Hood, W. C., [R] **ivregress**  
Hosmer Jr., D. W., [R] **clogit**, [R] **glm**, [R] **glogit**,  
[R] **lincom**, [R] **logistic**, [R] **logistic**  
**postestimation**, [R] **logit**, [R] **logit**  
**postestimation**, [R] **lrtest**, [R] **mlogit**,  
[R] **predictnl**, [R] **stepwise**  
Hotelling, H., [R] **roc**, [R] **rocf**  
Huang, C., [R] **sunflower**  
Huang, D. S., [R] **nlstur**, [R] **sureg**  
Huber, P. J., [R] **qreg**, [R] **rreg**, [R] **suest**  
Hunter, D. R., [R] **qreg**  
Hurd, M., [R] **intreg**, [R] **tobit**  
Hutto, C., [R] **exlogistic**  
Huynh, H., [R] **anova**

## I

Iglewicz, B., [R] **lv**  
Isaacs, D., [R] **fracpoly**  
Ishiguro, M., [R] **BIC note**

## J

Jackman, R. W., [R] **regress postestimation**  
Jacobs, K. B., [R] **symmetry**  
Jaeger, D. A., [R] **ivregress postestimation**  
Janes, H., [R] **rocf**  
Jann, B., [R] **estimates store**, [R] **ksmirnov**, [R] **saved**  
**results**, [R] **tabulate twoway**  
Jarque, C. M., [R] **sktest**  
Jeffreys, H., [R] **ci**, [R] **spearman**  
Jenkins, S. P., [R] **asmprobit**, [R] **do**, [R] **inequality**  
Joe, H., [R] **tabulate twoway**  
Johnson, D. E., [R] **anova**  
Johnson, M. E., [R] **sdtest**  
Johnson, M. M., [R] **sdtest**  
Johnson, N. L., [R] **ksmirnov**, [R] **nbreg**, [R] **poisson**  
Jolliffe, D., [R] **inequality**, [R] **qreg**, [R] **regress**  
Jolliffe, I. T., [R] **brier**  
Jones, A., [R] **heckman**, [R] **logit**, [R] **probit**  
Jones, D. R., [R] **meta**  
Jones, M. C., [R] **kdensity**, [R] **lpoly**  
Judge, G. G., [R] **estat**, [R] **glogit**, [R] **ivregress**,  
[R] **ivregress postestimation**, [R] **logit**,  
[R] **probit**, [R] **regress postestimation**, [R] **test**

Judson, D. H., [R] [poisson](#), [R] [tabulate twoway](#),  
[R] [ztp](#)

Juul, S., [R] [dstdize](#), [R] [roc](#)

## K

Kahn, H. A., [R] [dstdize](#)

Kaiser, J., [R] [permute](#), [R] [signrank](#)

Kalmijn, M., [R] [tetrachoric](#)

Keane, M. P., [R] [asmprobit](#)

Keeler, E. B., [R] [brier](#)

Kemp, A. W., [R] [nbreg](#), [R] [poisson](#)

Kemphorne, P. J., [R] [regress postestimation](#)

Kendall, M. G., [R] [centile](#), [R] [spearman](#),

[R] [tabulate twoway](#), [R] [tobit](#)

Kennedy Jr., W. J., [R] [anova](#), [R] [nl](#), [R] [regress](#),  
[R] [stepwise](#)

Kettenring, J. R., [R] [diagnostic plots](#)

Keynes, J. M., [R] [ameans](#)

Kiernan, M., [R] [kappa](#)

Kirkwood, B. R., [R] [dstdize](#), [R] [summarize](#)

Kish, L., [R] [loneway](#)

Kitagawa, G., [R] [BIC note](#)

Klar, J., [R] [logistic postestimation](#)

Kleiber, C., [R] [inequality](#)

Klein, L. R., [R] [reg3](#), [R] [regress postestimation time series](#)

Klein, M., [R] [binreg](#), [R] [clogit](#), [R] [logistic](#),  
[R] [lrtest](#), [R] [mlogit](#), [R] [ologit](#)

Kleinbaum, D. G., [R] [binreg](#), [R] [clogit](#), [R] [logistic](#),  
[R] [lrtest](#), [R] [mlogit](#), [R] [ologit](#)

Kleiner, B., [R] [diagnostic plots](#), [R] [lowess](#)

Kmenta, J., [R] [eivreg](#), [R] [ivregress](#), [R] [regress](#)

Koch, G. G., [R] [anova](#), [R] [kappa](#), [R] [vwls](#)

Koehler, K. J., [R] [diagnostic plots](#)

Koenker, R., [R] [qreg](#), [R] [regress postestimation](#)

Kohler, U., [R] [kdensity](#), [R] [regress](#), [R] [regress postestimation](#)

Kolmogorov, A. N., [R] [ksmirnov](#)

Koopmans, T. C., [R] [ivregress](#)

Korn, E. L., [R] [margins](#), [R] [ml](#), [R] [test](#)

Kotz, S., [R] [inequality](#), [R] [ksmirnov](#), [R] [nbreg](#),  
[R] [nlogit](#), [R] [poisson](#)

Kreuter, F., [R] [kdensity](#), [R] [regress](#), [R] [regress postestimation](#)

Krushelnitsky, B., [R] [inequality](#), [R] [qreg](#)

Kruskal, W. H., [R] [kwallis](#), [R] [ranksum](#),

[R] [spearman](#), [R] [tabulate twoway](#)

Kuehl, R. O., [R] [anova](#), [R] [oneway](#)

Kuh, E., [R] [estat](#), [R] [regress postestimation](#)

Kumbhakar, S. C., [R] [frontier](#)

Kung, D. S., [R] [qreg](#)

Kutner, M. H., [R] [pkcross](#), [R] [pkequiv](#), [R] [pkshape](#),  
[R] [regress postestimation](#)

Lacy, M. G., [R] [permute](#)

Lafontaine, F., [R] [boxcox](#)

Lahiri, K., [R] [tobit](#)

Lai, S., [R] [exlogistic](#)

Laird, N. M., [R] [expoison](#)

Lambert, D., [R] [zip](#)

Landis, J. R., [R] [kappa](#)

Lane, P. W., [R] [margins](#)

Lange, K., [R] [qreg](#)

Larsen, W. A., [R] [regress postestimation](#)

Lash, T. L., [R] [ci](#), [R] [glogit](#), [R] [poisson](#)

Lauritzen, S. L., [R] [summarize](#)

Lee, E. S., [R] [dstdize](#)

Lee, E. T., [R] [roc](#), [R] [rocfit](#)

Lee, T.-C., [R] [estat](#), [R] [glogit](#), [R] [ivregress](#),  
[R] [ivregress postestimation](#), [R] [logit](#),  
[R] [probit](#), [R] [regress postestimation](#), [R] [test](#)

Lee, W. C., [R] [roc](#)

Legendre, A. M., [R] [regress](#)

Lehmann, E. L., [R] [oneway](#)

Lemeshow, S., [R] [clogit](#), [R] [glm](#), [R] [glogit](#),  
[R] [lincom](#), [R] [logistic](#), [R] [logistic postestimation](#), [R] [logit](#), [R] [logit postestimation](#), [R] [lrtest](#), [R] [mlogit](#),  
[R] [predictnl](#), [R] [stepwise](#)

Leroy, A. M., [R] [qreg](#), [R] [regress postestimation](#),  
[R] [rreg](#)

Levene, H., [R] [sdtest](#)

Levin, B., [R] [dstdize](#), [R] [kappa](#), [R] [sampsi](#)

Levinsohn, J., [R] [frontier](#)

Levy, D., [R] [sunflower](#)

Lewis, H. G., [R] [heckman](#)

Lewis, I. G., [R] [binreg](#)

Lewis, J. D., [R] [fracpoly](#)

Li, G., [R] [rreg](#)

Li, W., [R] [pkcross](#), [R] [pkequiv](#), [R] [pkshape](#)

Likert, R. A., [R] [alpha](#)

Lim, G. C., [R] [cnsreg](#), [R] [regress](#), [R] [regress postestimation](#)

Lindley, D. V., [R] [ci](#)

Linhart, J. M., [R] [lpoly](#)

Lipset, S. M., [R] [histogram](#)

Liu, J. P., [R] [pk](#), [R] [pkcross](#), [R] [pkequiv](#),  
[R] [pkexamine](#), [R] [pkshape](#)

Locke, C. S., [R] [pkequiv](#)

Lokshin, M., [R] [heckman](#), [R] [heckprob](#), [R] [oprobit](#)

Long, J. S., [R] [asprobit](#), [R] [clogit](#), [R] [cloglog](#),  
[R] [intreg](#), [R] [logistic](#), [R] [logit](#), [R] [mlogit](#),  
[R] [mprobit](#), [R] [nbreg](#), [R] [ologit](#), [R] [oprobit](#),  
[R] [poisson](#), [R] [probit](#), [R] [regress](#), [R] [regress postestimation](#), [R] [testnl](#), [R] [tobit](#), [R] [zinb](#),  
[R] [zip](#), [R] [ztnb](#), [R] [ztp](#)

Longley, J. D., [R] [kappa](#)

Longton, G., [R] [rocfit](#)

López-Feldman, A., [R] [inequality](#)

Lorenz, M. O., [R] [inequality](#)

Louis, T. A., [R] [tabulate twoway](#)

Lovell, C. A. K., [R] [frontier](#)

## L

Lachenbruch, P. A., [R] [diagnostic plots](#)



Lovie, A. D., [R] **spearman**  
 Lovie, P., [R] **spearman**  
 Lucas, H. L., [R] **pkcross**  
 Luce, R. D., [R] **rologit**  
 Lumley, T. S., [R] **anova**, [R] **dstdize**, [R] **oneway**  
 Lunt, M., [R] **ologit**, [R] **slogit**  
 Lütkepohl, H., [R] **estat**, [R] **glogit**, [R] **ivregress**,  
 [R] **ivregress postestimation**, [R] **logit**,  
 [R] **probit**, [R] **regress postestimation**, [R] **test**

## M

Ma, G., [R] **roc**, [R] **rocf**  
 Machin, D., [R] **ci**, [R] **kappa**, [R] **tabulate twoway**  
 Mack, T. M., [R] **symmetry**  
 MacKinnon, J. G., [R] **boxcox**, [R] **cnsreg**, [R] **gmm**,  
 [R] **intreg**, [R] **ivregress**, [R] **ivregress**  
**postestimation**, [R] **mlogit**, [R] **nl**, [R] **nl**  
**nl**, [R] **nl**, [R] **nl**, [R] **nl**, [R] **nl**,  
 [R] **reg3**, [R] **regress**, [R] **regress postestimation**  
**time series**, [R] **tobit**, [R] **truncreg**  
 MacRae, K. D., [R] **binreg**  
 Madansky, A., [R] **runtest**  
 Maddala, G. S., [R] **nlogit**, [R] **tobit**, [R] **treatreg**  
 Mallows, C. L., [R] **regress postestimation**  
 Mander, A., [R] **symmetry**  
 Mann, H. B., [R] **kwallis**, [R] **ranksum**  
 Manning, W. G., [R] **heckman**  
 Manski, C. F., [R] **gmm**  
 Mantel, N., [R] **stepwise**  
 Marchenko, Y., [R] **anova**, [R] **loneway**, [R] **oneway**  
 Marden, J. I., [R] **rologit**  
 Markowski, C. A., [R] **sdtest**  
 Markowski, E. P., [R] **sdtest**  
 Marschak, J., [R] **ivregress**  
 Martin, W., [R] **regress**  
 Martínez, M. A., [R] **logistic**  
 Massey Jr., F. J., [R] **ttest**  
 Master, I. M., [R] **exlogistic**  
 Mastrucci, M. T., [R] **exlogistic**  
 Matthews, J. N. S., [R] **ameans**, [R] **expoisson**,  
 [R] **sdtest**  
 Mátyás, L., [R] **gmm**  
 Maxwell, A. E., [R] **symmetry**  
 McCleary, S. J., [R] **regress postestimation**  
 McCullagh, P., [R] **binreg**, [R] **binreg postestimation**,  
 [R] **glm**, [R] **glm postestimation**, [R] **ologit**,  
 [R] **rologit**  
 McCulloch, C. E., [R] **logistic**  
 McDonald, J. A., [R] **sunflower**  
 McDonald, J. F., [R] **tobit**, [R] **tobit postestimation**  
 McDowell, A. W., [R] **sureg**  
 McFadden, D. L., [R] **asclogit**, [R] **asmpobit**,  
 [R] **clogit**, [R] **hausman**, [R] **maximize**,  
 [R] **nlogit**, [R] **suest**  
 McGill, R., [R] **sunflower**  
 McGinnis, R. E., [R] **symmetry**  
 McGuire, T. J., [R] **dstdize**  
 McKelvey, R. D., [R] **ologit**  
 McNeil, B. J., [R] **roc**, [R] **rocf**  
 McNeil, D., [R] **poisson**  
 Meeusen, W., [R] **frontier**  
 Mehta, C. R., [R] **exlogistic**, [R] **exlogistic**  
**postestimation**, [R] **expoisson**, [R] **tabulate**  
**twoway**  
 Mensing, R. W., [R] **anova postestimation**  
 Metz, C. E., [R] **logistic postestimation**  
 Michels, K. M., [R] **anova**, [R] **loneway**, [R] **oneway**  
 Mielke, P., [R] **brier**  
 Miller, A. B., [R] **kappa**  
 Miller Jr., R. G., [R] **diagnostic plots**, [R] **oneway**  
 Milliken, G. A., [R] **anova**, [R] **margins**  
 Miranda, A., [R] **heckprob**, [R] **ivprobit**, [R] **ivtobit**,  
 [R] **logistic**, [R] **logit**, [R] **nbreg**, [R] **ologit**,  
 [R] **oprobit**, [R] **poisson**, [R] **probit**  
 Mitchell, C., [R] **exlogistic**  
 Mitchell, M. N., [R] **logistic**, [R] **logistic**  
**postestimation**, [R] **logit**  
 Moffitt, R. A., [R] **tobit**, [R] **tobit postestimation**  
 Monfort, A., [R] **hausman**, [R] **suest**, [R] **test**  
 Monson, R. R., [R] **bitest**  
 Mood, A. M., [R] **centile**  
 Mooney, C. Z., [R] **bootstrap**, [R] **jackknife**  
 Moran, J. L., [R] **dstdize**  
 Morris, C., [R] **bootstrap**  
 Morris, N. F., [R] **binreg**  
 Moskowitz, M., [R] **kappa**  
 Mosteller, F., [R] **jackknife**, [R] **regress**, [R] **regress**  
**postestimation**, [R] **rreg**  
 Moulton, L. H., [R] **permute**  
 Muellbauer, J., [R] **nl**  
 Mullahy, J., [R] **gmm**, [R] **zinb**, [R] **zip**  
 Müller, H. G., [R] **lpoly**  
 Muñoz, J., [R] **exlogistic**  
 Murphy, A. H., [R] **brier**  
 Murray-Lyon, I. M., [R] **binreg**

## N

Nachtsheim, C. J., [R] **pkcross**, [R] **pkequiv**,  
 [R] **pkshape**, [R] **regress postestimation**  
 Nadarajah, S., [R] **nlogit**  
 Nadaraya, E. A., [R] **lpoly**  
 Nagler, J., [R] **scobit**  
 Naiman, D. Q., [R] **qreg**  
 Nannicini, T., [R] **treatreg**  
 Narula, S. C., [R] **qreg**  
 Nee, J. C. M., [R] **kappa**  
 Nelder, J. A., [R] **binreg**, [R] **binreg postestimation**,  
 [R] **glm**, [R] **glm postestimation**, [R] **margins**,  
 [R] **ologit**  
 Nelson, C. R., [R] **ivregress postestimation**  
 Nelson, E. C., [R] **alpha**, [R] **lincom**, [R] **mlogit**,  
 [R] **mprobit**, [R] **mprobit postestimation**,  
 [R] **predictnl**, [R] **slogit**  
 Nelson, F. D., [R] **logit**, [R] **probit**

Neter, J., [R] **pkcross**, [R] **pkequiv**, [R] **pkshape**,  
[R] **regress postestimation**  
Newey, W. K., [R] **glm**, [R] **gmm**, [R] **ivprobit**,  
[R] **ivregress**, [R] **ivtobit**  
Newman, S. C., [R] **poisson**  
Newson, R., [R] **centile**, [R] **glm**, [R] **inequality**,  
[R] **kwallis**, [R] **mkspline**, [R] **ranksum**,  
[R] **signrank**, [R] **spearman**, [R] **tabulate**  
**twoway**  
Newton, H. J., [R] **kdensity**  
Neyman, J., [R] **ci**  
Nicewander, W. A., [R] **correlate**  
Nichols, A., [R] **ivregress**, [R] **reg3**, [R] **treatreg**  
Nickell, S., [R] **gmm**  
Nolan, D., [R] **diagnostic plots**  
Nunnally, J. C., [R] **alpha**

## O

Oehlert, G. W., [R] **nlcom**  
O'Fallon, W. M., [R] **logit**  
Olivier, D., [R] **expoisson**  
Olkin, I., [R] **kwallis**  
Olson, J. M., [R] **symmetry**  
Ord, J. K., [R] **centile**, [R] **mean**, [R] **proportion**,  
[R] **qreg**, [R] **ratio**, [R] **summarize**, [R] **total**  
Ostle, B., [R] **anova postestimation**  
Over, M., [R] **regress**

## P

Pacheco, J. M., [R] **dstdize**  
Pagan, A. R., [R] **frontier**, [R] **mvreg**, [R] **regress**  
**postestimation**, [R] **sureg**  
Pagano, M., [R] **dstdize**, [R] **logistic**, [R] **margins**,  
[R] **sampsi**, [R] **tabulate twoway**  
Paik, M. C., [R] **dstdize**, [R] **kappa**, [R] **sampsi**  
Pampel, F. C., [R] **logistic**, [R] **logit**, [R] **probit**  
Panis, C., [R] **mkspline**  
Park, H. J., [R] **regress**  
Park, J. Y., [R] **boxcox**, [R] **margins**, [R] **nlcom**,  
[R] **predictnl**, [R] **testnl**  
Parks, W. P., [R] **exlogistic**  
Parzen, E., [R] **estat**, [R] **kdensity**  
Pasquini, J., [R] **vwls**  
Patel, N. R., [R] **exlogistic**, [R] **exlogistic**  
**postestimation**, [R] **expoisson**, [R] **tabulate**  
**twoway**  
Patterson, H. D., [R] **pkcross**  
Paul, C., [R] **logistic**  
Pearce, M. S., [R] **logistic**  
Pearson, E. S., [R] **ci**, [R] **ttest**  
Pearson, K., [R] **correlate**, [R] **tabulate twoway**  
Pepe, M. S., [R] **roc**, [R] **rocfit**  
Peracchi, F., [R] **regress**, [R] **regress postestimation**  
Pérez-Hernández, M. A., [R] **kdensity**  
Pérez-Hoyos, S., [R] **lrtest**  
Perkins, A. M., [R] **ranksum**

Perrin, E., [R] **alpha**, [R] **lincom**, [R] **mlogit**,  
[R] **mprobit**, [R] **mprobit postestimation**,  
[R] **predictnl**, [R] **slogit**  
Peterson, B., [R] **ologit**  
Peterson, W. W., [R] **logistic postestimation**  
Petitclerc, M., [R] **kappa**  
Petkova, E., [R] **suest**  
Petrin, A., [R] **frontier**  
Pfeffer, R. I., [R] **symmetry**  
Phillips, P. C. B., [R] **boxcox**, [R] **margins**, [R] **nlcom**,  
[R] **predictnl**, [R] **regress postestimation time**  
**series**, [R] **testnl**  
Pickles, A., [R] **gllamm**, [R] **glm**  
Pike, M. C., [R] **symmetry**  
Pindyck, R. S., [R] **biprobit**, [R] **heckprob**  
Pischke, J.-S., [R] **ivregress**, [R] **ivregress**  
**postestimation**, [R] **qreg**, [R] **regress**  
Pitblado, J. S., [R] **frontier**, [R] **lpoly**, [R] **maximize**,  
[R] **ml**  
Plackett, R. L., [R] **ameans**, [R] **regress**, [R] **rologit**,  
[R] **summarize**, [R] **ttest**  
Plummer Jr., W. D., [R] **sunflower**  
Pocock, S. J., [R] **sampsi**  
Poi, B. P., [R] **bootstrap**, [R] **bstat**, [R] **frontier**,  
[R] **ivregress**, [R] **ivregress postestimation**,  
[R] **nl**, [R] **nlstur**, [R] **reg3**  
Poirier, D. J., [R] **biprobit**  
Poisson, S. D., [R] **poisson**  
Ponce de Leon, A., [R] **roc**  
Porter, T. M., [R] **correlate**  
Powers, D. A., [R] **logit**, [R] **probit**  
Preece, D. A., [R] **ttest**  
Pregibon, D., [R] **glm**, [R] **linktest**, [R] **logistic**,  
[R] **logistic postestimation**, [R] **logit**, [R] **logit**  
**postestimation**  
Press, W. H., [R] **dydx**, [R] **vwls**  
Punj, G. N., [R] **rologit**

## R

Rabe-Hesketh, S., [R] **anova**, [R] **gllamm**, [R] **glm**,  
[R] **heckprob**, [R] **ivprobit**, [R] **ivtobit**,  
[R] **logistic**, [R] **logit**, [R] **nbreg**, [R] **ologit**,  
[R] **oprobit**, [R] **poisson**, [R] **probit**  
Raftery, A., [R] **BIC note**, [R] **estat**, [R] **glm**  
Ramalheira, C., [R] **ameans**  
Ramsey, J. B., [R] **regress postestimation**  
Ratkowsky, D. A., [R] **nl**, [R] **pk**, [R] **pkcross**  
Redelmeier, D. A., [R] **brier**  
Reichenheim, M. E., [R] **kappa**, [R] **roc**  
Reid, C., [R] **ci**  
Reilly, M., [R] **logistic**  
Relles, D. A., [R] **rreg**  
Rencher, A. C., [R] **anova postestimation**  
Revankar, N. S., [R] **frontier**  
Richardson, W., [R] **ttest**  
Riffenburgh, R. H., [R] **ksmirnov**, [R] **kwallis**  
Riley, A. R., [R] **net search**



Rivers, D., [R] **ivprobit**  
 Roberson, P. K., [R] **logistic postestimation**  
 Robyn, D. L., [R] **cumul**  
 Rodgers, J. L., [R] **correlate**  
 Rodríguez, G., [R] **nbreg**, [R] **poisson**, [R] **ztnb**  
 Rogers, W. H., [R] **brier**, [R] **glm**, [R] **heckman**,  
     [R] **lincom**, [R] **mlogit**, [R] **mprobit**,  
     [R] **mprobit postestimation**, [R] **nbreg**,  
     [R] **poisson**, [R] **predictnl**, [R] **qreg**, [R] **regress**,  
     [R] **rreg**, [R] **sktest**, [R] **slogit**, [R] **suest**  
 Ronning, G., [R] **clogit**  
 Rose, J. M., [R] **nlogit**  
 Rosner, B., [R] **sampsi**  
 Ross, G. J. S., [R] **nl**  
 Rothman, K. J., [R] **ci**, [R] **dstdize**, [R] **glogit**,  
     [R] **poisson**  
 Rothstein, H. R., [R] **meta**  
 Rousseeuw, P. J., [R] **qreg**, [R] **regress postestimation**,  
     [R] **rreg**  
 Rovine, M. J., [R] **correlate**  
 Royston, P., [R] **centile**, [R] **cusum**, [R] **diagnostic**  
     **plots**, [R] **dotplot**, [R] **dydx**, [R] **estat**,  
     [R] **fracpoly**, [R] **fracpoly postestimation**,  
     [R] **glm**, [R] **kdensity**, [R] **lnskew0**, [R] **lowess**,  
     [R] **mfp**, [R] **ml**, [R] **nl**, [R] **regress**, [R] **sampsi**,  
     [R] **sktest**, [R] **smooth**, [R] **swilk**  
 Rubin, H., [R] **ivregress postestimation**  
 Rubinfeld, D. L., [R] **biprobit**, [R] **heckprob**  
 Rudebusch, G. D., [R] **ivregress postestimation**  
 Ruppert, D., [R] **boxcox**, [R] **rreg**  
 Rutherford, E., [R] **poisson**  
 Ruud, P. A., [R] **gmm**, [R] **rologit**, [R] **suest**  
 Ryan, T. P., [R] **qc**

## S

Sakamoto, Y., [R] **BIC note**  
 Salgado-Ugarte, I. H., [R] **kdensity**, [R] **lowess**,  
     [R] **smooth**  
 Salim, A., [R] **logistic**  
 Sanders, F., [R] **brier**  
 Santos Silva, J. M. C., [R] **gmm**  
 Sargan, J. D., [R] **ivregress postestimation**  
 Sasieni, P., [R] **dotplot**, [R] **lowess**, [R] **nptrend**,  
     [R] **smooth**  
 Satterthwaite, F. E., [R] **ttest**  
 Sauerbrei, W., [R] **estat**, [R] **fracpoly**, [R] **mfp**  
 Savin, N. E., [R] **regress postestimation time series**  
 Saw, S. L. C., [R] **qc**  
 Sawa, T., [R] **estat**  
 Saxl, I., [R] **correlate**  
 Schaalje, G. B., [R] **anova postestimation**  
 Schaffer, M. E., [R] **ivregress**, [R] **ivregress**  
     **postestimation**  
 Scheffé, H., [R] **anova**, [R] **oneway**  
 Schlesselman, J. J., [R] **boxcox**  
 Schlossmacher, E. J., [R] **qreg**  
 Schmidt, C. H., [R] **brier**  
 Schmidt, P., [R] **frontier**, [R] **regress postestimation**

Schneider, H., [R] **sdtest**  
 Schnell, D., [R] **regress**  
 Schonlau, M., [R] **glm**, [R] **logistic**, [R] **logit**,  
     [R] **poisson**, [R] **regress**  
 Schuurmann, D. J., [R] **pkequiv**  
 Schwarz, G., [R] **BIC note**, [R] **estat**  
 Scott, D. W., [R] **kdensity**  
 Scott, G. B., [R] **exlogistic**  
 Scotto, M. G., [R] **diagnostic plots**  
 Searle, S. R., [R] **margins**  
 Seed, P. T., [R] **ci**, [R] **correlate**, [R] **logistic**  
     **postestimation**, [R] **roc**, [R] **sampsi**, [R] **sdtest**,  
     [R] **spearman**  
 Seidler, J., [R] **correlate**  
 Selvin, S., [R] **poisson**  
 Sempas, C. T., [R] **dstdize**  
 Semykina, A., [R] **inequality**, [R] **qreg**  
 Seneta, E., [R] **correlate**  
 Senn, S. J., [R] **glm**, [R] **ttest**  
 Shapiro, S. S., [R] **swilk**  
 Shavelson, R. J., [R] **alpha**  
 Shea, J., [R] **ivregress postestimation**  
 Sheather, S. J., [R] **lpoly**  
 Sheldon, T. A., [R] **meta**  
 Shewhart, W. A., [R] **qc**  
 Shiboski, S. C., [R] **logistic**  
 Shiller, R. J., [R] **tobit**  
 Shimizu, M., [R] **kdensity**, [R] **lowess**  
 Shrout, P. E., [R] **kappa**  
 Šidák, Z., [R] **oneway**  
 Silverman, B. W., [R] **kdensity**  
 Silvey, S. D., [R] **ologit**, [R] **oprobit**  
 Simonoff, J. S., [R] **kdensity**, [R] **ztnb**, [R] **ztp**  
 Simor, I. S., [R] **kappa**  
 Singleton, K. J., [R] **gmm**  
 Skrandal, A., [R] **gllamm**, [R] **glm**  
 Smeeton, N. C., [R] **ranksum**, [R] **signrank**  
 Smirnov, N. V., [R] **ksmirnov**  
 Smith, H., [R] **eivreg**, [R] **oneway**, [R] **regress**,  
     [R] **stepwise**  
 Smith, J. M., [R] **fracpoly**  
 Smith, R. J., [R] **ivprobit**  
 Snedecor, G. W., [R] **ameans**, [R] **anova**, [R] **correlate**,  
     [R] **oneway**, [R] **ranksum**, [R] **signrank**  
 Snell, E. J., [R] **exlogistic**, [R] **expoisson**  
 Song, F., [R] **meta**  
 Soon, T. W., [R] **qc**  
 Spearman, C., [R] **spearman**  
 Speed, F. M., [R] **margins**  
 Speed, T., [R] **diagnostic plots**  
 Spiegelhalter, D. J., [R] **brier**  
 Spiedman, R. S., [R] **symmetry**  
 Spitzer, J. J., [R] **boxcox**  
 Sprent, P., [R] **ranksum**, [R] **signrank**  
 Sribney, W. M., [R] **frontier**, [R] **maximize**, [R] **ml**,  
     [R] **orthog**, [R] **ranksum**, [R] **signrank**,  
     [R] **stepwise**, [R] **test**

Staelin, R., [R] **rologit**  
 Staiger, D., [R] **ivregress postestimation**  
 Starmer, C. F., [R] **vwls**  
 Startz, R., [R] **ivregress postestimation**  
 Stegun, I. A., [R] **orthog**  
 Steichen, T. J., [R] **kappa**, [R] **kdensity**, [R] **sunflower**  
 Steiger, W., [R] **qreg**  
 Stein, C., [R] **bootstrap**  
 Stephenson, D. B., [R] **brier**  
 Stepniewska, K. A., [R] **nptrend**  
 Sterne, J. A. C., [R] **dstdize**, [R] **meta**, [R] **summarize**  
 Stevenson, R. E., [R] **frontier**  
 Stewart, M. B., [R] **intreg**, [R] **oprobit**, [R] **tobit**  
 Stigler, S. M., [R] **ameans**, [R] **ci**, [R] **correlate**,  
 [R] **kwallis**, [R] **qreg**, [R] **regress**  
 Stillman, S., [R] **ivregress**, [R] **ivregress postestimation**  
 Stine, R., [R] **bootstrap**  
 Stock, J. H., [R] **ivregress**, [R] **ivregress postestimation**  
 Stoto, M. A., [R] **lv**  
 Street, J. O., [R] **rreg**  
 Stryhn, H., [R] **regress**  
 Stuart, A., [R] **centile**, [R] **mean**, [R] **proportion**,  
 [R] **qreg**, [R] **ratio**, [R] **summarize**,  
 [R] **symmetry**, [R] **tobit**, [R] **total**  
 Stuetzle, W., [R] **sunflower**  
 Sullivan, G., [R] **regress**  
 Sutton, A. J., [R] **meta**  
 Swed, F. S., [R] **runtest**  
 Swets, J. A., [R] **logistic postestimation**  
 Szroeter, J., [R] **regress postestimation**

## T

Taka, M. T., [R] **pkcross**  
 Tamhane, A. C., [R] **oneway**, [R] **sampsi**  
 Taniuchi, T., [R] **kdensity**  
 Tanner Jr., W. P., [R] **logistic postestimation**  
 Tanur, J. M., [R] **kwallis**  
 Tapia, R. A., [R] **kdensity**  
 Tarlov, A. R., [R] **alpha**, [R] **lincom**, [R] **mlogit**,  
 [R] **mprobit**, [R] **mprobit postestimation**,  
 [R] **predictnl**, [R] **slogit**  
 Taylor, C., [R] **gglamm**, [R] **glm**  
 Teukolsky, S. A., [R] **dydx**, [R] **vwls**  
 Theil, H., [R] **ivregress**, [R] **reg3**  
 Thiele, T. N., [R] **summarize**  
 Thompson, J. C., [R] **diagnostic plots**  
 Thompson, J. R., [R] **kdensity**  
 Thorndike, F., [R] **poisson**  
 Thurstone, L. L., [R] **rologit**  
 Tibshirani, R. J., [R] **bootstrap**, [R] **qreg**  
 Tidmarsh, C. E., [R] **fracpoly**  
 Tilford, J. M., [R] **logistic postestimation**  
 Tobias, A., [R] **alpha**, [R] **logistic postestimation**,  
 [R] **lrtest**, [R] **poisson**, [R] **roc**, [R] **sdtest**  
 Tobin, J., [R] **tobit**  
 Toman, R. J., [R] **stepwise**  
 Toplis, P. J., [R] **binreg**

Tosetto, A., [R] **logistic**, [R] **logit**  
 Train, K. E., [R] **asmprobit**  
 Trapido, E., [R] **exlogistic**  
 Treiman, D. J., [R] **eivreg**, [R] **mlogit**  
 Trivedi, P. K., [R] **asclogit**, [R] **asmprobit**,  
 [R] **bootstrap**, [R] **gmm**, [R] **heckman**,  
 [R] **intreg**, [R] **ivregress**, [R] **ivregress**  
**postestimation**, [R] **logit**, [R] **mprobit**,  
 [R] **nbgreg**, [R] **ologit**, [R] **oprobit**, [R] **poisson**,  
 [R] **probit**, [R] **qreg**, [R] **regress**, [R] **regress**  
**postestimation**, [R] **simulate**, [R] **sureg**,  
 [R] **tobit**, [R] **ztnb**, [R] **ztp**  
 Tsiatis, A. A., [R] **exlogistic**  
 Tufte, E. R., [R] **stem**  
 Tukey, J. W., [R] **jackknife**, [R] **ladder**, [R] **linktest**,  
 [R] **lv**, [R] **regress**, [R] **regress postestimation**,  
 [R] **rreg**, [R] **smooth**, [R] **spikeplot**, [R] **stem**  
 Tukey, P. A., [R] **diagnostic plots**, [R] **lowess**  
 Tyler, J. H., [R] **regress**

## U

Uebersax, J. S., [R] **tetrachoric**  
 Uhlendorff, A., [R] **asmprobit**, [R] **mlogit**, [R] **mprobit**  
 University Group Diabetes Program, [R] **glogit**  
 Utts, J. M., [R] **ci**

## V

Valman, H. B., [R] **fracpoly**  
 van Belle, G., [R] **anova**, [R] **dstdize**, [R] **oneway**  
 Van de Ven, W. P. M. M., [R] **biprobbit**, [R] **heckprob**  
 van den Broeck, J., [R] **frontier**  
 Van Kerm, P., [R] **inequality**, [R] **kdensity**  
 Van Loan, C. F., [R] **orthog**, [R] **tetrachoric**  
 Van Pragg, B. M. S., [R] **biprobbit**, [R] **heckprob**  
 Velleman, P. F., [R] **regress postestimation**, [R] **smooth**  
 Vetterling, W. T., [R] **dydx**, [R] **vwls**  
 Vidmar, S., [R] **ameans**  
 Vittinghoff, E., [R] **logistic**  
 von Bortkewitsch, L., [R] **poisson**  
 von Eye, A., [R] **correlate**  
 Von Storch, H., [R] **brier**  
 Vondráček, J., [R] **correlate**  
 Vuong, Q. H., [R] **ivprobit**, [R] **zinb**, [R] **zip**

## W

Wacholder, S., [R] **binreg**  
 Wagner, H. M., [R] **qreg**  
 Wallis, W. A., [R] **kwallis**  
 Walters, S. J., [R] **ci**, [R] **kappa**, [R] **tabulate twoway**  
 Wand, M. P., [R] **kdensity**  
 Wang, D., [R] **ci**, [R] **dstdize**, [R] **prtest**  
 Wang, Y., [R] **asmprobit**  
 Wang, Z., [R] **logistic postestimation**, [R] **lrtest**,  
 [R] **stepwise**

Ware Jr., J. E., [R] [alpha](#), [R] [lincom](#), [R] [mlogit](#),  
[R] [mprobit](#), [R] [mprobit postestimation](#),  
[R] [predictnl](#), [R] [slogit](#)

Waterson, E. J., [R] [binreg](#)

Watson, G. S., [R] [lpoly](#), [R] [regress postestimation time series](#)

Watson, M. W., [R] [ivregress](#)

Webster, A. D., [R] [fracpoly](#)

Wedderburn, R. W. M., [R] [glm](#)

Weesie, J., [R] [alpha](#), [R] [constraint](#), [R] [hausman](#),  
[R] [ladder](#), [R] [logistic postestimation](#), [R] [reg3](#),  
[R] [regress](#), [R] [regress postestimation](#),  
[R] [rologit](#), [R] [simulate](#), [R] [suest](#), [R] [sureg](#),  
[R] [tabstat](#), [R] [tabulate twoway](#), [R] [test](#),  
[R] [tetrachoric](#)

Weisberg, H. F., [R] [summarize](#)

Weisberg, S., [R] [boxcox](#), [R] [regress](#), [R] [regress postestimation](#)

Welch, B. L., [R] [ttest](#)

Wellington, J. F., [R] [qreg](#)

Wells, K. B., [R] [lincom](#), [R] [mlogit](#), [R] [mprobit](#),  
[R] [mprobit postestimation](#), [R] [predictnl](#),  
[R] [slogit](#)

Welsch, R. E., [R] [estat](#), [R] [regress postestimation](#)

West, K. D., [R] [glm](#), [R] [gmm](#), [R] [ivregress](#)

West, S. G., [R] [pcorr](#)

Westlake, W. J., [R] [pkequiv](#)

White, H., [R] [regress](#), [R] [regress postestimation](#),  
[R] [suest](#)

White, K. J., [R] [boxcox](#), [R] [regress postestimation time series](#)

Whitehouse, E., [R] [inequality](#)

Whitney, D. R., [R] [kwallis](#), [R] [ranksum](#)

Wiggins, V. L., [R] [regress postestimation](#), [R] [regress postestimation time series](#)

Wilcox, D. W., [R] [ivregress postestimation](#)

Wilcoxon, F., [R] [kwallis](#), [R] [ranksum](#), [R] [signrank](#)

Wilde, J., [R] [gmm](#)

Wilks, M. B., [R] [cumul](#), [R] [diagnostic plots](#), [R] [swilk](#)

Wilks, D. S., [R] [brier](#)

Williams, R., [R] [ologit](#), [R] [pcorr](#), [R] [stepwise](#)

Wilson, E. B., [R] [ci](#)

Wilson, S. R., [R] [bootstrap](#)

Windmeijer, F., [R] [gmm](#)

Winer, B. J., [R] [anova](#), [R] [loneway](#), [R] [oneway](#)

Wolfe, F., [R] [correlate](#), [R] [spearman](#)

Wolfe, R., [R] [ologit](#), [R] [oprobit](#), [R] [tabulate twoway](#)

Wolfson, C., [R] [kappa](#)

Wolpin, K. I., [R] [asmprobit](#)

Wood, F. S., [R] [diagnostic plots](#)

Wooldridge, J. M., [R] [gmm](#), [R] [intreg](#), [R] [ivprobit](#),  
[R] [ivregress](#), [R] [ivregress postestimation](#),  
[R] [ivtobit](#), [R] [margins](#), [R] [regress](#), [R] [regress postestimation](#), [R] [regress postestimation time series](#), [R] [tobit](#)

Working, H., [R] [roc](#), [R] [rocfits](#)

Wright, J. H., [R] [ivregress](#), [R] [ivregress postestimation](#)

Wright, J. T., [R] [binreg](#)

Wright, P. G., [R] [ivregress](#)

Wu, C. F. J., [R] [qreg](#)

Wu, D.-M., [R] [ivregress postestimation](#)

## X

Xie, Y., [R] [logit](#), [R] [probit](#)

Xu, J., [R] [cloglog](#), [R] [logistic](#), [R] [logit](#), [R] [mlogit](#),  
[R] [ologit](#), [R] [oprobit](#), [R] [probit](#)

## Y

Yates, J. F., [R] [brier](#)

Yee, T. W., [R] [slogit](#)

Yellott Jr., J. I., [R] [rologit](#)

Yen, W. M., [R] [alpha](#)

Yogo, M., [R] [ivregress](#), [R] [ivregress postestimation](#)

## Z

Zabell, S., [R] [kwallis](#)

Zavoina, W., [R] [ologit](#)

Zelen, M., [R] [ttest](#)

Zellner, A., [R] [frontier](#), [R] [nlstur](#), [R] [reg3](#), [R] [sureg](#)

Zelterman, D., [R] [tabulate twoway](#)

Zimmerman, F., [R] [regress](#)

Zubkoff, M., [R] [alpha](#), [R] [lincom](#), [R] [mlogit](#),  
[R] [mprobit](#), [R] [mprobit postestimation](#),  
[R] [predictnl](#), [R] [slogit](#)

Zwiers, F. W., [R] [brier](#)



# Subject index

This is the subject index for the 3-volume *Base Reference Manual*. Readers may also want to consult the [combined subject index](#) (and the [combined author index](#)) in the *Quick Reference and Index*. The combined index indexes the *Getting Started* manuals, the *User's Guide*, and all the reference manuals except the *Mata Reference Manual*.

Semicolons set off the most important entries for the rest. Sometimes no entry will be set off with semicolons, meaning that all entries are equally important.

## A

about command, [R] [about](#)  
 absorption in regression, [R] [areg](#)  
 acprplot command, [R] [regress postestimation](#)  
 added-variable plots, [R] [regress postestimation](#)  
 adjusted margins, [R] [margins](#)  
 adjusted means, [R] [margins](#)  
 adjusted partial residual plot, [R] [regress postestimation](#)  
 ado, update subcommand, [R] [update](#)  
 ado, view subcommand, [R] [view](#)  
 ado command, [R] [net](#)  
 ado\_d, view subcommand, [R] [view](#)  
 ado describe command, [R] [net](#)  
 ado dir command, [R] [net](#)  
 ado-files,  
     editing, [R] [doedit](#)  
     installing, [R] [net](#), [R] [sj](#), [R] [ssc](#)  
     location of, [R] [which](#)  
     official, [R] [update](#)  
 adosize, set subcommand, [R] [set](#)  
 ado uninstall command, [R] [net](#)  
 adoupdate command, [R] [adoupdate](#)  
 agreement, interrater, [R] [kappa](#)  
 AIC, [R] [BIC note](#), [R] [estat](#), [R] [estimates stats](#), [R] [glm](#)  
 Akaike information criterion, see AIC  
 all, update subcommand, [R] [update](#)  
 alpha coefficient, Cronbach's, [R] [alpha](#)  
 alpha command, [R] [alpha](#)  
 alternative-specific multinomial probit regression, [R] [asmprobit](#)  
 alternatives, estat subcommand, [R] [asclogit postestimation](#), [R] [asmprobit postestimation](#), [R] [asroprobit postestimation](#)  
 amean command, [R] [ameans](#)  
 analysis of covariance, see ANCOVA  
 analysis of variance, see ANOVA  
 analysis-of-variance test of normality, [R] [swilk](#)  
 ANCOVA, [R] [anova](#)  
 ANOVA, [R] [anova](#), [R] [loneaway](#), [R] [oneway](#)  
     Kruskal–Wallis, [R] [kwallis](#)  
     repeated measures, [R] [anova](#)  
 anova command, [R] [anova](#), [R] [anova postestimation](#), also see [postestimation](#) command

ARCH effects, testing for, [R] [regress postestimation time series](#)  
 archlm, estat subcommand, [R] [regress postestimation time series](#)  
 areg command, [R] [areg](#), [R] [areg postestimation](#), also see [postestimation](#) command  
 asclogit command, [R] [asclogit](#), [R] [asclogit postestimation](#), also see [postestimation](#) command  
 asmprobit command, [R] [asmprobit](#), [R] [asmprobit postestimation](#), also see [postestimation](#) command  
 asroprobit command, [R] [asroprobit](#), [R] [asroprobit postestimation](#), also see [postestimation](#) command  
 association, measures of, [R] [tabulate twoway](#)  
 asymmetry, [R] [lnskew0](#), [R] [lv](#), [R] [sktest](#), [R] [summarize](#)  
 attribute tables, [R] [table](#), [R] [tabulate](#), [summarize\(\)](#), [R] [tabulate twoway](#)  
 AUC, [R] [pk](#)  
 augmented  
     component-plus-residual plot, [R] [regress postestimation](#)  
     partial residual plot, [R] [regress postestimation](#)  
 autocorrelation, [R] [regress postestimation time series](#)  
 autoregressive conditional heteroskedasticity, testing for, [R] [regress postestimation time series](#)  
 autotabgraphs, set subcommand, [R] [set](#)  
 average marginal effects, [R] [margins](#)  
 average partial effects (APEs), [R] [margins](#)  
 average predictions, [R] [margins](#)  
 averages, see [means](#)  
 avplot and avplots commands, [R] [regress postestimation](#)

## B

Bartlett's test for equal variances, [R] [oneway](#)  
 Bayesian information criterion, see BIC  
 bcskev0 command, [R] [lnskew0](#)  
 Berndt–Hall–Hall–Hausman algorithm, [R] [ml](#)  
 beta coefficients, [R] [gmm](#), [R] [ivregress](#), [R] [nlshr](#), [R] [regress](#), [R] [regress postestimation](#)  
 BFGS algorithm, [R] [ml](#)  
 bgodfrey, estat subcommand, [R] [regress postestimation time series](#)  
 BHHH algorithm, [R] [ml](#)  
 bias corrected and accelerated, [R] [bootstrap postestimation](#)  
 BIC, [R] [BIC note](#), [R] [estat](#), [R] [estimates stats](#), [R] [glm](#)  
 binary outcome models, see [dichotomous outcome models](#)  
 binomial  
     distribution, confidence intervals, [R] [ci](#)  
     family regression, [R] [binreg](#)  
     probability test, [R] [bittest](#)  
 binreg command, [R] [binreg](#), [R] [binreg postestimation](#), also see [postestimation](#) command  
 bioequivalence tests, [R] [pk](#), [R] [pkequiv](#)  
 biopharmaceutical data, see [pk](#) (pharmacokinetic data)

biprobit command, [R] **biprobit**, [R] **biprobit postestimation**, also see postestimation command  
 bitest and bitesti commands, [R] **bitest**  
 bivariate probit regression, [R] **biprobit**  
 biweight regression estimates, [R] **rreg**  
 blogit command, [R] **glogit**, [R] **glogit postestimation**, also see postestimation command  
 Bonferroni  
   adjustment, [R] **correlate**, [R] **spearman**, [R] **test**, [R] **testnl**  
   multiple comparison test, [R] **anova postestimation**, [R] **oneway**, [R] **regress postestimation**  
 bootstrap, estat subcommand, [R] **bootstrap postestimation**  
 bootstrap  
   sampling and estimation, [R] **bootstrap**, [R] **bsample**, [R] **bstat**, [R] **qreg**, [R] **simulate**  
   standard errors, [R] **vce\_option**  
 bootstrap prefix command, [R] **bootstrap**, [R] **bootstrap postestimation**, also see postestimation command  
 Boston College archive, see SSC archive  
 Box–Cox  
   power transformations, [R] **lnskew0**  
   regression, [R] **boxcox**  
 boxcox command, [R] **boxcox**, [R] **boxcox postestimation**, also see postestimation command  
 Box’s conservative epsilon, [R] **anova**  
 bprobit command, [R] **glogit**, [R] **glogit postestimation**, also see postestimation command  
 Breusch–Godfrey test, [R] **regress postestimation time series**  
 Breusch–Pagan test of independence, [R] **mvreg**, [R] **sureg**  
 brier command, [R] **brier**  
 Brier score decomposition, [R] **brier**  
 browse, view subcommand, [R] **view**  
 Broyden–Fletcher–Goldfarb–Shanno algorithm, [R] **ml**  
 bsample command, [R] **bsample**  
 bsqreg command, [R] **qreg**, [R] **qreg postestimation**, also see postestimation command  
 bstat command, [R] **bstat**

## C

c(seed) c-class value, [R] **set emptycells**, [R] **set seed**  
 calculator, [R] **display**  
 case–control data, [R] **clogit**, [R] **logistic**, [R] **symmetry**  
 categorical data  
   agreement, measures for, [R] **kappa**  
 cchart command, [R] **qc**  
 cd, net subcommand, [R] **net**  
 censored-normal regression, see interval regression  
 centile command, [R] **centile**  
 centiles, see percentiles, displaying  
 central tendency, measures of, [R] **summarize**; [R] **ameans**, [R] **lv**, [R] **mean**  
 check, ml subcommand, [R] **ml**

checksum, set subcommand, [R] **set**  
 chelp command, [R] **help**  
 chi-squared  
   hypothesis test, [R] **sdtest**, [R] **test**, [R] **testnl**  
   probability plot, [R] **diagnostic plots**  
   test of independence, [R] **tabulate twoway**  
 choice models, [R] **asclogit**, [R] **asmprobit**, [R] **asprobit**, [R] **clogit**, [R] **cloglog**, [R] **exlogistic**, [R] **glogit**, [R] **heckprob**, [R] **hetprob**, [R] **ivprobit**, [R] **logistic**, [R] **logit**, [R] **mlogit**, [R] **mprobit**, [R] **nlogit**, [R] **ologit**, [R] **oprobit**, [R] **probit**, [R] **rologit**, [R] **scobit**, [R] **slogit**  
 Chow test, [R] **anova**  
 ci and cii commands, [R] **ci**  
 classification, estat subcommand, [R] **logistic postestimation**  
 clear,  
   estimates subcommand, [R] **estimates store**  
   ml subcommand, [R] **ml**  
 clearing estimation results, [R] **estimates store**  
 clogit command, [R] **clogit**, [R] **clogit postestimation**, [R] **mlogit**, [R] **rologit**, also see postestimation command  
 cloglog command, [R] **cloglog**, [R] **cloglog postestimation**, also see postestimation command  
 cluster estimator of variance, [R] **vce\_option**  
   alternative-specific conditional logit model, [R] **asclogit**  
   alternative-specific multinomial probit regression, [R] **asmprobit**  
   alternative-specific rank-ordered probit regression, [R] **asprobit**  
   bivariate probit regression, [R] **biprobit**  
   complementary log-log regression, [R] **cloglog**  
   conditional logistic regression, [R] **clogit**  
   constrained linear regression, [R] **cnsreg**  
   estimate mean, [R] **mean**  
   estimate proportions, [R] **proportion**  
   estimate ratios, [R] **ratio**  
   estimate totals, [R] **total**  
   generalized linear models, [R] **glm**  
   generalized linear models for binomial family, [R] **binreg**  
   generalized methods of moments, [R] **gmm**  
   heckman selection model, [R] **heckman**  
   heteroskedastic probit model, [R] **hetprob**  
   instrumental-variables regression, [R] **ivregress**  
   interval regression, [R] **intreg**  
   linear regression, [R] **regress**  
   linear regression with dummy-variable set, [R] **areg**  
   logistic regression, [R] **logistic**, [R] **logit**  
   logit and probit estimation for grouped data, [R] **glogit**  
   maximum likelihood estimation, [R] **ml**  
   multinomial logistic regression, [R] **mlogit**  
   multinomial probit regression, [R] **mprobit**  
   negative binomial regression, [R] **nbreg**

- cluster estimator of variance, *continued*
  - nested logit regression, [R] **nlogit**
  - nonlinear least-squares estimation, [R] **nl**
  - nonlinear systems of equations, [R] **nlshr**
  - ordered logistic regression, [R] **ologit**
  - ordered probit regression, [R] **oprobit**
  - Poisson regression, [R] **poisson**
  - probit model with endogenous regressors, [R] **ivprobit**
  - probit model with sample selection, [R] **heckprob**
  - probit regression, [R] **probit**
  - rank-ordered logistic regression, [R] **rologit**
  - skewed logistic regression, [R] **scobit**
  - stereotype logistic regression, [R] **slogit**
  - tobit model with endogenous regressors, [R] **ivtobit**
  - treatment-effects model, [R] **treatreg**
  - truncated regression, [R] **truncreg**
  - zero-inflated negative binomial regression, [R] **zinb**
  - zero-inflated Poisson regression, [R] **zip**
  - zero-truncated negative binomial regression, [R] **ztnb**
  - zero-truncated Poisson regression, [R] **ztp**
- cluster sampling, [R] **bootstrap**, [R] **bsample**, [R] **gmm**, [R] **jackknife**
- cmdlog command, [R] **log**
- cnsreg command, [R] **cnsreg**, [R] **cnsreg**
  - postestimation**, *also see* postestimation command
- coefficient
  - alpha, [R] **alpha**
  - of variation, [R] **tabstat**
- coefficients (from estimation),
  - accessing, [R] **estimates store**
  - estimated linear combinations of, *see* linear combinations of estimators
  - testing equality of, [R] **test**, [R] **testnl**
- collinearity, handling by **regress**, [R] **regress**
- command line, launching dialog box from, [R] **db**
- commands, reviewing, [R] **#review**
- comparative scatterplot, [R] **dotplot**
- complementary log-log regression, [R] **cloglog**, [R] **glm**
- completely determined outcomes, [R] **logit**
- component-plus-residual plot, [R] **regress**
  - postestimation**
- conditional logistic regression, [R] **asclogit**, [R] **clogit**, [R] **rologit**, [R] **slogit**
- conditional marginal effects, [R] **margins**
- conditional margins, [R] **margins**
- confidence interval, set default, [R] **level**
- confidence intervals
  - for bootstrap statistics, [R] **bootstrap postestimation**
  - for linear combinations of coefficients, [R] **lincom**
  - for means, proportions, and counts, [R] **ci**, [R] **mean**, [R] **proportion**, [R] **ttest**
  - for medians and percentiles, [R] **centile**
  - for nonlinear combinations of coefficients, [R] **nlcom**
  - for odds and risk ratios, [R] **lincom**, [R] **nlcom**
  - for ratios, [R] **ratio**
  - for totals, [R] **total**
- conjoint analysis, [R] **rologit**
- conren, set subcommand, [R] **set**
- console, controlling scrolling of output, [R] **more**
- constrained estimation, [R] **constraint**, [R] **estimation options**
  - alternative-specific multinomial probit regression, [R] **asmprobit**
  - bivariate probit regression, [R] **biprobit**
  - complementary log-log regression, [R] **cloglog**
  - conditional logistic regression, [R] **asclogit**, [R] **clogit**
  - generalized negative binomial regression, [R] **nbreg**
  - heckman selection model, [R] **heckman**
  - heteroskedastic probit model, [R] **hetprobit**
  - interval regression, [R] **intreg**
  - linear regression, [R] **cnsreg**
  - logistic regression, [R] **logistic**, [R] **logit**
  - logit and probit estimation for grouped data, [R] **glogit**
  - multinomial logistic regression, [R] **mlogit**
  - multinomial probit regression, [R] **mprobit**
  - multivariate regression, [R] **mvreg**
  - negative binomial regression, [R] **nbreg**
  - nested logit regression, [R] **nlogit**
  - ordered logistic regression, [R] **ologit**
  - ordered probit regression, [R] **oprobit**
  - Poisson regression, [R] **poisson**
  - probit model with
    - endogenous regressors, [R] **ivprobit**
    - selection, [R] **heckprob**
  - probit regression, [R] **probit**
  - seemingly unrelated regression, [R] **sureg**
  - stereotype logistic regression, [R] **slogit**
  - stochastic frontier models, [R] **frontier**
  - three-stage least squares, [R] **reg3**
  - tobit model with endogenous regressors, [R] **ivtobit**
  - treatment-effects model, [R] **treatreg**
  - zero-inflated negative binomial regression, [R] **zinb**
  - zero-inflated Poisson regression, [R] **zip**
  - zero-truncated negative binomial regression, [R] **ztnb**
  - zero-truncated Poisson regression, [R] **ztp**
- constraint command, [R] **constraint**
- contingency tables, [R] **table**, [R] **tabulate**
- control charts, [R] **qc**
- convergence criteria, [R] **maximize**
- Cook–Weisberg test for heteroskedasticity, [R] **regress**
  - postestimation**
- Cook’s *D*, [R] **predict**, [R] **regress postestimation**
- copy, **ssc** subcommand, [R] **ssc**
- copycolor, set subcommand, [R] **set**
- copyright
  - lapack, [R] **copyright lapack**
  - scintilla, [R] **copyright scintilla**
  - tft2pt1, [R] **copyright tft2pt1**
- copyright command, [R] **copyright**
- correlate command, [R] **correlate**



correlated errors, see robust, Huber/White/sandwich estimator of variance

correlation, [R] **correlate**

for binary variables, [R] **tetrachoric**

intracluster, [R] **loneway**

Kendall's rank, [R] **spearman**

matrices, [R] **correlate**, [R] **estat**

pairwise, [R] **correlate**

partial, [R] **pcorr**

Pearson's product-moment, [R] **correlate**

Spearman's rank, [R] **spearman**

tetrachoric, [R] **tetrachoric**

correlation, **estat** subcommand, [R] **asmpb**  
**postestimation**, [R] **asroprobit postestimation**

cost frontier model, [R] **frontier**

count, **ml** subcommand, [R] **ml**

count data, [R] **expoisson**, [R] **glm**, [R] **nbreg**,  
[R] **poisson**, [R] **zinb**, [R] **zip**, [R] **ztnb**, [R] **ztp**

covariance, **estat** subcommand, [R] **asmpb**  
**postestimation**, [R] **asroprobit postestimation**

covariance matrix of estimators, [R] **correlate**,  
[R] **estat**, [R] **estimates store**

covariate patterns, [R] **logistic postestimation**, [R] **logit**  
**postestimation**, [R] **probit postestimation**

COVRATIO, [R] **regress postestimation**

cprplot command, [R] **regress postestimation**

Cramér's *V*, [R] **tabulate twoway**

Cronbach's alpha, [R] **alpha**

crossover designs, [R] **pk**, [R] **pkcross**, [R] **pkshape**

cross-tabulations, see tables

**cumul** command, [R] **cumul**

cumulative distribution, empirical, [R] **cumul**

cumulative incidence data, [R] **poisson**

**cusum** command, [R] **cusum**

## D

data,

autocorrelated, see autocorrelation

case-control, see case-control data

categorical, see categorical data

large, dealing with, see memory, **set** subcommand

matched case-control, see matched case-control data

range of, see range of data

ranking, see ranking data

sampling, see sampling

summarizing, see summarizing data

survival-time, see survival analysis

time-series, see time-series analysis

data manipulation, [R] **fvvar**, [R] **fvset**

Davidon-Fletcher-Powell algorithm, [R] **ml**

**db** command, [R] **db**

DBETAs, [R] **regress postestimation**

default settings of system parameters, [R] **query**,  
[R] **set\_defaults**

define, **transmap** subcommand, [R] **translate**

delta beta influence statistic, [R] **logistic**  
**postestimation**, [R] **logit postestimation**,  
[R] **regress postestimation**

delta chi-squared influence statistic, [R] **logistic**  
**postestimation**, [R] **logit postestimation**

delta deviance influence statistic, [R] **logistic**  
**postestimation**, [R] **logit postestimation**

delta method, [R] **nlcom**, [R] **predictnl**, [R] **testnl**

density-distribution sunflower plot, [R] **sunflower**

density estimation, kernel, [R] **kdensity**

derivatives, numeric, [R] **dydx**, [R] **testnl**

describe,

**ado** subcommand, [R] **net**

**estimates** subcommand, [R] **estimates describe**

**net** subcommand, [R] **net**

**ssc** subcommand, [R] **ssc**

descriptive statistics, displaying, [R] **summarize**,  
[R] **table**, [R] **tabstat**, [R] **tabulate**,  
**summarize()**; [R] **ameans**, [R] **lv**, [R] **mean**,  
[R] **proportion**, [R] **ratio**, [R] **total**

design effects, [R] **loneway**

deviance residual, [R] **binreg postestimation**,  
[R] **fracpoly postestimation**, [R] **glm**  
**postestimation**, [R] **logistic postestimation**,  
[R] **logit postestimation**, [R] **mfp postestimation**

**dfbeta** command, [R] **regress postestimation**

DFBETAs, [R] **regress postestimation**

DFITS, [R] **regress postestimation**

DFF algorithm, [R] **ml**

diagnostic plots, [R] **diagnostic plots**, [R] **logistic**  
**postestimation**, [R] **regress postestimation**

diagnostics, regression, [R] **regress postestimation**

dialog box, [R] **db**

dichotomous outcome models, [R] **exlogistic**,  
[R] **logistic**; [R] **aslogit**, [R] **binreg**,  
[R] **biprobit**, [R] **brier**, [R] **clogit**, [R] **cloglog**,  
[R] **cusum**, [R] **glm**, [R] **glogit**, [R] **heckprob**,  
[R] **hetprob**, [R] **ivprobit**, [R] **logit**, [R] **probit**,  
[R] **rocfit**, [R] **scobit**

difference of estimated coefficients, see linear combinations of estimators

difficult option, [R] **maximize**

**dir**,

**ado** subcommand, [R] **net**

**estimates** subcommand, [R] **estimates store**

direct standardization, [R] **dstdize**, [R] **mean**,  
[R] **proportion**, [R] **ratio**

dispersion, measures of, [R] **summarize**, [R] **table**;  
[R] **centile**, [R] **lv**

**display**, **ml** subcommand, [R] **ml**

**display** command, as a calculator, [R] **display**

display saved results, [R] **saved results**

display width and length, [R] **log**

displaying, *also* see printing, logs (output)  
previously typed lines, [R] **#review**

saved results, [R] **saved results**

distributions,

diagnostic plots, [R] **diagnostic plots**



distributions, *continued*

examining, [R] **ameans**, [R] **centile**, [R] **kdensity**,  
[R] **lv**, [R] **mean**, [R] **stem**, [R] **summarize**,  
[R] **total**

testing equality of, [R] **ksmirnov**, [R] **kwallis**,  
[R] **ranksum**, [R] **signrank**

testing for normality, [R] **sktest**, [R] **swilk**

do command, [R] **do**

dockable, set subcommand, [R] **set**

dockingguides, set subcommand, [R] **set**

documentation, keyword search on, [R] **search**

doedit command, [R] **doedit**

do-files, [R] **do**

editing, [R] **doedit**

domain sampling, [R] **alpha**

dose-response models, [R] **binreg**, [R] **glm**, [R] **logistic**

dotplot command, [R] **dotplot**

doublebuffer, set subcommand, [R] **set**

dp, set subcommand, [R] **set**

drop, estimates subcommand, [R] **estimates store**

dstdize command, [R] **dstdize**

dummy variables, see indicator variables

Durbin-Watson statistic, [R] **regress postestimation time series**

durbinalt, estat subcommand, [R] **regress postestimation time series**

Durbin's alternative test, [R] **regress postestimation time series**

dwatson, estat subcommand, [R] **regress postestimation time series**

dydx command, [R] **dydx**

## E

e() scalars, macros, matrices, functions, [R] **saved results**

e(sample), resetting, [R] **estimates save**

e-class command, [R] **saved results**

editing

ado-files and do-files, [R] **doedit**

files while in Stata, [R] **doedit**

eform\_option, [R] **eform\_option**

eivreg command, [R] **eivreg**, [R] **eivreg**

**postestimation**, also see **postestimation command**

empirical cumulative distribution function, [R] **cumul**

emptycells, set subcommand, [R] **set emptycells**;  
[R] **set**

ending a Stata session, [R] **exit**

endless loop, see loop, endless

endogenous, estat subcommand, [R] **ivregress postestimation**

endogenous

regressors, [R] **gmm**, [R] **ivprobit**, [R] **ivregress**,  
[R] **ivtobit**

treatment, [R] **treatreg**

variables, [R] **gmm**, [R] **ivprobit**, [R] **ivregress**,  
[R] **ivtobit**, [R] **reg3**

Engle's LM test, [R] **regress postestimation time series**

eolchar, set subcommand, [R] **set**

Epanechnikov kernel density function, [R] **kdensity**

epidemiological tables, [R] **dstdize**, [R] **symmetry**,  
[R] **tabulate twoway**

equality tests of

binomial proportions, [R] **bitest**

coefficients, [R] **test**, [R] **testnl**

distributions, [R] **ksmirnov**, [R] **kwallis**,  
[R] **ranksum**, [R] **signrank**

means, [R] **ttest**

medians, [R] **ranksum**

proportions, [R] **bitest**, [R] **prtest**

variances, [R] **sdtest**

equivalence tests, [R] **pk**, [R] **pkequiv**

ereturn list command, [R] **saved results**

error-bar charts, [R] **serrbar**

error messages and return codes, [R] **error messages**

errors-in-variables regression, [R] **eivreg**

esample, estimates subcommand, [R] **estimates save estat**

alternatives command, [R] **asclogit**

**postestimation**, [R] **asmprobit postestimation**,  
[R] **asroprobit postestimation**

archlm command, [R] **regress postestimation time series**

bgodfrey command, [R] **regress postestimation time series**

bootstrap command, [R] **bootstrap postestimation**

classification command, [R] **logistic postestimation**

correlation command, [R] **asmprobit postestimation**, [R] **asroprobit postestimation**

covariance command, [R] **asmprobit postestimation**, [R] **asroprobit postestimation**

durbinalt command, [R] **regress postestimation time series**

dwatson command, [R] **regress postestimation time series**

endogenous command, [R] **ivregress postestimation**

facweights command, [R] **asmprobit postestimation**, [R] **asroprobit postestimation**

firststage command, [R] **ivregress postestimation**

gof command, [R] **logistic postestimation**,  
[R] **poisson postestimation**

hettest command, [R] **regress postestimation ic command**, [R] **estat**

imtest command, [R] **regress postestimation**

mfx command, [R] **asclogit postestimation**,  
[R] **asmprobit postestimation**, [R] **asroprobit postestimation**

overid command, [R] **gmm postestimation**,  
[R] **ivregress postestimation**

ovtest command, [R] **regress postestimation**

predict command, [R] **exlogistic postestimation**

se command, [R] **exlogistic postestimation**,  
[R] **expoisson postestimation**

*estat*, *continued*

- summarize command, [R] [estat](#)
- szroeter command, [R] [regress postestimation](#)
- vce command, [R] [estat](#)
- vif command, [R] [regress postestimation](#)

estimable, [R] [margins](#)

estimate linear combinations of coefficients, *see* linear combinations of estimators

estimated marginal means, [R] [margins](#)

estimates

- clear command, [R] [estimates store](#)
- command, [R] [suest](#)
  - introduction, [R] [estimates](#)
- describe command, [R] [estimates describe](#)
- dir command, [R] [estimates store](#)
- drop command, [R] [estimates store](#)
- esample command, [R] [estimates save](#)
- for command, [R] [estimates for](#)
- notes command, [R] [estimates notes](#)
- query command, [R] [estimates store](#)
- replay command, [R] [estimates replay](#)
- restore command, [R] [estimates store](#)
- save command, [R] [estimates save](#)
- stats command, [R] [estimates stats](#)
- store command, [R] [estimates store](#)
- table command, [R] [estimates table](#)
- title command, [R] [estimates title](#)
- use command, [R] [estimates save](#)

estimation

- options, [R] [estimation options](#)
- results,
  - clearing, [R] [estimates store](#)
  - storing and restoring, [R] [estimates store](#)
  - tables of, [R] [estimates table](#)
- sample, summarizing, [R] [estat](#)

estimators,

- covariance matrix of, [R] [correlate](#), [R] [estat](#)
- linear combinations of, [R] [lincom](#)
- nonlinear combinations of, [R] [nlcom](#)

exact

- logistic regression model, [R] [exlogistic](#)
- Poisson regression model, [R] [expoissom](#)
- test, Fisher's, [R] [tabulate twoway](#)

executable, update subcommand, [R] [update](#)

exit command, [R] [exit](#)

exiting Stata, *see* exit command

exlogistic command, [R] [exlogistic](#), [R] [exlogistic postestimation](#), *also see* postestimation command

exploded logit model, [R] [rologit](#)

expoissom command, [R] [expoissom](#), [R] [expoissom postestimation](#), *also see* postestimation command

exponential model, stochastic frontier, [R] [frontier](#)

exponentiated coefficients, [R] [eform\\_option](#)

## F

factor

- analysis, [R] [alpha](#)
- variables, [R] [fvvar](#), [R] [fvset](#)

factor-variable settings, [R] [fvset](#)

factorial design, [R] [anova](#)

faceweights, estat subcommand, [R] [asmprobit postestimation](#), [R] [asroprobit postestimation](#)

failure-time models, *also see* survival analysis

FAQs, search, [R] [search](#)

fastscroll, set subcommand, [R] [set](#)

feasible generalized least squares, *see* FGLS

fences, [R] [lv](#)

FGLS (feasible generalized least squares), [R] [reg3](#)

files, downloading, [R] [adoupdate](#), [R] [net](#), [R] [sj](#), [R] [ssc](#), [R] [update](#)

findit command, [R] [search](#)

firststage, estat subcommand, [R] [ivregress postestimation](#)

Fisher's exact test, [R] [tabulate twoway](#)

fixed-effects models, [R] [anova](#), [R] [areg](#), [R] [asclogit](#), [R] [clogit](#), [R] [xi](#)

flexible functional form, [R] [boxcox](#), [R] [fracpoly](#), [R] [mfp](#)

floatresults, set subcommand, [R] [set](#)

floatwindows, set subcommand, [R] [set](#)

footnote, ml subcommand, [R] [ml](#)

for, estimates subcommand, [R] [estimates for](#)

forecast, standard error of, [R] [predict](#), [R] [regress postestimation](#)

fracgen command, [R] [fracpoly](#)

fracplot command, [R] [fracpoly postestimation](#)

fracpoly prefix command, [R] [fracpoly](#), [R] [fracpoly postestimation](#), *also see* postestimation command

fracpred command, [R] [fracpoly postestimation](#)

fraction defective, [R] [qc](#)

fractional polynomial regression, [R] [fracpoly](#)

multivariable, [R] [mfp](#)

frequencies,

- graphical representation, [R] [histogram](#), [R] [kdensity](#)

table of, [R] [table](#), [R] [tabstat](#), [R] [tabulate](#), [summarize\(\)](#), [R] [tabulate oneway](#), [R] [tabulate twoway](#)

from() option, [R] [maximize](#)

from,

- net subcommand, [R] [net](#)
- update subcommand, [R] [update](#)

frontier command, [R] [frontier](#), [R] [frontier postestimation](#), *also see* postestimation command

frontier models, [R] [frontier](#)

fully factorial, [R] [margins](#)

functions,

- link, [R] [glm](#)
- piecewise cubic, [R] [mkspline](#)
- piecewise linear, [R] [mkspline](#)

**fvrevar** command, [R] **fvrevar**

**fvset** command, [R] **fvset**

## G

generalized

least squares, see **FGLS**

linear latent and mixed models, see **GLLAMM**

linear models, see **GLM**

method of moments, see **gmm** command

negative binomial regression, [R] **nbreg**

**get, net** subcommand, [R] **net**

**gladder** command, [R] **ladder**

**GLLAMM**, [R] **gllamm**

**gllamm** command, [R] **gllamm**

**GLM**, [R] **binreg**, [R] **glm**

**glm** command, [R] **glm**, [R] **glm postestimation**, also see **postestimation** command

**glogit** command, [R] **glogit**, [R] **glogit postestimation**, also see **postestimation** command

**gmm** command, [R] **gmm**, [R] **gmm postestimation**, also see **postestimation** command

**gnbreg** command, [R] **nbreg**, [R] **nbreg postestimation**, also see **postestimation** command

**gof, estat** subcommand, [R] **logistic postestimation**, [R] **poisson postestimation**

Goodman and Kruskal's gamma, [R] **tabulate twoway**

goodness-of-fit tests, [R] **brier**, [R] **diagnostic plots**, [R] **ksmirnov**, [R] **logistic postestimation**, [R] **poisson postestimation**, [R] **regress postestimation**, [R] **swilk**

**gprobit** command, [R] **glogit**, [R] **glogit postestimation**, also see **postestimation** command

gradient option, [R] **maximize**

graph, **ml** subcommand, [R] **ml**

graphics, **set** subcommand, [R] **set**

graphs,

added-variable plot, [R] **regress postestimation**

adjusted partial residual plot, [R] **regress postestimation**

augmented component-plus-residual plot, [R] **regress postestimation**

augmented partial residual plot, [R] **regress postestimation**

binary variable cumulative sum, [R] **cusum**

component-plus-residual, [R] **regress postestimation**

cumulative distribution, [R] **cumul**

density, [R] **kdensity**

density-distribution sunflower, [R] **sunflower**

derivatives, [R] **dydx**, [R] **testnl**

diagnostic, [R] **diagnostic plots**

dotplot, [R] **dotplot**

error-bar charts, [R] **serbrar**

fractional polynomial, [R] **fracpoly**

histograms, [R] **histogram**, [R] **kdensity**

integrals, [R] **dydx**

ladder-of-power histograms, [R] **ladder**

graphs, *continued*

leverage-versus-(squared)-residual, [R] **regress postestimation**

logistic diagnostic, [R] **logistic postestimation**

lowess smoothing, [R] **lowess**

means and medians, [R] **grmeanby**

normal probability, [R] **diagnostic plots**

partial-regression leverage, [R] **regress**

**postestimation**

partial residual, [R] **regress postestimation**

quality control, [R] **qc**

quantile, [R] **diagnostic plots**

quantile-normal, [R] **diagnostic plots**

quantile-quantile, [R] **diagnostic plots**

regression diagnostic, [R] **regress postestimation**

residual versus fitted, [R] **regress postestimation**

residual versus predictor, [R] **regress postestimation**

ROC curve, [R] **logistic postestimation**, [R] **roc**, [R] **rocfit postestimation**

rootograms, [R] **spikeplot**

smoothing, [R] **kdensity**, [R] **lowess**, [R] **lpoly**

spike plot, [R] **spikeplot**

stem-and-leaf, [R] **stem**

sunflower, [R] **sunflower**

symmetry, [R] **diagnostic plots**

time-versus-concentration curve, [R] **pk**, [R] **pkexamine**

Greenhouse-Geisser epsilon, [R] **anova**

**grmeanby** command, [R] **grmeanby**

group-data regression, [R] **glogit**, [R] **intreg**

## H

half-normal model, stochastic frontier, [R] **frontier**

harmonic mean, [R] **ameans**

hat matrix, see projection matrix, diagonal elements of

**hausman** command, [R] **hausman**

Hausman specification test, [R] **hausman**

health ratios, [R] **binreg**

**heckman** command, [R] **heckman**, [R] **heckman**

**postestimation**, also see **postestimation** command

Heckman selection model, [R] **heckman**, [R] **heckprob**

**heckprob** command, [R] **heckprob**, [R] **heckprob**

**postestimation**, also see **postestimation** command

**help, view** subcommand, [R] **view**

**help** command, [R] **help**

**help** file search, [R] **hsearch**

**help** system, searching, [R] **hsearch**

**help\_d, view** subcommand, [R] **view**

**hessian** option, [R] **maximize**

heteroskedastic probit regression, [R] **hetprob**

heteroskedasticity,

conditional, [R] **regress postestimation time series**

robust variances, see robust, Huber/White/sandwich estimator of variance

stochastic frontier, [R] **frontier**

test for, [R] **regress postestimation**, [R] **regress postestimation time series**

hetprob command, [R] **hetprob**, [R] **hetprob postestimation**, *also see* postestimation command

hettest, estat subcommand, [R] **regress postestimation**

hierarchical  
   regression, [R] **areg**, [R] **nestreg**  
   samples, [R] **anova**, [R] **loneway**; [R] **areg**

histogram command, [R] **histogram**

histograms, [R] **histogram**  
   dotplots, [R] **dotplot**  
   kernel density estimator, [R] **kdensity**  
   ladder-of-powers, [R] **ladder**  
   of categorical variables, [R] **histogram**  
   stem-and-leaf, [R] **stem**

Holm adjustment, [R] **test**

homogeneity of variances, [R] **oneway**, [R] **sdtest**

homoskedasticity tests, [R] **regress postestimation**

Hosmer–Lemeshow goodness-of-fit test, [R] **logistic postestimation**

hot, **ssc** subcommand, [R] **ssc**

hsearch command, [R] **hsearch**

httpproxy, set subcommand, [R] **netio**, [R] **set**

httpproxyauth, set subcommand, [R] **netio**, [R] **set**

httpproxyhost, set subcommand, [R] **netio**, [R] **set**

httpproxyport, set subcommand, [R] **netio**, [R] **set**

httpproxypw, set subcommand, [R] **netio**, [R] **set**

httpproxypw, set subcommand, [R] **netio**, [R] **set**

Huber weighting, [R] **rreg**

Huber/White/sandwich estimator of variance, *see* robust, Huber/White/sandwich estimator of variance

Huynh–Feldt epsilon, [R] **anova**

hypertext help, [R] **help**

hypothesis tests, *see* tests

## I

ic, estat subcommand, [R] **estat**

IIA, [R] **asclogit**, [R] **asmprob**, [R] **clogit**, [R] **hausman**, [R] **nlogit**

immediate commands, [R] **bittest**, [R] **ci**, [R] **prtest**, [R] **sampsi**, [R] **sdtest**, [R] **symmetry**, [R] **tabulate twoway**, [R] **ttest**

imtest, estat subcommand, [R] **regress postestimation**

incidence-rate ratio, [R] **poisson**, [R] **zip**  
   differences, [R] **lincom**, [R] **nlcom**

income distributions, [R] **inequality**

independence of irrelevant alternatives, *see* IIA

independence tests, *see* tests

index of probit and logit, [R] **logit postestimation**, [R] **predict**, [R] **probit postestimation**

index search, [R] **search**

indicator variables, [R] **anova**, [R] **areg**, [R] **xi**

indirect standardization, [R] **dstdize**

inequality measures, [R] **inequality**

influence statistics, [R] **logistic postestimation**, [R] **predict**, [R] **regress postestimation**

information  
   criteria, *see* AIC, BIC  
   matrix, [R] **correlate**, [R] **maximize**  
   matrix test, [R] **regress postestimation**

init, ml subcommand, [R] **ml**

inner fence, [R] **lv**

install,  
   **net** subcommand, [R] **net**  
   **ssc** subcommand, [R] **ssc**

installation  
   of official updates, [R] **update**  
   of SJ and STB, [R] **net**, [R] **sj**

instrumental-variables regression, [R] **gmm**, [R] **ivprobit**, [R] **ivregress**, [R] **ivtobit**, [R] **nlstur**

integ command, [R] **dydx**

integrals, numeric, [R] **dydx**

interaction, [R] **margins**

interaction expansion, [R] **xi**

internal consistency, test for, [R] **alpha**

Internet,  
   commands to control connections to, [R] **netio**  
   installation of updates from, [R] **adoupdate**, [R] **net**, [R] **sj**, [R] **update**  
   search, [R] **net search**

interquantile range, [R] **qreg**

interquartile range, [R] **lv**  
   reporting, [R] **table**, [R] **tabstat**

intrater agreement, [R] **kappa**

interval regression, [R] **intreg**

intracluster correlation, [R] **loneway**

intreg command, [R] **intreg**, [R] **intreg postestimation**, *also see* postestimation command

IQR, *see* interquartile range

iqreg command, [R] **qreg**, [R] **qreg postestimation**, *also see* postestimation command

IRLS, [R] **glm**, [R] **reg3**

istdize command, [R] **dstdize**

iterate() option, [R] **maximize**

iterated least squares, [R] **reg3**, [R] **sureg**

iterations, controlling the maximum number, [R] **maximize**

ivprobit command, [R] **ivprobit**, [R] **ivprobit postestimation**, *also see* postestimation command

ivregress command, [R] **ivregress**, [R] **ivregress postestimation**, *also see* postestimation command

ivtobit command, [R] **ivtobit**, [R] **ivtobit postestimation**, *also see* postestimation command

## J

jackknife  
   estimation, [R] **jackknife**  
   standard errors, [R] **vce\_option**

jackknife prefix command, [R] **jackknife**, [R] **jackknife postestimation**, *also see* postestimation command

jackknifed residuals, [R] **predict**, [R] **regress postestimation**

## K

kap command, [R] **kappa**  
 kappa command, [R] **kappa**  
 kapwgt command, [R] **kappa**  
 kdensity command, [R] **kdensity**  
 Kendall's tau, [R] **spearman**, [R] **tabulate twoway**  
 kernel density estimator, [R] **kdensity**  
 kernel-weighted local polynomial estimator, [R] **lpoly**  
 Kish design effects, [R] **loneway**  
 Kolmogorov–Smirnov test, [R] **ksmirnov**  
 KR-20, [R] **alpha**  
 Kruskal–Wallis test, [R] **kwallis**  
 ksmirnov command, [R] **ksmirnov**  
 ktau command, [R] **spearman**  
 Kuder–Richardson Formula 20, [R] **alpha**  
 kurtosis, [R] **lv**, [R] **regress postestimation**, [R] **sktest**,  
     [R] **summarize**, [R] **tabstat**  
 kwallis command, [R] **kwallis**

## L

L-R plots, [R] **regress postestimation**  
 L1-norm models, [R] **qreg**  
 LAD regression, [R] **qreg**  
 ladder command, [R] **ladder**  
 ladder of powers, [R] **ladder**  
 Lagrange-multiplier test, [R] **regress postestimation time series**  
 Latin-square designs, [R] **anova**, [R] **pkshape**  
 launch dialog box, [R] **db**  
 LAV regression, [R] **qreg**  
 least absolute  
     deviations, [R] **qreg**  
     residuals, [R] **qreg**  
     value regression, [R] **qreg**  
 least squared deviations, [R] **regress** [R] **regress postestimation**; [R] **areg**, [R] **cnsreg**, [R] **nl**  
 least squares, see linear regression  
     generalized, see FGLS  
 least-squares means, [R] **margins**  
 letter values, [R] **lv**  
 level, set subcommand, [R] **level**, [R] **set**  
 Levene's robust test statistic, [R] **sdtest**  
 leverage, [R] **logistic postestimation**, [R] **predict**,  
     [R] **regress postestimation**  
     obtaining with weighted data, [R] **predict**  
 leverage-versus-(squared)-residual plot, [R] **regress postestimation**  
 license, [R] **about**  
 likelihood, see maximum likelihood estimation  
 likelihood-ratio  
     chi-squared of association, [R] **tabulate twoway**  
     test, [R] **lrtest**  
 Likert summative scales, [R] **alpha**  
 limits, [R] **matsize**  
 lincom command, [R] **lincom**, [R] **test**  
 linear  
     combinations of estimators, [R] **lincom**  
     regression, [R] **anova**, [R] **areg**, [R] **cnsreg**,  
         [R] **eivreg**, [R] **frontier**, [R] **glm**, [R] **gmm**,  
         [R] **heckman**, [R] **intreg**, [R] **ivregress**,  
         [R] **ivtobit**, [R] **mvreg**, [R] **qreg**, [R] **reg3**,  
         [R] **regress**, [R] **rreg**, [R] **sureg**, [R] **tobit**,  
         [R] **vwls**  
     splines, [R] **mkspline**  
 linegap, set subcommand, [R] **set**  
 linesize, set subcommand, [R] **log**, [R] **set**  
 link, net subcommand, [R] **net**  
 link function, [R] **glm**  
 linktest command, [R] **linktest**  
 list,  
     ereturn subcommand, [R] **saved results**  
     return subcommand, [R] **saved results**  
     sreturn subcommand, [R] **saved results**  
 lnskev0 command, [R] **lnskew0**  
 local linear, [R] **lpoly**  
 local polynomial, [R] **lpoly**  
 locally weighted smoothing, [R] **lowess**  
 location, measures of, [R] **lv**, [R] **summarize**, [R] **table**  
 locksplitters, set subcommand, [R] **set**  
 log command, [R] **log**, [R] **view**  
 log files, printing, [R] **translate**, also see log command  
 log-linear model, [R] **glm**, [R] **poisson**, [R] **zip**  
 log or no log option, [R] **maximize**  
 log transformations, [R] **boxcox**, [R] **lnskew0**  
 logistic and logit regression, [R] **logistic**, [R] **logit**  
     complementary log-log, [R] **cloglog**  
     conditional, [R] **asclogit**, [R] **clogit**, [R] **rologit**  
     exact, [R] **exlogistic**  
     fixed-effects, [R] **asclogit**, [R] **clogit**  
     generalized linear model, [R] **glm**  
     multinomial, [R] **asclogit**, [R] **clogit**, [R] **mlogit**  
     nested, [R] **nlogit**  
     ordered, [R] **ologit**  
     polytomous, [R] **mlogit**  
     rank-ordered, [R] **rologit**  
     skewed, [R] **scobit**  
     stereotype, [R] **slogit**  
     with grouped data, [R] **glogit**  
 logistic command, [R] **logistic**, [R] **logistic postestimation**, also see postestimation command  
 logit command, [R] **logit**, [R] **logit postestimation**,  
     also see postestimation command  
 logit regression, see logistic and logit regression  
 lognormal distribution, [R] **ameans**  
 logtype, set subcommand, [R] **log**, [R] **set**  
 loneway command, [R] **loneway**  
 loop, endless, see endless loop  
 Lorenz curve, [R] **inequality**  
 lowess, see locally weighted smoothing  
 lowess command, [R] **lowess**  
 lpoly command, [R] **lpoly**, also see smooth command  
 lroc command, [R] **logistic postestimation**

lrtest command, [R] [lrtest](#)  
lsens command, [R] [logistic postestimation](#)  
lstat command, see [estat classification](#) command  
ltolerance() option, [R] [maximize](#)  
lv command, [R] [lv](#)  
lvr2plot command, [R] [regress postestimation](#)

## M

MAD regression, [R] [qreg](#)  
main effects, [R] [anova](#)  
man command, [R] [help](#)  
Mann–Whitney two-sample statistics, [R] [ranksum](#)  
marginal  
    effects, [R] [margins](#)  
    homogeneity, test of, [R] [symmetry](#)  
    means, [R] [margins](#)  
margins command, [R] [margins](#), [R] [margins postestimation](#)  
matacache, set subcommand, [R] [set](#)  
matafavor, set subcommand, [R] [set](#)  
matalibs, set subcommand, [R] [set](#)  
matalnum, set subcommand, [R] [set](#)  
matamofirst, set subcommand, [R] [set](#)  
mataoptimize, set subcommand, [R] [set](#)  
matastrict, set subcommand, [R] [set](#)  
matched case–control data, [R] [asclogit](#), [R] [clogit](#), [R] [symmetry](#)  
matched-pairs tests, [R] [signrank](#), [R] [ttest](#)  
matsize, set subcommand, [R] [matsize](#), [R] [set](#)  
maxdb, set subcommand, [R] [db](#), [R] [set](#)  
maximization technique explained, [R] [maximize](#), [R] [ml](#)  
maximize, ml subcommand, [R] [ml](#)  
maximum  
    likelihood estimation, [R] [maximize](#), [R] [ml](#)  
    number of variables in a model, [R] [matsize](#)  
maximums and minimums, reporting, [R] [lv](#), [R] [summarize](#), [R] [table](#)  
maxiter, set subcommand, [R] [maximize](#), [R] [set](#)  
maxvar, set subcommand, [R] [set](#)  
McFadden’s choice model, [R] [asclogit](#)  
McNemar’s chi-squared test, [R] [clogit](#)  
mean command, [R] [mean](#), [R] [mean postestimation](#), *also see* [postestimation](#) command  
means,  
    arithmetic, geometric, and harmonic, [R] [ameans](#)  
    confidence interval and standard error, [R] [ci](#)  
    displaying, [R] [ameans](#), [R] [summarize](#), [R] [table](#), [R] [tabstat](#), [R] [tabulate](#), [summarize\(\)](#)  
    estimates of, [R] [mean](#)  
    graphing, [R] [grmeanby](#)  
    robust, [R] [rreg](#)  
    testing equality of, [R] [ttest](#)  
    testing equality of, sample size or power, [R] [sampsi](#)  
measurement error, [R] [alpha](#), [R] [vwls](#)

measures of  
    association, [R] [tabulate twoway](#)  
    inequality, [R] [inequality](#)  
    location, [R] [lv](#), [R] [summarize](#)  
median command, [R] [ranksum](#)  
median regression, [R] [qreg](#)  
median test, [R] [ranksum](#)  
medians,  
    displaying, [R] [centile](#), [R] [lv](#), [R] [summarize](#), [R] [table](#), [R] [tabstat](#)  
    graphing, [R] [grmeanby](#)  
    testing equality of, [R] [ranksum](#)  
memory, matsize, see [matsize](#), set subcommand  
memory, set subcommand, [R] [set](#)  
messages and return codes, see [error messages and return codes](#)  
meta-analysis, [R] [meta](#)  
mfp prefix command, [R] [mfp](#), [R] [mfp postestimation](#), *also see* [postestimation](#) command  
mfx, estat subcommand, [R] [asclogit postestimation](#), [R] [asmprobit postestimation](#), [R] [asroprobit postestimation](#)  
midsummaries, [R] [lv](#)  
mild outliers, [R] [lv](#)  
Mills’ ratio, [R] [heckman](#), [R] [heckman postestimation](#)  
minimum  
    absolute deviations, [R] [qreg](#)  
    squared deviations, [R] [areg](#), [R] [cnsreg](#), [R] [nl](#), [R] [regress](#), [R] [regress postestimation](#)  
minimums and maximums, see [maximums and minimums](#)  
missing values, [R] [misstable](#)  
misstable  
    nested command, [R] [misstable](#)  
    patterns command, [R] [misstable](#)  
    summarize command, [R] [misstable](#)  
    tree command, [R] [misstable](#)  
mixed designs, [R] [anova](#)  
mkspline command, [R] [mkspline](#)  
ml  
    check command, [R] [ml](#)  
    clear command, [R] [ml](#)  
    count command, [R] [ml](#)  
    display command, [R] [ml](#)  
    footnote command, [R] [ml](#)  
    graph command, [R] [ml](#)  
    init command, [R] [ml](#)  
    maximize command, [R] [ml](#)  
    model command, [R] [ml](#)  
    plot command, [R] [ml](#)  
    query command, [R] [ml](#)  
    report command, [R] [ml](#)  
    score command, [R] [ml](#)  
    search command, [R] [ml](#)  
    trace command, [R] [ml](#)  
mlevel command, [R] [ml](#)



mlmatbysum command, [R] **ml**  
 mlmatsum command, [R] **ml**  
 mlogit command, [R] **mlogit**, [R] **mlogit**  
     **postestimation**, *also see* postestimation command  
 mlsum command, [R] **ml**  
 mlvecsum command, [R] **ml**  
 MNP, *see* multinomial outcome model  
 model,  
     maximum number of variables in, [R] **matsize**  
     sensitivity, [R] **regress postestimation**, [R] **rreg**  
     specification test, [R] **linktest**, [R] **regress**  
         **postestimation**  
 model, **ml** subcommand, [R] **ml**  
 modulus transformations, [R] **boxcox**  
 monotone missing values, [R] **misstable**  
 Monte Carlo simulations, [R] **permute**, [R] **simulate**  
 more, **set** subcommand, [R] **more**, [R] **set**  
 more command and parameter, [R] **more**  
 more condition, [R] **query**  
 mprobit command, [R] **mprobit**  
 multilevel models, [R] **gllamm**  
 multinomial outcome model, [R] **asclogit**,  
     [R] **asmprobit**, [R] **asroprobit**, [R] **clogit**,  
     [R] **mlogit**, [R] **mprobit**, [R] **nlogit**, [R] **slogit**,  
     *also see* polytomous outcome model  
 multiple  
     comparison tests, [R] **oneway**  
     regression, *see* linear regression  
     testing, [R] **regress postestimation**, [R] **test**,  
         [R] **testnl**  
 multivariable fractional polynomial regression, [R] **mfp**  
 multivariate analysis,  
     bivariate probit, [R] **biprobit**  
     regression, [R] **mvreg**  
     three-stage least squares, [R] **reg3**  
     Zellner's seemingly unrelated, [R] **nlstur**, [R] **sureg**  
 mvreg command, [R] **mvreg**, [R] **mvreg**  
     **postestimation**, *also see* postestimation command

## N

natural splines, [R] **mkspline**  
 nbreg command, [R] **nbreg**, [R] **nbreg postestimation**,  
     *also see* postestimation command  
 needle plot, [R] **spikeplot**  
 negative binomial regression, [R] **nbreg**; [R] **glm**  
     zero-inflated, [R] **zinb**  
     zero-truncated, [R] **ztnb**  
 nested, **misstable** subcommand, [R] **misstable**  
 nested  
     designs, [R] **anova**  
     effects, [R] **anova**  
     logit, [R] **nlogit**  
     model statistics, [R] **nestreg**  
     regression, [R] **nestreg**  
 nestreg prefix command, [R] **nestreg**  
 net, **view** subcommand, [R] **view**

net  
     **cd** command, [R] **net**  
     describe command, [R] **net**  
     from command, [R] **net**  
     get command, [R] **net**  
     install command, [R] **net**  
     link command, [R] **net**  
     query command, [R] **net**  
     search command, [R] **net search**  
     set ado command, [R] **net**  
     set other command, [R] **net**  
     sj command, [R] **net**  
     stb command, [R] **net**  
 net\_d, **view** subcommand, [R] **view**  
 new, **ssc** subcommand, [R] **ssc**  
 Newey–West standard errors, [R] **glm**  
 news, **view** subcommand, [R] **view**  
 news command, [R] **news**  
 Newton–Raphson algorithm, [R] **ml**  
 nl command, [R] **nl**, [R] **nl postestimation**, *also see*  
     postestimation command  
 nlcom command, [R] **nlcom**  
 nlogit command, [R] **nlogit**, [R] **nlogit**  
     **postestimation**, *also see* postestimation command  
 nlogitgen command, [R] **nlogit**  
 nlogittree command, [R] **nlogit**  
 nlsur command, [R] **nlsur**, [R] **nlsur postestimation**,  
     *also see* postestimation command  
 nolog or log option, [R] **maximize**  
 nonconformities, quality control, [R] **qc**  
 nonconstant variance, *see* robust, Huber/White/sandwich  
     estimator of variance  
 nonlinear  
     combinations of estimators, [R] **nlcom**  
     least squares, [R] **nl**  
     predictions, [R] **predictnl**  
     regression, [R] **boxcox**, [R] **nl**, [R] **nlsur**  
 nonparametric estimation, [R] **lpoly**  
 nonparametric tests,  
     association, [R] **spearman**  
     equality of  
         distributions, [R] **ksmirnov**, [R] **kwallis**,  
             [R] **ranksum**, [R] **signrank**  
         medians, [R] **ranksum**  
         proportions, [R] **bitest**, [R] **prtest**  
     percentiles, [R] **centile**  
     ROC analysis, [R] **roc**  
     serial independence, [R] **runtest**  
     tables, [R] **tabulate twoway**  
     trend, [R] **cusum**, [R] **nptrend**  
 nonrtolerance option, [R] **maximize**  
 normal distribution and normality,  
     examining distributions for, [R] **diagnostic plots**,  
         [R] **lv**  
     probability and quantile plots, [R] **diagnostic plots**  
     test for, [R] **sktest**, [R] **swilk**

normal distribution and normality, *continued*  
transformations to achieve, [R] **boxcox**, [R] **ladder**,  
[R] **lnskew0**  
**notes**, **estimates** subcommand, [R] **estimates notes**  
notes on estimation results, [R] **estimates notes**  
**notifyuser**, **set** subcommand, [R] **set**  
**npntrend** command, [R] **npntrend**  
NR algorithm, [R] **ml**  
**nrtolerance()** option, [R] **maximize**  
N-way analysis of variance, [R] **anova**

## O

**obs**, **set** subcommand, [R] **set**  
observed information matrix, see OIM  
**odbcngr**, **set** subcommand, [R] **set**  
odds ratio, [R] **asclogit**, [R] **asroprobit**, [R] **binreg**,  
[R] **clogit**, [R] **cloglog**, [R] **glm**, [R] **glogit**,  
[R] **logistic**, [R] **logit**, [R] **mlogit**, [R] **rologit**,  
[R] **scobit**  
differences, [R] **lincom**, [R] **nlcom**  
OIM, [R] **ml**, [R] **vce\_option**  
**ologit** command, [R] **ologit**, [R] **ologit**  
**postestimation**, *also see* **postestimation** command  
OLS regression, see linear regression  
omitted variables test, [R] **regress postestimation**  
one-way analysis of variance, [R] **kwallis**, [R] **loneway**,  
[R] **oneway**  
**oneway** command, [R] **oneway**  
online help, [R] **help**, [R] **hsearch**, [R] **search**  
OPG, [R] **ml**, [R] **vce\_option**  
**oprobit** command, [R] **oprobit**, [R] **oprobit**  
**postestimation**, *also see* **postestimation** command  
order statistics, [R] **lv**  
**ordered**  
logit, [R] **ologit**  
probit, [R] **oprobit**  
ordinal outcome model, [R] **ologit**, [R] **oprobit**,  
[R] **rologit**, [R] **slogit**, *also see* polytomous  
outcome model  
ordinary least squares, see linear regression  
**orthog** command, [R] **orthog**  
orthogonal polynomials, [R] **orthog**  
**orthpoly** command, [R] **orthog**  
outer fence, [R] **lv**  
outer product of the gradient, see OPG  
outliers, [R] **lv**, [R] **qreg**, [R] **regress postestimation**,  
[R] **rreg**  
out-of-sample predictions, [R] **predict**, [R] **predictnl**  
**output**,  
controlling the scrolling of, [R] **more**  
printing, [R] **translate**  
recording, [R] **log**  
**output**, **set** subcommand, [R] **set**  
outside values, [R] **lv**  
**overid**, **estat** subcommand, [R] **gmm postestimation**,  
[R] **ivregress postestimation**

**ovtest**, **estat** subcommand, [R] **regress**  
**postestimation**

## P

**pagesize**, **set** subcommand, [R] **more**, [R] **set**  
paging of screen output, controlling, [R] **more**  
pairwise correlation, [R] **correlate**  
parameters, system, see system parameters  
**partial**  
correlation, [R] **pcorr**  
effects, [R] **margins**  
regression leverage plot, [R] **regress postestimation**  
regression plot, [R] **regress postestimation**  
residual plot, [R] **regress postestimation**  
Parzen kernel density function, [R] **kdensity**  
pattern of missing values, [R] **misstable**  
**patterns**, **misstable** subcommand, [R] **misstable**  
pausing until key is pressed, [R] **more**  
**pchart** command, [R] **qc**  
**pchi** command, [R] **diagnostic plots**  
**pcorr** command, [R] **pcorr**  
PDF files (Mac only), [R] **translate**  
Pearson goodness-of-fit test, [R] **logistic postestimation**,  
[R] **poisson postestimation**  
Pearson product-moment correlation coefficient,  
[R] **correlate**  
Pearson residual, [R] **binreg postestimation**, [R] **glm**  
**postestimation**, [R] **logistic postestimation**,  
[R] **logit postestimation**  
percentiles, displaying, [R] **centile**, [R] **lv**,  
[R] **summarize**, [R] **table**, [R] **tabstat**  
permutation tests, [R] **permute**  
**permute** prefix command, [R] **permute**  
**persistfv**, **set** subcommand, [R] **set**  
**persistvtopic**, **set** subcommand, [R] **set**  
pharmaceutical statistics, [R] **pk**, [R] **pksumm**  
pharmacokinetic data, see **pk** (pharmacokinetic data)  
**piecewise**  
cubic functions, [R] **mkspline**  
linear functions, [R] **mkspline**  
**pinnable**, **set** subcommand, [R] **set**  
**pk** (pharmacokinetic data), [R] **pk**, [R] **pkcollapse**,  
[R] **pkcross**, [R] **pkequiv**, [R] **pkexamine**,  
[R] **pkshape**, [R] **pksumm**  
**pkcollapse** command, [R] **pkcollapse**  
**pkcross** command, [R] **pkcross**  
**pkequiv** command, [R] **pkequiv**  
**pkexamine** command, [R] **pkexamine**  
.pkg filename suffix, [R] **net**  
**pkshape** command, [R] **pkshape**  
**pksumm** command, [R] **pksumm**  
Plackett–Luce model, [R] **rologit**  
**playsnd**, **set** subcommand, [R] **set**  
**plot**, **ml** subcommand, [R] **ml**  
**pnorm** command, [R] **diagnostic plots**



poisson command, [R] **poisson**, [R] **poisson postestimation**; [R] **nbreg**, *also see* postestimation command

Poisson distribution,  
   confidence intervals, [R] **ci**  
   regression, *see* Poisson regression

Poisson regression, [R] **glm**, [R] **nbreg**, [R] **poisson**  
   zero-inflated, [R] **zip**  
   zero-truncated, [R] **ztp**

polynomial, smoothing, *see* local polynomial

polynomials, orthogonal, [R] **orthog**

polytomous outcome model, [R] **asclogit**,  
   [R] **asmprobit**, [R] **asroprobit**, [R] **clogit**,  
   [R] **mlogit**, [R] **mprobit**, [R] **nlogit**, [R] **ologit**,  
   [R] **oprobit**, [R] **rologit**, [R] **sllogit**

population marginal means, [R] **margins**

populations,  
   diagnostic plots, [R] **diagnostic plots**  
   examining, [R] **histogram**, [R] **lv**, [R] **stem**,  
     [R] **summarize**, [R] **table**  
   standard, [R] **dstdize**  
   testing equality of, [R] **ksmirnov**, [R] **kwallis**,  
     [R] **signrank**  
   testing for normality, [R] **sktest**, [R] **swilk**

post hoc tests, [R] **oneway**

postestimation command, [R] **estat**, [R] **estimates**,  
   [R] **hausman**, [R] **lincom**, [R] **linktest**,  
   [R] **lrtest**, [R] **margins**, [R] **nlcom**, [R] **predict**,  
   [R] **predictnl**, [R] **suest**, [R] **test**, [R] **testnl**

poverty indices, [R] **inequality**

power of a test, [R] **sampsi**

power transformations, [R] **boxcox**, [R] **lnskew0**

P–P plot, [R] **diagnostic plots**

predict, **estat** subcommand, [R] **exlogistic postestimation**

predict command, [R] **predict**, [R] **regress postestimation**

predicted marginals, [R] **margins**

predicted population margins, [R] **margins**

prediction, standard error of, [R] **glm**, [R] **predict**,  
   [R] **regress postestimation**

predictions, [R] **margins**, [R] **predict**  
   nonlinear, [R] **predictnl**

predictive marginal means, [R] **margins**

predictive margins, [R] **margins**

predictnl command, [R] **predictnl**

prefix command, [R] **bootstrap**, [R] **fracpoly**,  
   [R] **jackknife**, [R] **mfp**, [R] **nestreg**,  
   [R] **permute**, [R] **simulate**, [R] **stepwise**, [R] **xi**

Pregibon delta beta influence statistic, [R] **logistic**

preprocessor commands, [R] **#review**

prevalence studies, *see* case–control data

print command, [R] **translate**

printcolor, **set** subcommand, [R] **set**

printing, logs (output), [R] **translate**

probit command, [R] **probit**, [R] **probit postestimation**, *also see* postestimation command

probit regression, [R] **probit**  
   alternative-specific multinomial probit,  
     [R] **asmprobit**  
   alternative-specific rank-ordered, [R] **asroprobit**  
   bivariate, [R] **biprobit**  
   generalized linear model, [R] **glm**  
   heteroskedastic, [R] **hetprob**  
   multinomial, [R] **mprobit**  
   ordered, [R] **oprobit**  
   two-equation, [R] **biprobit**  
   with endogenous regressors, [R] **ivprobit**  
   with grouped data, [R] **glogit**  
   with sample selection, [R] **heckprob**

processors, **set** subcommand, [R] **set**

product-moment correlation, [R] **correlate**  
   between ranks, [R] **spearman**

production frontier models, [R] **frontier**

programs, user-written, [R] **sj**, [R] **ssc**

projection matrix, diagonal elements of, [R] **logistic postestimation**, [R] **logit postestimation**, [R] **probit postestimation**, [R] **regress postestimation**, [R] **rreg**

proportion command, [R] **proportion**,  
   [R] **proportion postestimation**, *also see* postestimation command

proportional  
   hazards models, *see* survival analysis  
   odds model, [R] **ologit**, [R] **slogit**  
   sampling, [R] **bootstrap**

proportions, [R] **ci**, [R] **sampsi**  
   estimating, [R] **proportion**  
   testing equality of, [R] **bitest**, [R] **prtest**

prtest command, [R] **prtest**

prtesti command, [R] **prtest**

pseudo *R*-squared, [R] **maximize**

pseudosigmas, [R] **lv**

pwcorr command, [R] **correlate**

## Q

qc charts, *see* quality control charts

qchi command, [R] **diagnostic plots**

qladder command, [R] **ladder**

qnorm command, [R] **diagnostic plots**

Q–Q plot, [R] **diagnostic plots**

qqplot command, [R] **diagnostic plots**

\_qreg command, [R] **qreg**

qreg command, [R] **qreg**, [R] **qreg postestimation**,  
   *also see* postestimation command

qualitative dependent variables, [R] **asclogit**,  
   [R] **asmprobit**, [R] **asroprobit**, [R] **binreg**,  
   [R] **biprobit**, [R] **brier**, [R] **clogit**, [R] **cloglog**,  
   [R] **cusum**, [R] **exlogistic**, [R] **glm**, [R] **glogit**,  
   [R] **heckprob**, [R] **hetprob**, [R] **ivprobit**,  
   [R] **logistic**, [R] **logit**, [R] **mlogit**, [R] **mprobit**,  
   [R] **nlogit**, [R] **ologit**, [R] **oprobit**, [R] **probit**,  
   [R] **rocfit**, [R] **rologit**, [R] **scobit**, [R] **slogit**

quality control charts, [R] **qc**, [R] **serrbar**

quantile command, [R] **diagnostic plots**  
 quantile–normal plots, [R] **diagnostic plots**  
 quantile plots, [R] **diagnostic plots**  
 quantile–quantile plots, [R] **diagnostic plots**  
 quantile regression, [R] **qreg**  
 quantiles, see percentiles, displaying  
 query,  
   estimates subcommand, [R] **estimates store**  
   ml subcommand, [R] **ml**  
   net subcommand, [R] **net**  
   translator subcommand, [R] **translate**  
   transmap subcommand, [R] **translate**  
   update subcommand, [R] **update**  
 query command, [R] **query**  
 quitting Stata, see exit command

## R

r() saved results, [R] **saved results**  
 Ramsey test, [R] **regress postestimation**  
 random  
   order, test for, [R] **runtest**  
   sample, [R] **bootstrap**  
 random-effects models, [R] **anova**, [R] **loneaway**  
 range chart, [R] **qc**  
 range of data, [R] **lv**, [R] **summarize**, [R] **table**  
 rank correlation, [R] **spearman**  
 rank-order statistics, [R] **signrank**, [R] **spearman**  
 rank-ordered logistic regression, [R] **rologit**  
 ranking data, [R] **rologit**  
 ranksum command, [R] **ranksum**  
 rate ratio, see incidence-rate ratio  
 ratio command, [R] **ratio**, [R] **ratio postestimation**,  
   also see postestimation command  
 ratios, estimating, [R] **ratio**  
 rc (return codes), see error messages and return codes  
 rchart command, [R] **qc**  
 receiver operating characteristic (ROC) analysis,  
   [R] **logistic postestimation**, [R] **roc**, [R] **rocfit**,  
   [R] **rocfit postestimation**  
 recycled predictions, [R] **margins**  
 reexpression, [R] **boxcox**, [R] **ladder**, [R] **lnskew0**  
 reg3 command, [R] **reg3**, [R] **reg3 postestimation**,  
   also see postestimation command  
 regress command, [R] **regress**, [R] **regress**  
   **postestimation**, [R] **regress postestimation time**  
   **series**, also see postestimation command  
 regression,  
   constrained, [R] **cnsreg**  
   creating orthogonal polynomials for, [R] **orthog**  
   dummy variables, with, [R] **anova**, [R] **areg**, [R] **xi**  
   fixed-effects, [R] **areg**  
   fractional polynomial, [R] **fracpoly**, [R] **mfp**  
   graphing, [R] **logistic**, [R] **regress postestimation**  
   grouped data, [R] **intreg**  
   increasing number of variables allowed, [R] **matsize**  
   instrumental variables, [R] **ivprobit**, [R] **ivregress**,  
   [R] **ivtobit**, [R] **nlshr**

regression, *continued*  
   linear, see linear regression  
   system, [R] **mvreg**, [R] **reg3**, [R] **sureg**  
   truncated, [R] **truncreg**  
 regression  
   diagnostics, [R] **predict**; [R] **ladder**, [R] **logistic**,  
     [R] **regress postestimation**, [R] **regress**  
     **postestimation time series**  
   function, estimating, [R] **lpolys**  
   instrumental variables, [R] **gmm**  
 reliability, [R] **alpha**, [R] **eivreg**, [R] **loneaway**  
 reliability theory, see survival analysis  
 repeated measures ANOVA, [R] **anova**  
 repeating and editing commands, [R] **#review**  
 replay, estimates subcommand, [R] **estimates**  
   **replay**  
 report, ml subcommand, [R] **ml**  
 reset, translator subcommand, [R] **translate**  
 RESET test, [R] **regress postestimation**  
 residual-versus-fitted plot, [R] **regress postestimation**  
 residual-versus-predictor plot, [R] **regress**  
   **postestimation**  
 residuals, [R] **logistic**, [R] **predict**, [R] **regress**  
   **postestimation**, [R] **rreg**  
 resistant smoothers, [R] **smooth**  
 restore, estimates subcommand, [R] **estimates**  
   **store**  
 restricted cubic splines, [R] **mkspline**  
 results,  
   saved, [R] **saved results**  
   saving, [R] **estimates save**  
 return codes, see error messages and return codes  
 return list command, [R] **saved results**  
 reventries, set subcommand, [R] **set**  
 #review command, [R] **#review**  
 revkeyboard, set subcommand, [R] **set**  
 risk ratio, [R] **binreg**  
 rmsg, set subcommand, [R] **set**  
 robust, Huber/White/sandwich estimator of variance,  
   [R] **vce\_option**  
   alternative-specific conditional logit model,  
     [R] **asclogit**  
   alternative-specific multinomial probit regression,  
     [R] **asmprobit**  
   alternative-specific rank-ordered probit regression,  
     [R] **asroprobit**  
   bivariate probit regression, [R] **biprobit**  
   complementary log-log regression, [R] **cloglog**  
   conditional logistic regression, [R] **clogit**  
   constrained linear regression, [R] **cnsreg**  
   generalized linear models, [R] **glm**  
   generalized linear models for binomial family,  
     [R] **binreg**  
   generalized method of moments, [R] **gmm**  
   heckman selection model, [R] **heckman**  
   heteroskedastic probit model, [R] **hetprobit**  
   instrumental-variables regression, [R] **ivregress**  
   interval regression, [R] **intreg**

robust, Huber/White/sandwich estimator of variance,  
*continued*

linear regression, [R] [regress](#)

linear regression with dummy-variable set, [R] [areg](#)

logistic regression, [R] [logistic](#), [R] [logit](#)

logit and probit estimation for grouped data,  
[R] [glogit](#)

multinomial logistic regression, [R] [mlogit](#)

multinomial probit regression, [R] [mprobit](#)

negative binomial regression, [R] [nbreg](#)

nested logit regression, [R] [nlogit](#)

nonlinear least-squares estimation, [R] [nl](#)

nonlinear systems of equations, [R] [nlsur](#)

ordered logistic regression, [R] [ologit](#)

ordered probit regression, [R] [oprobit](#)

Poisson regression, [R] [poisson](#)

probit model with endogenous regressors,  
[R] [ivprobit](#)

probit model with sample selection, [R] [heckprob](#)

probit regression, [R] [probit](#)

rank-ordered logistic regression, [R] [rologit](#)

skewed logistic regression, [R] [scobit](#)

stereotype logistic regression, [R] [slogit](#)

tobit model, [R] [tobit](#)

tobit model with endogenous regressors, [R] [ivtobit](#)

treatment-effects model, [R] [treatreg](#)

truncated regression, [R] [truncreg](#)

zero-inflated negative binomial regression, [R] [zinb](#)

zero-inflated Poisson regression, [R] [zip](#)

zero-truncated negative binomial regression, [R] [ztnb](#)

zero-truncated Poisson regression, [R] [ztp](#)

robust, other methods of, [R] [qreg](#), [R] [rreg](#),  
[R] [smooth](#)

robust regression, [R] [regress](#), [R] [rreg](#), *also see* robust,  
Huber/White/sandwich estimator of variance

robust test for equality of variance, [R] [sdtest](#)

robvar command, [R] [sdtest](#)

ROC analysis, [R] [logistic postestimation](#), [R] [roc](#),  
[R] [rocfit](#), [R] [rocfit postestimation](#)

roccomp command, [R] [roc](#)

rocfit command, [R] [rocfit](#), [R] [rocfit postestimation](#),  
*also see* postestimation command

rocgold command, [R] [roc](#)

rocplot command, [R] [rocfit postestimation](#)

roctab command, [R] [roc](#)

roh, [R] [loneway](#)

rologit command, [R] [rologit](#), [R] [rologit](#)  
[postestimation](#), *also see* postestimation command

rootograms, [R] [spikeplot](#)

rreg command, [R] [rreg](#), [R] [rreg postestimation](#), *also*  
*see* postestimation command

run command, [R] [do](#)

runiform() function, [R] [set seed](#)

runtest command, [R] [runtest](#)

rvfplot command, [R] [regress postestimation](#)

rvpplot command, [R] [regress postestimation](#)

## S

s() saved results, [R] [saved results](#)

S\_ macros, [R] [saved results](#)

sample, random, *see* random sample

sample size, [R] [sampsi](#)

sampling, [R] [bootstrap](#), [R] [bsample](#), *also see* cluster  
sampling

sampsi command, [R] [sampsi](#)

sandwich/Huber/White estimator of variance, *see* robust,  
Huber/White/sandwich

save, estimates subcommand, [R] [estimates save](#)

saved results, [R] [saved results](#)

saving results, [R] [estimates save](#)

Scheffé multiple comparison test, [R] [oneway](#)

scheme, set subcommand, [R] [set](#)

Schwarz information criterion, *see* BIC

s-class command, [R] [saved results](#)

scobit command, [R] [scobit](#), [R] [scobit](#)  
[postestimation](#), *also see* postestimation command

score, ml subcommand, [R] [ml](#)

scores, [R] [predict](#)

scrollbufsize, set subcommand, [R] [set](#)

scrolling of output, controlling, [R] [more](#)

sdtest command, [R] [sdtest](#)

sdtesti command, [R] [sdtest](#)

se, estat subcommand, [R] [exlogistic postestimation](#),  
[R] [expoissn postestimation](#)

search,

ml subcommand, [R] [ml](#)

net subcommand, [R] [net](#)

view subcommand, [R] [view](#)

search

help, [R] [hsearch](#)

Internet, [R] [net search](#)

search command, [R] [search](#)

search\_d, view subcommand, [R] [view](#)

searchdefault, set subcommand, [R] [search](#), [R] [set](#)

seed, set subcommand, [R] [set seed](#); [R] [set](#)

seemingly unrelated

estimation, [R] [suest](#)

regression, [R] [nlsur](#), [R] [reg3](#), [R] [sureg](#)

selection models, [R] [heckman](#), [R] [heckprob](#)

sensitivity, [R] [logistic](#)

model, [R] [regress postestimation](#), [R] [rreg](#)

serial correlation, *see* autocorrelation

serial independence, test for, [R] [runtest](#)

serrbar command, [R] [serrbar](#)

session, recording, [R] [log](#)

set

adosize command, [R] [set](#)

autotabgraphs command, [R] [set](#)

checksum command, [R] [set](#)

command, [R] [query](#)

conren command, [R] [set](#)

copycolor command, [R] [set](#)

dockable command, [R] [set](#)

dockingguides command, [R] [set](#)

*set, continued*

doublebuffer command, [R] **set**  
 dp command, [R] **set**  
 emptycells command, [R] **set emptycells**; [R] **set**  
 eolchar command, [R] **set**  
 fastscroll command, [R] **set**  
 floatresults command, [R] **set**  
 floatwindows command, [R] **set**  
 graphics command, [R] **set**  
 httpproxy command, [R] **netio**; [R] **set**  
 httpproxyauth command, [R] **netio**; [R] **set**  
 httpproxyhost command, [R] **netio**; [R] **set**  
 httpproxyport command, [R] **netio**; [R] **set**  
 httpproxypw command, [R] **netio**; [R] **set**  
 httpproxyuser command, [R] **netio**; [R] **set**  
 level command, [R] **level**; [R] **set**  
 linegap command, [R] **set**  
 linesize command, [R] **log**; [R] **set**  
 locksplitters command, [R] **set**  
 logtype command, [R] **log**; [R] **set**  
 matacache command, [R] **set**  
 metafavor command, [R] **set**  
 matalibs command, [R] **set**  
 matalnum command, [R] **set**  
 matamofirst command, [R] **set**  
 mataoptimize command, [R] **set**  
 matastrict command, [R] **set**  
 matsize command, [R] **matsize**; [R] **set**  
 maxdb command, [R] **db**; [R] **set**  
 maxiter command, [R] **maximize**; [R] **set**  
 maxvar command, [R] **set**  
 memory command, [R] **set**  
 more command, [R] **more**; [R] **set**  
 notifyuser command, [R] **set**  
 obs command, [R] **set**  
 odbcmgr command, [R] **set**  
 output command, [R] **set**  
 pagesize command, [R] **more**; [R] **set**  
 persistfv command, [R] **set**  
 persistvtopic command, [R] **set**  
 pinnable command, [R] **set**  
 playsnd command, [R] **set**  
 printcolor command, [R] **set**  
 processors command, [R] **set**  
 reventries command, [R] **set**  
 revkeyboard command, [R] **set**  
 rmsg command, [R] **set**  
 scheme command, [R] **set**  
 scrollbufsize command, [R] **set**  
 searchdefault command, [R] **search**; [R] **set**  
 seed command, [R] **set seed**; [R] **set**  
 smoothfonts command, [R] **set**  
 timeout1 command, [R] **netio**; [R] **set**  
 timeout2 command, [R] **netio**; [R] **set**  
 trace command, [R] **set**  
 tracedepth command, [R] **set**  
 traceexpand command, [R] **set**

*set, continued*

tracehilite command, [R] **set**  
 traceindent command, [R] **set**  
 tracenumber command, [R] **set**  
 tracesep command, [R] **set**  
 type command, [R] **set**  
 update\_interval command, [R] **update**; [R] **set**  
 update\_prompt command, [R] **update**; [R] **set**  
 update\_query command, [R] **update**; [R] **set**  
 varabbrev command, [R] **set**  
 varkeyboard command, [R] **set**  
 varlabelpos command, [R] **set**  
 virtual command, [R] **set**  
 set ado, net subcommand, [R] **net**  
 set\_defaults command, [R] **set\_defaults**  
 set other, net subcommand, [R] **net**  
 sfrancia command, [R] **swilk**  
 Shapiro–Francia test for normality, [R] **swilk**  
 Shapiro–Wilk test for normality, [R] **swilk**  
 shewhart command, [R] **qc**  
 shownrtolerance option, [R] **maximize**  
 showstep option, [R] **maximize**  
 showtolerance option, [R] **maximize**  
 Šidák multiple comparison test, [R] **oneway**  
 significance levels, [R] **level**, [R] **query**  
 signrank command, [R] **signrank**  
 signtest command, [R] **signrank**  
 simulate prefix command, [R] **simulate**  
 simulations, Monte Carlo, [R] **simulate**; [R] **permute**  
 simultaneous  
     quantile regression, [R] **qreg**  
     systems, [R] **reg3**  
 sj, net subcommand, [R] **net**  
 SJ, *see Stata Journal and Stata Technical Bulletin*  
 skewed logistic regression, [R] **scobit**  
 skewness, [R] **summarize**; [R] **lnskew0**, [R] **lv**,  
     [R] **sktest**, [R] **tabstat**  
 sktest command, [R] **sktest**  
 slogit command, [R] **slogit**, [R] **slogit postestimation**,  
     *also see postestimation command*  
 smooth command, [R] **smooth**  
 smoothfonts, set subcommand, [R] **set**  
 smoothing, [R] **lpoly**, [R] **smooth**  
 smoothing graphs, [R] **kdensity**, [R] **lowess**  
 Spearman–Brown prophecy formula, [R] **alpha**  
 spearman command, [R] **spearman**  
 Spearman’s rho, [R] **spearman**  
 specification test, [R] **boxcox**, [R] **hausman**,  
     [R] **linktest**, [R] **regress postestimation**  
 specificity, [R] **logistic**  
 Spiegelhalter’s Z statistic, [R] **brier**  
 spike plot, [R] **spikeplot**  
 spikeplot command, [R] **spikeplot**  
 splines  
     linear, [R] **mkspline**  
     restricted cubic, [R] **mkspline**  
 split-plot designs, [R] **anova**

- spread, [R] [lv](#)
  - sqreg** command, [R] [qreg](#), [R] [qreg postestimation](#),  
also see [postestimation](#) command
  - sreturn** list command, [R] [saved results](#)
  - ssc**
    - copy command, [R] [ssc](#)
    - describe command, [R] [ssc](#)
    - hot command, [R] [ssc](#)
    - install command, [R] [ssc](#)
    - new command, [R] [ssc](#)
    - type command, [R] [ssc](#)
    - uninstall command, [R] [ssc](#)
  - SSC archive, [R] [ssc](#)
  - standard deviations,
    - displaying, [R] [lv](#), [R] [summarize](#), [R] [table](#),  
[R] [tabulate](#), [summarize\(\)](#)
    - testing equality of, [R] [sdtest](#)
  - standard errors,
    - for general predictions, [R] [predictnl](#)
    - forecast, [R] [predict](#), [R] [regress postestimation](#)
    - mean, [R] [ci](#), [R] [mean](#)
    - prediction, [R] [glm](#), [R] [predict](#), [R] [regress postestimation](#)
    - residuals, [R] [predict](#), [R] [regress postestimation](#)
    - robust, see [robust](#), [Huber/White/sandwich estimator](#) of variance
  - standardized
    - means, [R] [mean](#)
    - proportions, [R] [proportion](#)
    - rates, [R] [dstdize](#)
    - ratios, [R] [ratio](#)
    - residuals, [R] [binreg postestimation](#), [R] [glm postestimation](#), [R] [logistic postestimation](#),  
[R] [logit postestimation](#), [R] [predict](#), [R] [regress postestimation](#)
  - Stata Journal* and *Stata Technical Bulletin*
    - installation of, [R] [net](#), [R] [sj](#)
    - keyword search of, [R] [search](#)
  - stata.key** file, [R] [search](#)
  - Statistical Software Components (SSC) archive, [R] [ssc](#)
  - stats**, [estimates](#) subcommand, [R] [estimates stats](#)
  - stb**, [net](#) subcommand, [R] [net](#)
  - STB, see *Stata Journal* and *Stata Technical Bulletin*
  - stcox**, fractional polynomials, [R] [fracpoly](#), [R] [mfp](#)
  - stem** command, [R] [stem](#)
  - stem-and-leaf displays, [R] [stem](#)
  - stepwise estimation, [R] [stepwise](#)
  - stepwise** prefix command, [R] [stepwise](#)
  - stereotype logistic regression, [R] [slogit](#)
  - stochastic frontier models, [R] [frontier](#)
  - store**, [estimates](#) subcommand, [R] [estimates store](#)
  - storing and restoring estimation results, [R] [estimates store](#)
  - stratification, [R] [asclgit](#), [R] [clogit](#)
  - studentized residuals, [R] [predict](#), [R] [regress postestimation](#)
  - Student's  $t$  distribution
    - confidence interval for mean, [R] [ci](#), [R] [mean](#)
    - testing equality of means, [R] [ttest](#)
  - suest** command, [R] [suest](#); [R] [hausman](#)
  - summarize**,
    - [estat](#) subcommand, [R] [estat](#)
    - [misstable](#) subcommand, [R] [misstable](#)
  - summarize** command, [R] [summarize](#); [R] [tabulate](#),  
[summarize\(\)](#)
  - summarizing data, [R] [summarize](#), [R] [tabstat](#);  
[R] [lv](#), [R] [table](#), [R] [tabulate](#), [summarize\(\)](#),  
[R] [tabulate oneway](#), [R] [tabulate twoway](#)
  - summary statistics, see [descriptive statistics](#), displaying
  - summative (Likert) scales, [R] [alpha](#)
  - sums, over observations, [R] [summarize](#)
  - sunflower** command, [R] [sunflower](#)
  - sunflower** plots, [R] [sunflower](#)
  - sureg** command, [R] [sureg](#), [R] [sureg postestimation](#),  
also see [postestimation](#) command
  - survey sampling, see [cluster sampling](#)
  - survival analysis, [R] [expoisson](#), [R] [fracpoly](#),  
[R] [gllamm](#), [R] [glm](#), [R] [intreg](#), [R] [nbreg](#),  
[R] [oprobit](#), [R] [poisson](#), [R] [zip](#), [R] [ztp](#)
  - survival-time data, see [survival analysis](#)
  - SVAR, [postestimation](#), [R] [regress postestimation time series](#)
  - swap**, [update](#) subcommand, [R] [update](#)
  - swilk** command, [R] [swilk](#)
  - symbolic forms, [R] [anova](#)
  - symmetry, test of, [R] [symmetry](#)
  - symmetry** command, [R] [symmetry](#)
  - symmetry** plots, [R] [diagnostic plots](#)
  - symmi** command, [R] [symmetry](#)
  - symplo** command, [R] [diagnostic plots](#)
  - syntax diagrams explained, [R] [intro](#)
  - system
    - estimators, [R] [gmm](#), [R] [ivregress](#), [R] [nlsur](#),  
[R] [reg3](#), [R] [sureg](#)
    - parameters, [R] [query](#), [R] [set](#), [R] [set\\_defaults](#)
  - system1** command, [R] [gmm](#)
  - szoerter**, [estat](#) subcommand, [R] [regress postestimation](#)
  - Szroeter's test for heteroskedasticity, [R] [regress postestimation](#)
- ## T
- $t$  distribution
    - confidence interval for mean, [R] [ci](#), [R] [mean](#)
    - testing equality of means, [R] [ttest](#)
  - tab1** command, [R] [tabulate oneway](#)
  - tab2** command, [R] [tabulate twoway](#)
  - tabi** command, [R] [tabulate twoway](#)
  - table**, [estimates](#) subcommand, [R] [estimates table](#)
  - table** command, [R] [table](#)

tables,

- contingency, [R] [table](#), [R] [tabulate twoway](#)
- frequency, [R] [tabulate oneway](#), [R] [tabulate twoway](#); [R] [table](#), [R] [tabstat](#), [R] [tabulate](#), [R] [summarize\(\)](#)
- N-way, [R] [table](#)
- of means, [R] [table](#), [R] [tabulate](#), [R] [summarize\(\)](#)
- of statistics, [R] [table](#), [R] [tabstat](#)

tables of estimation results, [R] [estimates table](#)

tabstat command, [R] [tabstat](#)

tabulate command, [R] [tabulate oneway](#), [R] [tabulate twoway](#)

[summarize\(\)](#), [R] [tabulate](#), [R] [summarize\(\)](#)

tau, [R] [spearman](#)

TDT test, [R] [symmetry](#)

technique() option, [R] [maximize](#)

test command, [R] [anova postestimation](#), [R] [test](#)

testnl command, [R] [testnl](#)

testparm command, [R] [test](#)

tests,

ARCH effect, [R] [regress postestimation time series](#)

association, [R] [tabulate twoway](#)

binomial probability, [R] [bitest](#)

Breusch–Pagan, [R] [mvreg](#), [R] [sureg](#)

equality of

coefficients, [R] [test](#), [R] [testnl](#)

distributions, [R] [ksmirnov](#), [R] [kwallis](#), [R] [ranksum](#), [R] [signrank](#)

means, [R] [ttest](#)

medians, [R] [ranksum](#)

proportions, [R] [bitest](#), [R] [prtest](#)

variance, [R] [sdtest](#)

equivalence, [R] [pk](#), [R] [pkequiv](#)

heteroskedasticity, [R] [regress postestimation](#)

independence, [R] [tabulate twoway](#)

independence of irrelevant alternatives, see [IIA](#)

internal consistency, [R] [alpha](#)

interrater agreement, [R] [kappa](#)

kurtosis, [R] [regress postestimation](#), [R] [sktest](#)

likelihood-ratio, [R] [lrtest](#)

linear hypotheses after estimation, [R] [test](#)

marginal homogeneity, [R] [symmetry](#)

model coefficients, [R] [lrtest](#), [R] [test](#), [R] [testnl](#)

model specification, [R] [hausman](#), [R] [linktest](#)

nonlinear hypotheses after estimation, [R] [testnl](#)

normality, [R] [boxcox](#), [R] [ladder](#), [R] [sktest](#), [R] [swilk](#)

permutation, [R] [permute](#)

serial correlation, [R] [regress postestimation time series](#)

serial independence, [R] [runtest](#)

skewness, [R] [regress postestimation](#)

symmetry, [R] [symmetry](#)

TDT, [R] [symmetry](#)

trend, [R] [nptrend](#), [R] [symmetry](#)

variance-comparison, [R] [sdtest](#)

tetrachoric command, [R] [tetrachoric](#)

three-stage least squares, [R] [reg3](#)

time-series analysis, [R] [regress postestimation time series](#)

time-versus-concentration curve, [R] [pk](#)

timeout1, set subcommand, [R] [netio](#), [R] [set](#)

timeout2, set subcommand, [R] [netio](#), [R] [set](#)

title, estimates subcommand, [R] [estimates title](#)

tobit command, [R] [tobit](#), [R] [tobit postestimation](#), also see [postestimation](#) command

tobit regression, [R] [ivtobit](#), [R] [tobit](#), also see [intreg](#) command, [truncreg](#) command

.toc filename suffix, [R] [net](#)

tolerance() option, [R] [maximize](#)

total command, [R] [total](#), [R] [total postestimation](#), also see [postestimation](#) command

totals, estimation, [R] [total](#)

trace, ml subcommand, [R] [ml](#)

trace, set subcommand, [R] [set](#)

trace option, [R] [maximize](#)

tracedepth, set subcommand, [R] [set](#)

traceexpand, set subcommand, [R] [set](#)

tracehilite, set subcommand, [R] [set](#)

traceindent, set subcommand, [R] [set](#)

tracenumber, set subcommand, [R] [set](#)

traceseq, set subcommand, [R] [set](#)

tracing iterative maximization process, [R] [maximize](#)

transformations,

log, [R] [lnskew0](#)

modulus, [R] [boxcox](#)

power, [R] [boxcox](#), [R] [lnskew0](#)

transformations

to achieve normality, [R] [boxcox](#), [R] [ladder](#)

to achieve zero skewness, [R] [lnskew0](#)

translate command, [R] [translate](#)

translate logs, [R] [translate](#)

translator

query command, [R] [translate](#)

reset command, [R] [translate](#)

set command, [R] [translate](#)

transmap

define command, [R] [translate](#)

query command, [R] [translate](#)

transmission-disequilibrium test, [R] [symmetry](#)

treatment effects, [R] [treatreg](#)

treatreg command, [R] [treatreg](#), [R] [treatreg postestimation](#), also see [postestimation](#) command

tree, misstable subcommand, [R] [misstable](#)

trend, test for, [R] [nptrend](#), [R] [symmetry](#)

truncated-normal

model, stochastic frontier, [R] [frontier](#)

regression, [R] [truncreg](#)

truncreg command, [R] [truncreg](#), [R] [truncreg postestimation](#), also see [postestimation](#) command

ttest and ttesti commands, [R] [ttest](#)

tuning constant, [R] [rreg](#)



two-stage least squares, [R] [gmm](#), [R] [ivregress](#),  
[R] [nlsur](#), [R] [regress](#)

two-way

analysis of variance, [R] [anova](#)  
scatterplots, [R] [lowess](#)

type,

[set](#) subcommand, [R] [set](#)  
[ssc](#) subcommand, [R] [ssc](#)

## U

*U* statistic, [R] [ranksum](#)

uniformly distributed random-number function, [R] [set](#)  
[seed](#)

uninstall,

[net](#) subcommand, [R] [net](#)  
[ssc](#) subcommand, [R] [ssc](#)

unique values, counting, [R] [table](#), [R] [tabulate oneway](#)

univariate

distributions, displaying, [R] [cumul](#), [R] [diagnostic](#)  
[plots](#), [R] [histogram](#), [R] [ladder](#), [R] [lv](#),  
[R] [stem](#)

kernel density estimation, [R] [kdensity](#)

update, [view](#) subcommand, [R] [view](#)

update

[ado](#) command, [R] [update](#)  
[all](#) command, [R] [update](#)  
command, [R] [update](#)  
executable command, [R] [update](#)  
from command, [R] [update](#)  
query command, [R] [update](#)  
swap command, [R] [update](#)  
utilities command, [R] [update](#)

[update\\_d](#), [view](#) subcommand, [R] [view](#)

[update\\_interval](#), [set](#) subcommand, [R] [set](#),  
[R] [update](#)

[update\\_prompt](#), [set](#) subcommand, [R] [set](#),  
[R] [update](#)

[update\\_query](#), [set](#) subcommand, [R] [set](#), [R] [update](#)

updates to Stata, [R] [adoupdate](#), [R] [net](#), [R] [sj](#),  
[R] [update](#)

use, [estimates](#) subcommand, [R] [estimates save](#)

user-written additions,

installing, [R] [net](#), [R] [ssc](#)  
searching for, [R] [net search](#), [R] [ssc](#)

utilities, [update](#) subcommand, [R] [update](#)

## V

[varabbrev](#), [set](#) subcommand, [R] [set](#)

variables,

categorical, see categorical data  
dummy, see indicator variables  
factor, see factor variables  
in model, maximum number, [R] [matsize](#)  
orthogonalize, [R] [orthog](#)

variance,

analysis of, [R] [anova](#), [R] [loneway](#), [R] [oneway](#)  
displaying, [R] [summarize](#), [R] [table](#), [R] [tabulate](#),  
[summarize\(\)](#); [R] [lv](#)

Huber/White/sandwich estimator, see robust,

Huber/White/sandwich estimator of variance

nonconstant, see robust, Huber/White/sandwich  
estimator of variance

testing equality of, [R] [sdtest](#)

variance

estimators, [R] [vce\\_option](#)

inflation factors, [R] [regress postestimation](#)

stabilizing transformations, [R] [boxcox](#)

variance-comparison test, [R] [sdtest](#)

variance-covariance matrix of estimators, [R] [correlate](#),  
[R] [estat](#)

variance-weighted least squares, [R] [vwls](#)

[varkeyboard](#), [set](#) subcommand, [R] [set](#)

[varlabelpos](#), [set](#) subcommand, [R] [set](#)

[vce\(\)](#) option, [R] [vce\\_option](#)

[vce](#), [estat](#) subcommand, [R] [estat](#)

version of ado-file, [R] [which](#)

version of Stata, [R] [about](#)

[view](#)

[ado](#) command, [R] [view](#)

[ado\\_d](#) command, [R] [view](#)

[browse](#) command, [R] [view](#)

command, [R] [view](#)

[help](#) command, [R] [view](#)

[help\\_d](#) command, [R] [view](#)

[net](#) command, [R] [view](#)

[net\\_d](#) command, [R] [view](#)

[news](#) command, [R] [view](#)

[search](#) command, [R] [view](#)

[search\\_d](#) command, [R] [view](#)

[update](#) command, [R] [view](#)

[update\\_d](#) command, [R] [view](#)

[view\\_d](#) command, [R] [view](#)

[view\\_d](#), [view](#) subcommand, [R] [view](#)

viewing previously typed lines, [R] [#review](#)

[vif](#), [estat](#) subcommand, [R] [regress postestimation](#)

virtual, [set](#) subcommand, [R] [set](#)

[vwls](#) command, [R] [vwls](#), [R] [vwls postestimation](#),  
also see [postestimation](#) command

## W

Wald tests, [R] [predictnl](#), [R] [test](#), [R] [testnl](#)

weighted least squares, [R] [gmm](#), [R] [ivregress](#),  
[R] [nlsur](#), [R] [regress](#), [R] [regress](#)  
[postestimation](#), [R] [vwls](#)

Welsch distance, [R] [regress postestimation](#)

[which](#) command, [R] [which](#)

White/Huber/sandwich estimator of variance, see robust,  
Huber/White/sandwich estimator of variance

White's test for heteroskedasticity, [R] [regress](#)  
[postestimation](#)

Wilcoxon

rank-sum test, [R] [ranksum](#)

signed-ranks test, [R] [signrank](#)

## X

`xchart` command, [R] [qc](#)

`xi` prefix command, [R] [xi](#)

## Z

Zellner's seemingly unrelated regression, [R] [sureg](#);

[R] [reg3](#), [R] [suest](#)

zero-altered

negative binomial regression, [R] [zinb](#)

Poisson regression, [R] [zip](#)

zero-inflated

negative binomial regression, [R] [zinb](#)

Poisson regression, [R] [zip](#)

zero-skewness transform, [R] [lnskew0](#)

zero-truncated

negative binomial regression, [R] [ztnb](#)

Poisson regression, [R] [ztp](#)

`zinb` command, [R] [zinb](#), [R] [zinb postestimation](#), *also*  
see `postestimation` command

`zip` command, [R] [zip](#), [R] [zip postestimation](#), *also* see  
`postestimation` command

`ztnb` command, [R] [ztnb](#), [R] [ztnb postestimation](#),  
*also* see `postestimation` command

`ztp` command, [R] [ztp](#), [R] [ztp postestimation](#), *also*  
see `postestimation` command



