1 Read this—it will help

Contents

A Complete Stata Documentation Set contains more than 12,000 pages of information in the following manuals:

[GSM]	*Getting Started with Stata for Mac*
[GSU]	*Getting Started with Stata for Unix*
[GSW]	*Getting Started with Stata for Windows*
[U]	*Stata User's Guide*
[R]	*Stata Base Reference Manual*
[BAYES]	*Stata Bayesian Analysis Reference Manual*
[D]	*Stata Data Management Reference Manual*
[FN]	*Stata Functions Reference Manual*
[G]	*Stata Graphics Reference Manual*
[IRT]	*Stata Item Response Theory Reference Manual*
[XT]	*Stata Longitudinal-Data/Panel-Data Reference Manual*
[ME]	*Stata Multilevel Mixed-Effects Reference Manual*
[MI]	*Stata Multiple-Imputation Reference Manual*
[MV]	*Stata Multivariate Statistics Reference Manual*
[PSS]	*Stata Power and Sample-Size Reference Manual*
[P]	*Stata Programming Reference Manual*
[SEM]	*Stata Structural Equation Modeling Reference Manual*
[SVY]	*Stata Survey Data Reference Manual*
[ST]	*Stata Survival Analysis Reference Manual*
[TS]	*Stata Time-Series Reference Manual*
[TE]	*Stata Treatment-Effects Reference Manual: Potential Outcomes/Counterfactual Outcomes*
[I]	*Stata Glossary and Index*
[M]	*Mata Reference Manual*

In addition, installation instructions may be found in the *Installation Guide*, which comes in the DVD case.

1.1 Getting Started with Stata

There are three *Getting Started* manuals:

[GSM] *Getting Started with Stata for Mac*
[GSU] *Getting Started with Stata for Unix*
[GSW] *Getting Started with Stata for Windows*

1. Learn how to use Stata—read the *Getting Started* (GSM, GSU, or GSW) manual.
2. Now turn to the other manuals; see [U] **1.2 The User's Guide and the Reference manuals**.

1.2 The User's Guide and the Reference manuals

The *User's Guide* is divided into three sections: *Stata basics*, *Elements of Stata*, and *Advice*. The table of contents lists the chapters within each of these sections. Click on the chapter titles to see the detailed contents of each chapter.

The *Guide* is full of a lot of useful information about Stata; we recommend that you read it. If you only have time, however, to read one or two chapters, then read [U] **11 Language syntax** and [U] **12 Data**.

The other manuals are the *Reference* manuals. The Stata *Reference* manuals are each arranged like an encyclopedia—alphabetically. Look at the *Base Reference Manual*. Look under the name of a command. If you do not find the command, look in the subject index in [I] *Stata Glossary and Index*. A few commands are so closely related that they are documented together, such as `ranksum` and `median`, which are both documented in [R] **ranksum**.

Not all the entries in the *Base Reference Manual* are Stata commands; some contain technical information, such as [R] **maximize**, which details Stata's iterative maximization process, or [R] **error messages**, which provides information on error messages and return codes.

Like an encyclopedia, the *Reference* manuals are not designed to be read from cover to cover. When you want to know what a command does, complete with all the details, qualifications, and pitfalls, or when a command produces an unexpected result, read its description. Each entry is written at the level of the command. The descriptions assume that you have little knowledge of Stata's features when they are explaining simple commands, such as those for using and saving data. For more complicated commands, they assume that you have a firm grasp of Stata's other features.

If a Stata command is not in the *Base Reference Manual*, you can find it in one of the other *Reference* manuals. The titles of the manuals indicate the types of commands that they contain. The *Programming Reference Manual*, however, contains commands not only for programming Stata but also for manipulating matrices (not to be confused with the matrix programming language described in the *Mata Reference Manual*).

1.2.1 PDF manuals

Every copy of Stata comes with Stata's complete PDF documentation.

The PDF documentation may be accessed from within Stata by selecting **Help > PDF documentation**. Even more convenient, every help file in Stata links to the equivalent manual entry. If you are reading **help regress**, simply click on [R] **regress** in the `Title` section of the help file to go directly to the [R] **regress** manual entry.

We provide recommended settings for your PDF viewer to optimize it for Stata's documentation at http://www.stata.com/support/faqs/res/documentation.html.

1.2.1.1 Video example

PDF documentation in Stata

1.2.2 Example datasets

Various examples in this manual use what is referred to as the automobile dataset, `auto.dta`. We have created a dataset on the prices, mileages, weights, and other characteristics of 74 automobiles and have saved it in a file called `auto.dta`. (These data originally came from the April 1979 issue of *Consumer Reports* and from the United States Government EPA statistics on fuel consumption; they were compiled and published by Chambers et al. [1983].)

In our examples, you will often see us type

```
    . use http://www.stata-press.com/data/r14/auto
```

We include the `auto.dta` file with Stata. If you want to use it from your own computer rather than via the Internet, you can type

```
    . sysuse auto
```

See [D] **sysuse**.

You can also access `auto.dta` by selecting **File > Example datasets...**, clicking on *Example datasets installed with Stata*, and clicking on `use` beside the `auto.dta` filename.

There are many other example datasets that ship with Stata or are available over the web. Here is a partial list of the example datasets included with Stata:

`auto.dta`	1978 Automobile Data
`auto2.dta`	1978 Automobile Data
`autornd.dta`	Subset of 1978 Automobile Data
`bplong.dta`	fictional blood pressure data
`bpwide.dta`	fictional blood pressure data
`cancer.dta`	Patient Survival in Drug Trial
`census.dta`	1980 Census data by state
`citytemp.dta`	City Temperature Data
`citytemp4.dta`	City Temperature Data
`educ99gdp.dta`	Education and GDP
`gnp96.dta`	U.S. GNP, 1967–2002
`lifeexp.dta`	Life expectancy, 1998
`network1.dta`	fictional network diagram data
`network1a.dta`	fictional network diagram data
`nlsw88.dta`	U.S. National Longitudinal Study of Young Women (NLSW, 1988 extract)
`nlswide1.dta`	U.S. National Longitudinal Study of Young Women (NLSW, 1988 extract)
`pop2000.dta`	U.S. Census, 2000, extract
`sandstone.dta`	Subsea elevation of Lamont sandstone in an area of Ohio
`sp500.dta`	S&P 500
`surface.dta`	NOAA Sea Surface Temperature
`tsline1.dta`	simulated time-series data
`tsline2.dta`	fictional data on calories consumed
`uslifeexp.dta`	U.S. life expectancy, 1900–1999
`uslifeexp2.dta`	U.S. life expectancy, 1900–1940
`voter.dta`	1992 presidential voter data
`xtline1.dta`	fictional data on calories consumed

All of these datasets may be used or described from the **Example datasets...** menu listing.

Even more example datasets, including most of the datasets used in the reference manuals, are available at the Stata Press website (http://www.stata-press.com/data/). You can download the datasets with your browser, or you can use them directly from the Stata command line:

```
. use http://www.stata-press.com/data/r14/nlswork
```

An alternative to the use command for these example datasets is webuse. For example, typing

```
. webuse nlswork
```

is equivalent to the above use command. For more information, see [D] **webuse**.

1.2.2.1 Video example

Example data included with Stata

1.2.3 Cross-referencing

The *Getting Started* manual, the *User's Guide*, and the *Reference* manuals cross-reference each other.

[R] **regress**
[D] **reshape**
[XT] **xtreg**

The first is a reference to the regress entry in the *Base Reference Manual*, the second is a reference to the reshape entry in the *Data Management Reference Manual*, and the third is a reference to the xtreg entry in the *Longitudinal-Data/Panel-Data Reference Manual*.

[GSW] **B Advanced Stata usage**
[GSM] **B Advanced Stata usage**
[GSU] **B Advanced Stata usage**

are instructions to see the appropriate section of the *Getting Started with Stata for Windows*, *Getting Started with Stata for Mac*, or *Getting Started with Stata for Unix* manual.

1.2.4 The index

The *Glossary and Index* contains a combined index for all the manuals.

To find information and commands quickly, you can use Stata's search command; see [R] **search**. At the Stata command prompt, type search geometric mean. search searches Stata's keyword database and the Internet to find more commands and extensions for Stata written by Stata users.

1.2.5 The subject table of contents

A subject table of contents for the *User's Guide* and all the *Reference* manuals except the *Mata Reference Manual* is located in the *Glossary and Index*. This subject table of contents may also be accessed by clicking on **Contents** in the PDF bookmarks.

1.2.6 Typography

We mix the ordinary typeface that you are reading now with a typewriter-style typeface `that looks like this`. When something is printed in the typewriter-style typeface, it means that something is a command or an option—it is something that Stata understands and something that you might actually type into your computer. Differences in typeface are important. If a sentence reads, "You could list the result ...", it is just an English sentence—you *could* list the result, but the sentence provides no clue as to how you might actually do that. On the other hand, if the sentence reads, "You could `list` the result ...", it is telling you much more—you could list the result, and you could do that by using the `list` command.

We will occasionally lapse into periods of inordinate cuteness and write, "We `describe`d the data and then `list`ed the data." You get the idea. `describe` and `list` are Stata commands. We purposely began the previous sentence with a lowercase letter. Because `describe` is a Stata command, it must be typed in lowercase letters. The ordinary rules of capitalization are temporarily suspended in favor of preciseness.

We also mix in words printed in italic type, such as "To perform the rank-sum test, type `ranksum` *varname*`, by(`*groupvar*`)`". Italicized words are not supposed to be typed; instead, you are to substitute another word for them.

We would also like users to note our rule for punctuation of quotes. We follow a rule that is often used in mathematics books and British literature. The punctuation mark at the end of the quote is included in the quote only if it is a part of the quote. For instance, the pleased Stata user said she thought that Stata was a "very powerful program". Another user simply said, "I love Stata."

In this manual, however, there is little dialogue, and we follow this rule to precisely clarify what you are to type, as in, type "`cd c:`". The period is outside the quotation mark because you should not type the period. If we had wanted you to type the period, we would have included two periods at the end of the sentence: one inside the quotation and one outside, as in, type "the orthogonal polynomial operator, `p.`".

We have tried not to violate the other rules of English. If you find such violations, they were unintentional and resulted from our own ignorance or carelessness. We would appreciate hearing about them.

We have heard from Nicholas J. Cox of the Department of Geography at Durham University, UK, and express our appreciation. His efforts have gone far beyond dropping us a note, and there is no way with words that we can fully express our gratitude.

1.2.7 Vignette

If you look, for example, at the entry [R] **brier**, you will see a brief biographical vignette of Glenn Wilson Brier (1913–1998), who did pioneering work on the measures described in that entry. A few such vignettes were added without fanfare in the Stata 8 manuals, just for interest, and many more were added in Stata 9, and even more have been added in each subsequent release. A vignette could often appropriately go in several entries. For example, George E. P. Box deserves to be mentioned in entries other than [TS] **arima**, such as [R] **boxcox**. However, to save space, each vignette is given once only, and an index of all vignettes is given in the *Glossary and Index*.

Most of the vignettes were written by Nicholas J. Cox, Durham University, and were compiled using a wide range of reference books, articles in the literature, Internet sources, and information from individuals. Especially useful were the dictionaries of Upton and Cook (2014) and Everitt and Skrondal (2010) and the compilations of statistical biographies edited by Heyde and Seneta (2001) and Johnson and Kotz (1997). Of these, only the first provides information on people living at the time of publication.

1.3 What's new

This section is intended for users of the previous version of Stata. If you are new to Stata, you may as well skip to [U] **1.3.19 What's more**.

As always, Stata 14 is 100% compatible with the previous releases, but we remind programmers that it is important to put `version 14`, `version 13.1`, or `version 12`, etc., at the top of old do- and ado-files so that they continue to work as you expect. You were supposed to do that when you wrote them, but if you did not, go back and do it now.

We will list all the changes, item by item, but first, here are the highlights.

1.3.1 What's new (highlights)

Here are the highlights. There are more, and do not assume that because we mention a category, we have mentioned everything new in the category. Detailed sections follow the highlights.

1. **Unicode support**

    `Здравствуйте`. こんにちは. Hello.

    Stata 14 supports Unicode (UTF-8). All of Stata is Unicode aware. You may use Unicode for variable names, labels, data, and whatever else you wish. Not only do you have more characters from which to choose, but when you share data, others will see what you see.

    Warning . . . Files may need translating

    If you previously used Extended ASCII to overcome the limitations of plain ASCII in your `.dta` files, do-files, and ado-files, they need to be translated from Extended ASCII to Unicode. We have made that easy: see [D] **unicode translate**. If you do not translate your files, they will not display properly. If you used Extended ASCII for variable names, you may not even be able to type the mangled names!

    New string functions

    Stata 14 has three times the number of string functions as Stata 13. To understand why, let's start with what you may find a surprising side effect of using Unicode. Say you had an `str3` variable and you replaced a value in one observation with "für". The variable would be `str4` after the change! It would be `str4` because it takes four memory positions (bytes) to store "für": one each for "f" and "r" and two for "ü".

    The standard ASCII characters consume one memory position just as they did previously, but the other Unicode characters need two, three, or even four memory positions. `strlen()` works in terms of memory positions, so `strlen("für")` reports 4 positions, not 3 characters. New function `ustrlen("für")` works in terms of character positions, so it reports 3 characters. `strlen("こんにちは")` is 15 if you can believe it, but `ustrlen("こんにちは")` is a reassuring 5. By the way, こんにちは is pronounced "Kon'nichiwa" and means "hello".

    Anyway, for each string function, *fcn*`()`, that needs it, there is a new, corresponding function, `u`*fcn*`()`. `u`*fcn*`()` uses the character-position metric instead of the memory-position metric. As another example, `usubstr(`*s*`, 2, 3)` returns up to three characters starting at the second. `substr(`*s*`, 2, 3)` returns up to three memory positions (bytes) starting at the second. Memory positions are the same as characters only if *s* is ASCII.

    If you are writing for an international audience, you need to distinguish between the two flavors of each string function.

The third new group of string functions are the `ud`*fcn*`()`s. They work in the display-position metric. `udstrlen("für")` is 3, meaning it takes 3 columns to display "für". `udstrlen("こんにちは")` is 10, meaning it takes 10 columns to display "こんにちは". You use the new `ud`*fcn*`()`s when you are aligning output in a table. `udsubstr(`*s*`, 2, 3)` returns however many characters it takes to fill up to three columns, starting after the second character.

Graphs, SEM Builder, Unicode, and Extended ASCII

Export graphs or output containing Unicode using PDF instead of PostScript (PS) or Encapsulated PostScript (EPS). PS and EPS do not support Unicode. In some cases, you can use PS and EPS because Stata converts accented Latin characters to the Extended ASCII characters that PS and EPS expect.

If you have Stata 13 or earlier `.gph` graph files or `.stsem` SEM Builder files, and those files contain Extended ASCII, Stata 14 will not display the Extended ASCII characters correctly, and they cannot be translated. You can edit them.

See [U] **12.4.2 Handling Unicode strings** and see [D] **unicode** for more information on Stata 14's new Unicode capabilities.

2. **More than 2 billion observations now allowed**

Stata/MP, the multiprocessor version of Stata 14, now allows more than 2 billion observations, or more correctly, more than 2,147,483,620 observations. The maximum now depends solely on the amount of memory on your computer. Stata will not limit you; it can now count up to 281 trillion observations.

How many observations you can process depends on the size of your computer and the width of your data. Here are some sample calculations and a formula:

Computer's memory	Memory used	Billions of observations scenario (1)	(2)	(3)
128GB	112GB	1.8	1.4	1.0
256GB	240GB	3.8	2.9	2.1
512GB	496GB	7.9	6.1	4.4
1024GB	1008GB	16.2	12.3	9.8
1536GB	1520GB	24.4	18.5	13.6

Notes:

Memory used is the total used for storing data. We left 16GB free for Stata and other processes, meaning that we assumed that Stata consumes nearly all the computer's resources (single user).

Observations leaves extra room for adding three doubles, because Stata commands often add working variables. The width used by the three scenarios is for your data exclusive of working variables.

Scenario 1: width = 43 bytes (same as `auto.dta`)
Scenario 2: width = 64 bytes
Scenario 3: width = 96 bytes

Calculation:

$$obs = \frac{memory\_used}{width + 24} \times \frac{1024^3}{1000^3}$$

where *memory_used* is in gigabytes and *obs* is in billions.

There is nothing more to know except that we have advice on how to improve Stata's performance when processing datasets with more than 2 billion observations; see `help obs advice`.

3. **Bayesian statistical analysis**

Stata 14 provides Bayesian statistical analyses with the new `bayesmh` command and corresponding suite of features. The *mh* on the end of `bayesmh` stands for Metropolis–Hastings. You can fit models by using an adaptive Metropolis–Hastings algorithm, or a full Gibbs sampling for some models, or a combination of the two algorithms. After estimation, you can diagnose convergence and analyze results.

Fitting a model can be as easy as typing

```
. bayesmh y x, likelihood(logit) prior({y:}, normal(0,100))
```

You can use our suite of preprogrammed likelihood models, or you can write your own. Postestimation features are the same either way. And even if you write your own models, you can still use the built-in priors.

We provide 12 built-in likelihood models and 22 built-in prior distributions. Built in are continuous, binary, ordinal, and count likelihood models. Built in are continuous univariate, continuous multivariate, discrete, and more prior distributions. Supported are univariate, multivariate, and multiple-equation models, including linear and nonlinear models and including generalized nonlinear models.

Results are reported with credible intervals (CrIs). You can check convergence visually by typing `bayesgraph diagnostics _all`. You can check Markov chain Monte Carlo (MCMC) efficiency by using the new `bayesstats ess` command.

You can obtain estimates of the posterior means and their MCMC standard errors not only for model parameters but also for functions of model parameters.

You can compare models using Bayesian information criteria such as the deviance information criterion or Bayes factors.

You can perform interval hypothesis testing by computing probabilities that a parameter or set of parameters or even functions of parameters belong to a specified range.

You can perform model hypothesis testing by computing probabilities of models given the observed data, which is to say, using model posterior probabilities.

And you can store your MCMC and estimation results for later analysis.

We provide an entire, new manual on all of this; see the *Stata Bayesian Analysis Reference Manual*.

4. **IRT models**

IRT stands for item response theory. IRT models explore the relationship between a latent (unobserved) trait and items that measure aspects of the trait. This often arises in standardized testing. A set of items (questions) is designed, the responses to which measure, say, the unobservable trait mathematical ability. Or questions are designed to measure unobservable cognitive abilities, personality traits, attitudes, health outcomes, quality of life, morale, and so on. The observable items do not have to be responses to questions, but they usually are. The items can be any observable variables that we believe measure the trait.

Stata can fit models for binary items, ordinal items, or categorical items. These include, for binary items, one-parameter logistic (1PL), two-parameter logistic (2PL), and three-parameter logistic (3PL); for ordinal items, graded response models, rating scale models, and partial credit models; and for categorical items, nominal response models. Stata can also fit hybrid models where different items use different models.

Once a model is fit, Stata can graph item characteristic curves (ICCs), test characteristic curves (TCCs), item information functions (IIFs), and test information functions (TIFs).

Stata includes a control panel to guide you through the fitting and analysis of models.

There is a lot more to say; see the all-new *Stata Item Response Theory Reference Manual*.

5. **Panel-data survival models**

   New estimation command `xtstreg` fits parametric panel-data survival models with random effects. Five distributions are provided: exponential, loglogistic, Weibull, lognormal, and gamma. Estimation is in the accelerated failure-time metric, but exponential and Weibull also allow the proportional-hazards metric.

   `xtstreg` is both an xt and an st command; you `xtset` the panel characteristics of the data and `stset` the survival characteristics. Hence, single- and multiple-record st data as well as all the other survival-data features are supported, and survivor, hazard, and cumulative hazard functions can be graphed using `stcurve`.

   Frequency, importance, and probability sampling weights are allowed.

   See [XT] **xtstreg**.

   New estimation command `mestreg` fits panel-data models with random coefficients and random intercepts. See [ME] **mestreg**.

6. **New in treatment effects**

   Stata 14 provides many new features for fitting and evaluating treatment effects. Treatment effects seek to extract experimental-style causal effects from observational data.

Treatment effects for survival models

New estimator `stteffects ra` estimates average treatment effects (ATEs), average treatment effects among the treated (ATETs), and potential-outcome means (POMs) via regression adjustment. See [TE] **stteffects ra**.

New estimator `stteffects ipw` estimates ATEs, ATETs, and POMs via inverse-probability weighting. See [TE] **stteffects ipw**.

New estimator `stteffects ipwra` estimates ATEs, ATETs, and POMs via inverse-probability-weighted regression adjustment. See [TE] **stteffects ipwra**.

New estimator `stteffects wra` estimates ATEs, ATETs, and POMs via weighted regression adjustment. See [TE] **stteffects wra**.

All the new estimators allow probability sampling weights.

Endogenous treatments

Stata 14 has a new estimator for endogenous treatments. Endogenous treatments arise when both the treatment model and the outcome model share unobserved covariates.

`etteffects` estimates ATEs, ATETs, and POMs for continuous, binary, count, fractional, and nonnegative outcomes when treatment assignment is correlated with outcome. See [TE] **eteffects**.

Probability weights

All the new estimators listed above support probability sampling weights. Support is also provided for [TE] **teffects ipwra**, [TE] **teffects ipw**, and [TE] **teffects ra**.

Balance analysis

Stata 14 performs balance analysis for treatment effects. A key requirement is that our treatment-effects model explicitly or implicitly reweights the data such that the treated and the untreated groups have comparable covariate values. Four new commands are provided to assess and to test balance.

`tebalance summarize` reports model-adjusted means and variances of covariates for the treated and untreated. See [TE] **tebalance summarize**.

`tebalance density` graphs kernel density plots for the model-adjusted data for the treated and untreated. See [TE] **tebalance density**.

`tebalance box` graphs box plots for the model-adjusted data for the treated and untreated. See [TE] **tebalance box**.

`tebalance overid` tests for covariate balance. See [TE] **tebalance overid**.

7. **Multilevel mixed-effects parametric survival models**

   New command `mestreg` fits multilevel mixed-effects parametric survival models.

   Five distributions are supported: exponential, loglogistic, Weibull, lognormal, and gamma.

   Both proportional-hazards and accelerated failure-time parameterizations are supported.

   Random effects, including random intercepts and random coefficients, at different levels of hierarchy are supported.

   Single- or multiple-record st data as well as other survival-data features are supported via the `stset` command. See [ST] **stset**.

   Survey data are supported via the `svy` prefix. See [SVY] **svy**.

   Relationships between multiple random effects can be independent or freely correlated, or you can specify the covariance structure.

   Postestimation statistics include predictions of mean and median survival times, and hazard and survivor functions. Predictions can be obtained conditionally or unconditionally on random effects.

   Survivor, hazard, and cumulative hazard functions can be graphed using the `stcurve` command. See [ST] **stcurve**.

   See [ME] **mestreg**.

8. **Small-sample inference for fixed effects in linear multilevel mixed models**

   Existing command `mixed` fits linear multilevel mixed models. `mixed` reports asymptotic test statistics for fixed effects by default. Those statistics have large-sample normal and χ^2 distributions. See [ME] **mixed**.

   When groups are balanced and sample size is small, these test statistics have exact t and F distributions for certain classes of models. In other situations, such as when groups are unbalanced, the sampling distributions of the statistics may be approximated by t and F distributions. Approximations differ in how (denominator) degrees of freedom are computed. The small-sample tests may yield better coverages.

New option dfmethod(*method*) provides various degree-of-freedom adjustments. Five methods are provided, including Kenward–Roger and Satterthwaite methods.

New postestimation command estat df reports the degrees of freedom for each coefficient.

test, testparm, and lincom have new option small to perform small-sample inference for fixed effects.

See *Small-sample inference for fixed effects* in [ME] **mixed**.

9. **New SEM (structural equation modeling) features**

   Existing commands sem and gsem provide the following new features:

Survival models

gsem now fits parametric survival models. With the new multilevel survival models previously mentioned, you might wonder why you would care. You care because SEM can fit multivariate models including survival models with unobserved components (latent variables), and combine survival models with other models involving continuous, binary, count, and other kinds of outcomes.

Five families are added to gsem: exponential, loglogistic, lognormal, Weibull, and gamma. Options are added for specifying right-censoring and left-truncation, as is common (and necessary) for analyzing survival times. See [SEM] **sem option method( )**.

predict after gsem has new option survival for computing survival-time predictions. The predicted survival function is computed using the current outcome values and the estimated parameters.

All the new families support the accelerated failure-time metric. Exponential and Weibull also support the proportional-hazards metric.

If you use the SEM Builder, simply select one of the survival families from the contextual toolbar.

Satorra–Bentler scaled χ^2 test

sem now provides the Satorra–Bentler scaled χ^2 model-versus-saturated test; specify new option vce(sbentler). In addition, corresponding robust standard errors (SEs) are produced and reported. This test and the SEs are robust to nonnormal distributions. These are an alternative to the previously provided robust SEs. Goodness-of-fit statistics based on the model χ^2 are also adjusted.

Support for survey data

gsem now supports the svy prefix for analyzing survey data, which includes multilevel weights. See [SVY] **svy**.

All the postestimation features available after survey estimation are also available. See [SVY] **svy postestimation** and [SVY] **estat**.

Support for observational and multilevel weights

gsem now supports observation-level weights and multilevel weights, even outside of a survey-data context.

Beta distribution

gsem adds the beta distribution to the choice of families and may be used with logit, probit, and cloglog links. The beta distribution is particularly appropriate for fractional or proportion data.

See *Stata Structural Equation Modeling Reference Manual*.

gsem provides the following new postestimation features:

predict has new options density and distribution for computing the density and distribution function for each outcome using their current values and the estimated parameters.

predict has new option marginal for computing observed endogenous predictions that are marginal with respect to the latent variables, meaning that prediction is produced by integrating over the distribution of the latent variable(s).

predict has new option expression() that calculates linear and nonlinear functions of the mean and linear predictions.

See [SEM] **predict after gsem**.

10. **Power analysis for survival and epidemiological methods**

Existing command power now provides all-new power analysis for epidemiological methods.

In addition, existing command stpower for performing power analysis on survival models is now undocumented, and its capabilities are now folded into existing command power. The advantage is that extensive graphing and tabulation of results are now available.

Here are the details. We will start with the survival models:

New methods for analysis of survival models

power cox estimates required sample size, power, and effect size using Cox proportional hazards models allowing for multiple covariates. It allows for correlation between the covariate of interest and other covariates, and it allows for withdrawal of subjects from the study. See [PSS] **power cox**.

power exponential estimates required sample size and power comparing two exponential survivor functions. It accommodates unequal allocation between the two groups, flexible accrual of subjects into the study (uniform and truncated exponential), and group-specific losses to follow-up. See [PSS] **power exponential**.

power logrank estimates required sample size, power, and effect size for comparing survivor functions in two groups using the log-rank test. It provides options to account for unequal allocation of subjects between the groups, possible withdrawal of subjects from the study (loss to follow-up), and uniform accrual of subjects into the study. See [PSS] **power logrank**.

As with all other power methods, cox, exponential, and logrank allow you to specify multiple values of parameters and automatically produce tabular and graphical results.

New methods for analysis of contingency tables

This will be of special interest to epidemiological researchers.

power cmh performs power and sample-size analysis for a Cochran–Mantel–Haenszel test of association in stratified 2×2 tables. It computes sample size, power, or effect size (common odds ratio) given other study parameters. It provides computations for designs with unbalanced stratum sizes as well as unbalanced group sizes within each stratum. See [PSS] **power cmh**.

power mcc performs power and sample-size analysis for a test of association between a risk factor and a disease in 1:M matched case–control studies. It computes sample size, power, or effect size (odds ratio) given other study parameters. See [PSS] **power cmh**.

power trend performs power and sample-size analysis for a Cochran–Armitage test of a linear trend in a probability of response in $J \times 2$ tables. The rows of the table correspond to ordinal exposure levels. The command computes sample size or power given other study parameters. It provides computations for unbalanced designs and for unequally spaced exposure levels (doses). With equally spaced exposure levels, a continuity correction is available. See [PSS] **power trend**.

As with all other `power` methods, `cmh`, `mcc`, and `trend` allow you to specify multiple values of parameters and automatically produce tabular and graphical results.

See [PSS] **power**.

11. **Markov-switching regression models**

New estimation command mswitch fits Markov-switching models. These are times-series models in which some of or all the parameters of a regression probabilistically transition among a finite set of unobserved states with unknown transition points.

Let's consider some examples. In economics, switching regression has been used to model growth rate of GDP to model asymmetric behavior observed over expansions and recessions.

In finance, switching has been used to model monthly stock returns.

In political science, switching has been used to model transitions between Democratic and Republican partisanship.

In psychology, switching has been used to model transitions between manic and depressive states.

In epidemiology, switching has been used to model the incidence rate of infectious disease in epidemic and nonepidemic states.

`mswitch` provides two ways of modeling the switching process: autoregressive (AR) and dynamic regression (DR). AR is typically used for processes that change slowly, and DR is typically used for processes that transition rapidly. See [TS] **mswitch**.

New postestimation commands estat transition and estat duration report the transition probabilities and expected state durations. See [TS] **mswitch postestimation**.

After estimation, you can `predict` the dependent variable, and you can also predict the probability of being in each of the unobserved states. These predictions are available as one-step ahead (static) or multi-step ahead (dynamic).

12. **Tests for structural breaks in time-series data**

Two new postestimation commands test for structural breaks after estimation by regress or ivregress.

estat sbknown tests for structural breaks at known dates.

estat sbsingle tests for a structural break at an unknown date.

See [TS] **estat sbknown** and [TS] **estat sbsingle**.

13. **Regression models for fractional data**

Two new estimation commands are provided for fitting models when the dependent variable is a fraction, a proportion, or a rate.

New estimation command fracreg allows the dependent variable to be in the range 0 to 1 inclusive. It fits probit, logit, and heteroskedastic probit models. See [R] **fracreg**.

New estimation command betareg requires the dependent variable to be in the range 0 to 1 exclusive. It fits beta regression. See [R] **betareg**.

Both new estimators support the svy prefix for analyzing survey data.

14. **Survey support and multilevel weights for multilevel models**

    We use the term "survey support" to include both Stata's svy prefix and multilevel probability weights outside of the survey context.

    The following estimation commands now support the svy prefix using the linearized estimate of the variance–covariance estimate: mecloglog, meglm, melogit, menbreg, meologit, meoprobit, mepoisson, and meprobit.

    The same estimation commands now support multilevel sampling and frequency weights, too.

15. **New random-number generators (RNGs)**

    Existing function runiform() now uses the 64-bit Mersenne Twister. runiform() produces uniformly distributed random numbers, and the functions providing random numbers for other distributions use runiform() in producing their results. Thus, all of Stata's RNGs are now based on the Mersenne Twister, too. Stata previously used KISS32 and still does under version control.

    KISS32 is an excellent RNG, but the Mersenne Twister has better properties and a longer period, namely $2^{19937} - 1$. The Mersenne Twister is 623-dimensionally equidistributed and has 53-bit resolution.

Uniformly distributed RNGs in specified intervals

Existing function runiform() now allows you to specify the range over which random variates will be supplied. runiform(*a*, *b*) returns values in the open interval (a, b).

New function runiformint(*a*,*b*) returns integer values in the closed interval $[a, b]$.

It is a minor technical detail, but existing function runiform() without arguments now produces random numbers in the open interval $(0, 1)$ instead of $[0, 1)$ as previously. It produces $[0, 1)$ values under version control when KISS32 is used.

New RNGs for distributions

Newly provided are

rexponential(*b*)
: exponential random variates with scale b

rlogistic()
: logistic variates with mean 0 and standard deviation $\pi/\sqrt{3}$

rlogistic(*s*)
: logistic variates with mean 0, scale s, and standard deviation $s\pi/\sqrt{3}$

rlogistic(*m*,*s*)
: logistic variates with mean m, scale s, and standard deviation $s\pi/\sqrt{3}$

rweibull(*a*,*b*)
: Weibull variates with shape a and scale b

rweibull(*a*,*b*,*g*)
: Weibull variates with shape a, scale b, and location g

rweibullph(*a*,*b*)
: Weibull (proportional hazards) variates with shape a and scale b

`rweibullph(`*a*`,`*b*`,`*g*`)`
Weibull (proportional hazards) variates with shape *a*, scale *b*, and location *g*

See [FN] **Random-number functions**.

Choosing which RNG to use

You are running Stata with version 14 set. You want values from `rlogistic()` but based on KISS32 rather than the version-14 default of Mersenne Twister. You could type

```
. version 13:  ... rlogistic() ...
```

or you can just use new function `rlogistic_kiss32()` without resetting the version:

```
. ... rlogistic_kiss32() ...
```

That is, every RNG *fcn*`()` comes in three flavors: *fcn*`()`, *fcn*`_mt64()`, and *fcn*`_kiss32()`.

Functions *fcn*`_mt64()` and *fcn*`_kiss32()` are now considered the true names of the RNGs, but still, you will usually type *fcn*`()`.

That is because of another new feature:

```
. set rng kiss32
```

`set rng kiss32` says that when you type *fcn*`()`, you mean *fcn*`_kiss32()`. You can `set rng` to `kiss32`, `mt64`, or `default`. That is how the meaning of *fcn*`()` is set. `default` means the default for the version. In version 14, the default is `mt64`. In version 13 and before, it is `kiss32`.

Programmers: Ado-file code written under previous versions of Stata now use modern RNGs! You do not have to modify your ado-files. That is because how `version` is set for the RNGs has been modified. Users typing `version` at the command line or in a do-file set RNG's version, too. Ado-files setting version, however, do not change RNG's version! In ado-files, the RNG's version can be set by setting the user version if you wanted to set it, but you do not. See [P] **version**.

Setting seeds and states

You previously could set the seed or reset the state of the RNGs by using `set seed`. Now, `set seed` is used solely for setting the seed. New command `set rngstate` is used for resetting the state. See [R] **set seed**.

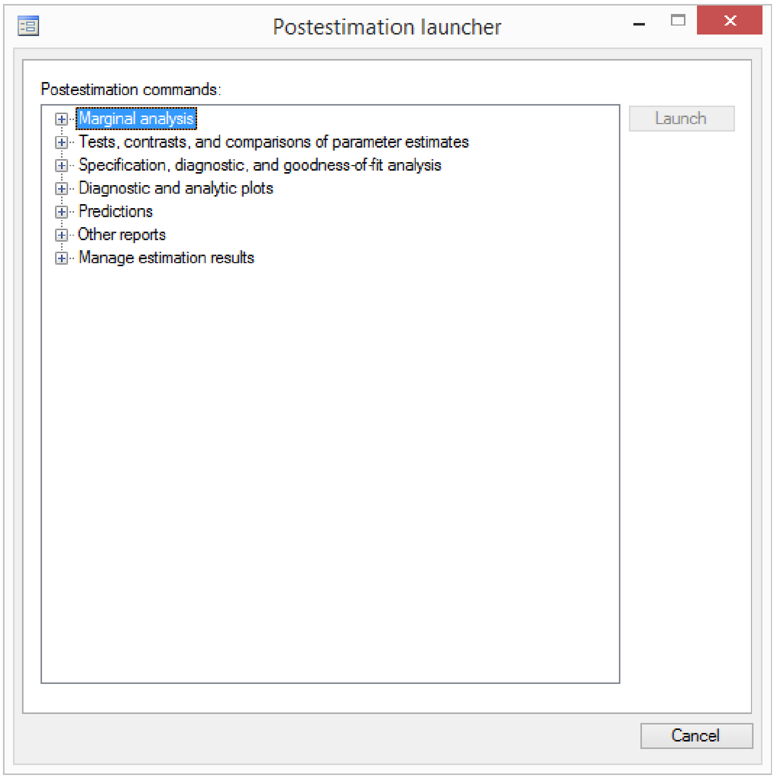
You previously obtained the state of the RNGs by using `c(seed)`. Stata continues to understand that, but officially, you are supposed to use `c(rngstate)`.

See [FN] **Random-number functions** and [R] **set seed**.

16. **Postestimation made easy**

You have to try this. Clear your Stata's estimation results, if any; type `discard`. Now type `postest`. A little, empty window will pop up. Move it to where it does not overlap Stata but you can see it. Now run an estimation command—any estimation command. You could type `use auto` and then type `regress mpg weight foreign`.

Isn't that neat? Well, if you are not following along, let us tell you what just appeared in that little, empty window. This appeared:


Postestimation launcher
Postestimation commands:
Marginal analysis
Tests, contrasts, and comparisons of parameter estimates
Specification, diagnostic, and goodness-of-fit analysis
Diagnostic and analytic plots
Predictions
Other reports
Manage estimation results
Launch
Cancel

Those are the postestimation things you can do after `regress`. Use a different estimation command and you will see a different list. Click on a topic and it expands. That list is tailored, too. Highlight a detailed element, click on Launch, and you are in the dialog box for the postestimation feature and it is filled in as much as it can be.

Enjoy.

17. **New and improved features in margins**

Existing command `margins` is used after estimation. `margins` uses the fitted results, the data in memory, and a little bit that you type to produce estimates of marginal effects, marginal means, predictive margins, population-averaged effects, and least-squares means, and presents the estimates in tables or graphs.

With `margins`, you can do what-if analyses. What would have been observed if everyone in the data were males? Females? What would have happened if the men in the data had their same characteristics but were relabeled women, and the women had their same characteristics but were relabeled men? If you can think of a counterfactual, potential outcome, comparison, or contrast, `margins` can do it.

We have improved `margins` in Stata 14.

Works with multiple outcomes simultaneously

margins previously restricted you to working with one outcome at a time. No longer does it do this, and by default, margins automatically produces its results for all equations, outcomes, or ordered levels of the fitted model. If you fit a multivariate regression on two variables, you get margins results for both variables. If you fit an ordinal model, you get results for each of the ordered levels.

Use the predict() option to restrict results to selected equations, outcomes, or levels if you wish. You may now specify multiple predict() options on the same margins command.

Integrates over unobserved components after multilevel and SEM models

Making predictions (even if counterfactual) is difficult with models that contain random or latent variables which are, after all, unobserved. margins now integrates over them and gets the average. Integrating over unobserved components is the logical counterpart of producing population-averaged results by averaging over your data.

There is even logic for it when making predictions about individuals; you are making average predictions for individuals with the same characteristics. What is the expected probability of high blood pressure for males, age 50, weight 190, based on a random-effects logistic regression? To compute that probability, you cannot simply take the random effect at its known mean of 0. You must integrate over the distribution of the random effect. margins does this.

predict has these features, too

margins has the new features just described, but in fact, it inherited them from new features of predict. You can now use predict to obtain predictions integrated over the distributions of unobserved components, random effects, and latent variables.

These marginal predictions are produced if you specify new option marginal with predict and are available after gsem, mecloglog, meglm, melogit, menbreg, meologit, meoprobit, mepoisson, and meprobit.

Better default statistics after some estimators

margins now uses its own default prediction statistic rather than the default prediction for the estimator. Sometimes, the default for the estimator is not statistically appropriate for use with margins, or there is an optional prediction statistic that is more interesting for marginal analysis. In such cases, margins now uses the most interesting or appropriate prediction statistic by default. You can still choose any available statistic by using the predict() option. The previous default is preserved under version control. See item 36 below for the complete list of new defaults.

margins is faster

margins is now much faster when computing predictive margins and marginal effects on predicted probabilities after ologit, oprobit, and mlogit.

It is also much faster when all of a model's *indepvars* are fixed to constants, such as when the atmeans option is specified.

margins can now add its results to your data

margins has a new generate() option. You supply the stub of a name, and margins fills in the rest. Specify the option, and margins creates new variables containing observation-by-observation values used to produce each of its reported results.

This is an often-requested feature and is useful when you use margins to produce a single or a small set of results. Otherwise, you will be inundated with new variables that are difficult to interpret.

This new, useful feature is currently "undocumented" in Stata speak, meaning that the documentation for it is available solely in `help margins generate`.

And it is now easier to determine which features are available

Each estimation command now documents which of the available predicted statistics is the default statistic for `margins` and which of the other statistics are appropriate for use with `margins`.

See [R] **margins**.

18. **Hurdle model estimation**

    New estimation command `churdle` fits linear or exponential hurdle models. Hurdle models allow us to model censored and uncensored outcomes separately. Uncensored outcomes are assumed to be observed when a hurdle is cleared. Censored outcomes are a result of not clearing the hurdle.

    Hurdle models come in two- and three-equation forms. The two-equation form handles a right- or left-censoring. The three-equation form handles combined censoring.

    Consider modeling how much people spend at the movies. We have data on a cross-section of people and the amount they spent last month. In our data, many people spent nothing because they did not even go to the movies; and how much the rest spent varies. In the hurdle model, we assume that once the decision is made to go to the movies, the amount spent can be treated independently of the decision to go. Those are the two equations: one for the decision to go and another for the amount spent.

    See [R] **churdle**.

19. **Censored Poisson estimation**

    New command `cpoisson` fits censored Poisson regressions to count outcomes. These are Poisson models with values that are not observed if they are below a threshold, or they are above a threshold, or both. The thresholds can be fixed values, such as 5 and 10, or can be recorded in variables, meaning that thresholds vary observation by observation.

    Postestimation, you can obtain predictions of the number of uncensored events, the uncensored incidence rate, the uncensored probability of a particular value or range of values, and the expected conditional probabilities of a value or range conditional on being within the censoring limits.

    See [R] **cpoisson**.

20. **Support for ICD-10 medical diagnosis codes**

    New command `icd10` joins existing commands `icd9` and `icd9p`.

    New command `icd10` provides automatic mapping of the World Health Organization's ICD-10 diagnosis codes for mortality and morbidity, and it adds some features not previously provided with `icd9` and `icd9p`. It allows a version control that sets descriptions to those that were current at the time your data were recorded, it works with a subset or with all records of your data, and it can be used with category or subcategory codes.

    `icd10` allows you to do the same things you could do with `icd9` and `icd9p`; namely, you can standardize the format of codes in your data, confirm that codes are defined, verify that codes are formatted correctly, and easily create indicators for the presence of different conditions.

Meanwhile, existing commands `icd9` and `icd9p` (for ICD-9, of course) provide the same new features, where applicable, provided by `icd10`. These new features include `if` *exp* and `in` *range* being allowed, and more.

See [D] **icd10** and [D] **icd9**.

We would like to thank the World Health Organization for making these codes available to Stata users. See `copyright icd10` for allowed usage.

21. **Excel reports get better**

    One of the more popular features of Stata 13 was the `putexcel` command for exporting results to Excel. All it did, however, was allow you to poke numbers and strings into Excel worksheets. Even so, a Stata blog entry about it was our most popular of the release!

    So we have added to `putexcel`. In addition to previous features, you can now

    - insert Stata graphs
    - insert text and format it with alignment, boldface, italics, color, and more
    - specify Excel formats, including date formats, currency formats, and more
    - make better tables with cell spanning, border formatting, and more
    - insert Excel formulas

    All the things you can do from `putexcel` you can also do from Mata's `xl()` Excel file I/O class.

    See [P] **putexcel** and [M-5] **xl()**.

22. **Manual entries now have Quick starts**

    Have you ever looked at a Stata command for the first time and wanted to see some examples without explanation that do something interesting? Or have you ever needed a quick refresher on a command's most common syntaxes?

    If so, what you want is now at the top of the command's manual entry. We call them Quick starts. We show a few examples for simple commands and sometimes more than a few when the command's syntax is more complex. If a command involves several steps, those steps are shown, too.

    The Quick starts are located right below the Description, and the Description now appears below the Title, where it always should have been.

    Quick starts do not appear in the help files, but they are just a click away. Click on the blue command name in the Title. In the help files, we keep Syntax near the top so that experts who just need the facts can see a quick refresher.

23. **Stata in Spanish and Japanese**

    It is called localization when a software's menus, dialogs, and the like are translated into other languages. We have completed localization of Stata for Spanish and Japanese. Manuals and help files remain in English.

    If your computer is set to a specific language, and that language is Spanish or Japanese, Stata will recognize this and automatically use that language. To manually change the language, select **Edit** > **Preferences** > **User-interface language...** (Windows and Unix), or select **Stata 14** > **Preferences** > **User-interface language...** (Mac). Alternatively, you can change the language from the command line: see [P] **set locale_ui**.

    StataCorp translated to Spanish, and StataCorp gratefully acknowledges the efforts of LightStone Corporation, Stata's official distributor in Japan, for translating to Japanese.

1.3.2 What's new in statistics (general)

Already mentioned as highlights of the release were the following:

Bayesian statistical analysis
Regression models for fractional data
Hurdle model estimation
Censored Poisson estimation
New random-number generators (RNGs)
New and improved features in margins

The following are also new:

24. New commands `ztest` and `ztesti` compare means in one or two samples using a z test and assuming known variances. With two samples, `ztest` supports both paired and unpaired data. `ztesti` is the immediate form of `ztest` that allows you to perform a test by typing summary statistics rather than using a dataset. See [R] **ztest**.

25. Almost every estimator now supports factor variables. New to this list in Stata 14 are `asmprobit`, `asroprobit`, `asclogit`, `nlogit`, `gmm`, and `mlexp`.

26. Existing estimation command `nlogittree` now reports when any observations violate the specified nesting structure of the model and will result in `nlogit` dropping observations or terminating with an error. See [R] **nlogit**.

27. Existing commands `test` and `testparm` provide new option `df(#)` to specify that the F distribution, rather than the default χ^2 distribution, be used when performing the Wald test.

28. When used with survey data, existing command `testparm` provides new option `nosvyadjust`, which specifies that the Wald test is to be performed without the default adjustment for the design degrees of freedom.

29. Existing command `cumul` now labels the generated variable. The label is "ECDF of *varname*".

30. Existing command `ksmirnov` no longer reports the corrected p-value. The "exact" p-values for the one-sample and two-sample tests are based on the asymptotic limiting distribution of the test statistic and involve infinite series. By default, Stata reports an approximate p-value computed using the first five terms of the series. The corrected p-value was obtained by applying an ad hoc correction to the approximate p-value to make it closer to the exact p-value. In recent simulation studies, we found that this correction does not perform satisfactorily in all situations and is thus no longer supported. The old results can be obtained under version control. For a two-sample test, you can use the `exact` option to obtain the exact p-value. See [R] **ksmirnov**.

31. Existing command `tabulate` now has new options `rowsort` and `colsort`, which specify that the rows (columns) of a two-way tabulation be presented in order of observed frequency. See [R] **tabulate twoway**.

32. Existing estimation commands `mprobit` and `mlogit` with constraints defined using equation indices rather than the equation names would apply the constraints without accounting for the base outcome equation. For example,

```
. sysuse auto
. constraint 1 [#2]turn = [#2]trunk
. mprobit rep78 turn trunk, baseoutcome(1) constraint(1)
```

would result in `[3]turn = [3]trunk` instead of `[2]turn = [2]trunk`. Now `mprobit` accounts for the base outcome equation when applying such constraints. The old behavior is preserved under version control. Either behavior is truly reasonable, but the new behavior is more consistent with how these commands report their results.

33. Existing estimation commands `mprobit` and `mlogit` did not respect constraints that had been defined using value labels associated with the levels of the outcome variable. For example,

    ```
    . label define replab 1 "A" 2 "B" 3 "C" 4 "D" 5 "E"
    . label values rep78 replab
    . constraint 1 [2]turn = [2]trunk
    . mprobit rep78 turn trunk, baseoutcome(1) constraint(1)
    ```

    would drop constraint 1. Now `mprobit` and `mlogit` work with constraints specified in this way.

    You can also now refer to the estimated coefficients using the equation name or the outcome value label in expressions. Using the above example, the following is now allowed:

    ```
    . test [3]turn = [3]trunk
    ```

    The old behavior disallowing these two behaviors is preserved under version control.

34. Existing command `nptrend` now displays value labels in its output. New option `nolabel` specifies that numerical codes be displayed rather than value labels. See [R] **nptrend**.

35. Existing postestimation commands `margins` and `marginsplot` have many new features. Some were mentioned in the highlights:

    Works with multiple outcomes simultaneously
    Integrates over unobserved components after multilevel and SEM models
    Better default statistics after some estimators
    margins is faster
    margins can now add its results to your data
    And it is now easier to determine which features are available

36. `margins` now defaults to prediction statistics different from the default for `predict` when `predict`'s default is not the most interesting statistic for marginal analysis or when that default is not statistically appropriate for marginal analysis.

    `margins` also defaults to producing statistics for all equations, outcomes, or levels when possible. The original behavior is preserved under version control. See the highlight *Better default statistics after some estimators*.

The following estimators have new defaults:

Command	New default statistic
clogit	probability assuming fixed effect is zero
gsem	expected values for each outcome
heckoprobit	marginal probabilities for each outcome
manova	linear predictions for each equation
meologit	probabilities for each outcome
meoprobit	probabilities for each outcome
meqrlogit	linear prediction
meqrpoisson	linear prediction
mgarch ccc	linear predictions for each equation
mgarch dcc	linear predictions for each equation
mgarch dvech	linear predictions for each equation
mgarch vcc	linear predictions for each equation
mlogit	probabilities for each outcome
mprobit	probabilities for each outcome
mvreg	linear predictions for each equation
ologit	probabilities for each outcome
oprobit	probabilities for each outcome
reg3	linear predictions for each equation
rologit	linear prediction
sem	linear predictions for each observed endogenous variable
slogit	probabilities for each outcome
sureg	linear predictions for each equation
varbasic	linear predictions for each equation
var	linear predictions for each equation
vec	linear predictions for each equation
xtlogit, fe	probability assuming fixed effect is zero

For multilevel model estimators and gsem, the default prediction statistics are not statistically appropriate with margins. These estimators now support marginal predictions, which are highly interpretable when used with margins. These marginal means and probabilities are now the default statistics produced by margins after gsem, mecloglog, meglm, melogit, menbreg, meologit, meoprobit, mepoisson, and meprobit.

37. margins, contrast() has new suboptions to support multiple predict() options. Also see the highlight *Works with multiple outcomes simultaneously*.

38. margins after mixed, when the model specification includes multilevel weights, now uses the product of the multilevel weights when computing the means and margins. The previous behavior of using only the observation-level weights is preserved under version control.

39. Support for multilevel weights has been added in Stata 14 to several estimation commands: gsem, mecloglog, meglm, melogit, menbreg, meologit, meoprobit, mepoisson, and meprobit. margins also supports multilevel weights for all of these commands.

    margins uses the product of the multilevel weights when computing means and margins.

40. marginsplot now supports the new multiple predict() options allowed on margins. It also automatically handles the new default of multiple results that margins produces with multivariate, multinomial, and ordinal estimators. You can customize how these plot, equation, and outcome dimensions are graphed using the new directives _predict, _equation, and _outcome in the

existing options xdimension(), plotdimension(), bydimension(), and graphdimension(). See [R] **marginsplot**.

1.3.3 What's new in statistics (SEM)

Already mentioned as highlights of the release were the following:

Survival models
Satorra–Bentler scaled χ^2 test
Support for survey data
Beta distribution
Multilevel weights
Prediction improvements

The following are also new:

41. Existing postestimation command predict after sem now has the scores option for predicting parameter-level scores. See [SEM] **predict after sem**.

42. Existing estimation command sem now reports information about each dependent variable in the header of the estimation table.

43. Existing estimation command sem's starting-values logic for startvalues(fixedonly) and for those built on it now considers all constraints on the fixed-effects parameters. This improves convergence for some models.

1.3.4 What's new in statistics (multilevel modeling)

Already mentioned as highlights of the release were the following:

Multilevel mixed-effects parametric survival models
Small-sample inference for linear multilevel mixed models
Survey support and multilevel weights for multilevel models

The following are also new:

44. Existing postestimation command predict supports new options after the following me estimators, nearly all multilevel: meglm, melogit, meprobit, mecloglog, meologit, meoprobit, mepoisson, and menbreg.

    predict supports new options density and distribution for computing the density and distribution function of the fitted model using the current values and the estimated parameters.

    predict supports new option scores for predicting parameter-level scores.

45. Existing me estimation commands now support iweights in the fixed-effects and random-effects equations.

1.3.5 What's new in statistics (treatment effects)

Already mentioned as highlights of the release were the following:

Treatment effects for survival models
Endogenous treatments
Probability weights
Balance analysis

The following are also new:

46. Existing estimation command `etregress` has new features, an improvement, and a change.

    New option `cfunction` specifies that the model be estimated using control function rather than the previously available GMM and maximum likelihood estimators. See [TE] **etregress**.

    New option `poutcomes` specifies that a potential-outcome model be fit with separate standard deviation and correlation parameters in the treatment and control regimes.

    `etregress` is now faster.

    The labeling of the coefficients has changed such that that the treatment is represented in factor variable notation. The old labeling is maintained under version control.

    See [TE] **etregress**.

1.3.6 What's new in statistics (longitudinal/panel data)

Already mentioned as a highlight of the release was the following:

Panel-data survival models

The following are also new:

47. Three estimators add the `vce(robust)` and `vce(cluster ...)` options to compute standard errors that are robust to distributional assumptions and correlated data. This new support is provided for `xthtaylor`, `xtivreg`, and the random-effects estimator of `xtpoisson`. (`xtpoisson` previously supported the options for Gaussian-distributed random effects but not for the default gamma distribution.)

    See [XT] **xthtaylor**, [XT] **xtivreg**, and [XT] **xtpoisson**.

48. Existing estimation commands `xtologit` and `xtoprobit` now support weights—frequency weights (`fweights`), sampling weights (`pweights`), and importance weights (`iweights`). See [XT] **xtologit** and [XT] **xtoprobit**.

49. Existing estimation command `xtreg, fe` is now orders of magnitude faster when there are many panels, and there always are.

1.3.7 What's new in statistics (time series)

Everything new was mentioned in the following highlights:

Markov-switching regression models
Tests for structural breaks in time-series data

1.3.8 What's new in statistics (survival analysis)

Already mentioned as highlights of the release were the following:

Multilevel mixed-effects parametric survival models
Small-sample inference for linear multilevel mixed models
Survey support and multilevel weights for multilevel models

The following are also new:

50. Existing command `stcurve` has new option `marginal`, which is a synonym for option `unconditional`; see [ST] **streg postestimation**.
51. Existing estimation command `stcox` now allows factor variables in option `tvc()`; see [ST] **stcox**.
52. System variables created by command `stset`—`_st`, `_d`, `_t0`, `_t`, and `_origin`—are now labeled.
53. Variables generated by existing command `sttocc` are now labeled.
54. Existing estimation command `streg`'s option `distribution(gamma)` was renamed to `distribution(ggamma)`. `gamma` continues to be supported under version control.

    This distribution should always have been designated `ggamma` because it is one of the generalized gamma distributions. It is renamed now to avoid confusion with the `gamma` argument allowed with the new option `distribution(gamma)` allowed on the `mestreg` command and on the `xtstreg` command and with the new option `family(gamma)` allowed on the `gsem` command.

1.3.9 What's new in statistics (survey data)

Already mentioned as highlights of the release were the following:

Survey support and multilevel weights for multilevel models
Support for survey data (SEM)

The following are also new:

55. Existing command `svyset` has a new syntax for specifying stage-level sampling weight variables. New syntax supports commands such as `gsem` and `meglm` that can fit hierarchical multilevel models with group-level weights. See [SVY] **svyset**, and for examples of fitting a multilevel model with stage-level sampling weights, see examples 5 and 6 in [ME] **meglm**.
56. The existing prefix command `svy jackknife:` with `svyset` replicate weight variables now uses the specified multiplier values as `svyset` in the `jkrweight()` option, even for unit-valued multipliers. The old behavior where `svy jackknife` used the default delete-1 multiplier instead of unit-valued multipliers is preserved under version control.

1.3.10 What's new in statistics (power and sample size)

Already mentioned as a highlight of the release was the following:

Power analysis for survival and epidemiological methods

The following are also new:

57. Existing command `power` now displays an estimated target variance in addition to the effect size when computing effect size for the analysis of variance and covariance methods. See [PSS] **power oneway**, [PSS] **power twoway**, and [PSS] **power repeated**.

1.3.11 What's new in statistics (multiple imputation)

58. Existing command mi impute pmm now requires the specification of the number of nearest neighbors in the knn() option. Before, mi impute pmm used the default of one nearest neighbor, knn(1).

    Recent simulation studies demonstrated that using one nearest neighbor performed poorly in many of the considered scenarios. In general, the optimal number of nearest neighbors varies from one application to another. Thus mi impute pmm now requires that the knn() option be specified. See [MI] **mi impute pmm** for details.

    This change will also affect mi impute monotone and mi impute chained when the pmm method is used in the conditional specifications. The old behavior of the commands is available under version control.

59. mi now requires that the names of imputation and passive variables not exceed 29 characters. In the wide style, the names of these variables may be restricted to fewer than 29 characters depending on the number of imputations. In the flongsep style, the names of regular variables in addition to the names of imputation and passive variables also may not exceed 29 characters. These requirements are imposed by the internal structure of the mi command. The affected commands are

    mi convert mi import mi passive mi register mi rename mi set M

60. mi supports new estimation command fracreg. See [R] **fracreg**.

1.3.12 What's new in statistics (multivariate)

61. margins used after existing estimation commands manova and mvreg now defaults to reporting linear predictions for all equations. Previous behavior was to default to reporting linear predictions for only the first equation.

1.3.13 What's new in data management

Already mentioned as highlights of the release were the following:

Unicode support
More than 2 billion observations now allowed
Support for ICD-10 medical procedure codes
Excel reports get better

The following are also new:

62. Stata's .dta dataset file format has changed. The change was unavoidable because of Stata 14's new Unicode features and the increase in the allowed maximum number of observations. use continues to read old-format files, of course. save writes new-format files. Use saveold when sharing datasets with users of previous versions of Stata.

    Few will be interested, but those who are should see dta for the technical specification of the new dataset format.

63. Existing command saveold has a new version() option and the corresponding ability to save files not only in the format of the previous Stata release, but in the formats of older releases, too. saveold can save data in the formats of Stata 11, 12, and 13. See saveold in [D] **save**.

64. Existing commands icd9 and icd9p have new features. This was mentioned in passing in the highlights of the new icd10 command. See [D] **icd9**.

65. Existing command destring now handles Unicode. destring's option ignore() has new suboptions asbytes and illegal in support of Unicode, but you are unlikely ever to want to specify either of these options.

    New suboption asbytes specifies that multibyte sequences be treated as nothing more than a series of bytes, meaning strings may contain untranslated Extended ASCII characters. New suboption illegal specifies that multibyte characters that do not make sense to Unicode are to be ignored.

    See [D] **destring**.

66. Existing command import delimited has new option encoding("*encoding*"). This option allows import delimited to import into Stata files that contain Extended ASCII strings. The Extended ASCII characters are translated into Unicode. In addition, import delimited's performance has been improved. See [D] **import delimited**.

67. Existing command generate has new options before() and after() so that the newly created variable be placed before(*existing_var*) or after(*existing_var*). See [D] **generate**.

68. New command insobs inserts new, empty observations into the dataset. Empty means that the variables in the new observations contain missing. See [D] **insobs**.

1.3.14 What's new in functions

Already mentioned as highlights of the release were the following:

Uniformly distributed RNGs in specified intervals
New RNGs for distributions

It is also worth mentioning that Stata has a new *Functions Reference Manual* dedicated solely to the documentation of functions.

The following new functions are added to Stata and Mata:

69. New function strrpos(s_1,s_2) returns the position of the last occurrence of s_2 in s_1. That is to say, strrpos() searches backward from the right. See [FN] **String functions**.

70. New string functions u*fcn*() and ud*fcn*() corresponding to each string function *fcn*() are added. These functions work in the Unicode metric as described in highlights.

The following u*fcn*()s are added:

New function	Corresponding to
ustrlen()	strlen()
usubstr()	substr()
usubinstr()	subinstr()
ustrpos()	strpos()
ustrrpos()	strrpos()
ustrlower()	strlower()
ustrupper()	strupper()
ustrtitle()	strproper()
ustrltrim()	strltrim()
ustrrtrim()	strrtrim()
ustrtrim()	strtrim()
ustrregexm()	regexm()
ustrregexra()	regexr()
ustrregexrf()	regexr() [*sic*]
ustrregexs()	regexs()
ustrreverse()	strreverse()
uchar()	char()
ustrtoname()	strtoname()
ustrword()	word()
ustrwordcount()	wordcount()

The following ud*fcn*()s are added:

New function	Corresponding to
udstrlen()	strlen()
udsubstr()	substr()

See [FN] **String functions**.

71. New Unicode string functions `ustrcompare()`, `ustrcompareex()`, `ustrsortkey()`, and `ustrsortkeyex()` compare and sort Unicode strings in a locale-aware manner, for instance, by generating a key for use with the `sort` command. See [U] **12.4.2.5 Sorting strings containing Unicode characters**.

72. New Unicode string functions `ustrfrom()` and `ustrto()` convert strings between UTF-8 and Extended ASCII encodings. When converting from Extended ASCII to UTF-8, that is, when using `ustrfrom()`, you may want to first use new function `ustrinvalidcnt()`, which counts the number of character sequences not already UTF-8. `ustrinvalidcnt()` indicates that a string needs conversion.

73. New Unicode string functions `ustrunescape()` and `ustrtohex()` translate escape sequences to and from Unicode. Escape sequences look like `\u00e8`, which is the escape sequence for “è”. Some websites write this as U+00E8. If you wanted “è” and did not know how to type it, you could type `ustrunescape("\u00e8")`. The function would return `"è"`.

    If you wanted to know the escape sequence for “è”, you could use function `ustrtohex("è")` and get back the string `"\u00e8"`.

74. New Unicode string function `tobytes()` returns the byte values of a Unicode string. For instance, `ustrtohex("è")` returns `"\d195\d168"` because the UTF-8 encoded form of “è” is two bytes long. The byte value is 195 followed by 168, written in decimal form.

75. New Unicode string functions `ustrleft()` and `ustrright()` are most easily understood in terms of `usubstr()`. `ustrleft(`s`, #)` is equal to `substr(`s`, 1, #)`. and `ustrright(`s`, #)` is equal to `substr(`s`, -#, .)`.

76. The following new Unicode string functions are rarely used and highly technical:

    `uisdigit()` `uisletter()` `ustrfix()` `ustrnormalize()` `wordbreaklocale()` `collator-locale()` `collatorversion()`

    See [FN] **String functions**.

77. There are many new random-number functions. The new functions are

    `runiform()` `runiformint()` `rexponential()` `rlogistic()` `rweibull()` `rweibullph()`

    See [FN] **Random-number functions**.

78. A new family of functions computes probabilities and other quantities of the logistic distribution.

    `logistic(`x`)` computes the cumulative distribution function of the logistic distribution with mean 0 and standard deviation $\pi/\sqrt{3}$.

    `logistic(`s`,`x`)` computes the cumulative distribution function of a logistic distribution with mean 0, scale s, and standard deviation $s\pi/\sqrt{3}$.

    `logistic(`m`,`s`,`x`)` computes the cumulative distribution function of a logistic distribution with mean m, scale s, and standard deviation $s\pi/\sqrt{3}$.

    `logisticden(`x`)` computes the density of the logistic distribution with mean 0 and standard deviation $\pi/\sqrt{3}$.

    `logisticden(`s`,`x`)` computes the density of the logistic distribution with mean 0, scale s, and standard deviation $s\pi/\sqrt{3}$.

    `logisticden(`m`,`s`,`x`)` computes the density of the logistic distribution with mean m, scale s, and standard deviation $s\pi/\sqrt{3}$.

    `logistictail(`x`)` computes the reverse cumulative logistic distribution with mean 0 and standard deviation $\pi/\sqrt{3}$.

    `logistictail(`s`,`x`)` computes the reverse cumulative logistic distribution with mean 0, scale s, and standard deviation $s\pi/\sqrt{3}$.

    `logistictail(`m`,`s`,`x`)` computes the reverse cumulative logistic distribution with mean m, scale s, and standard deviation $\pi/\sqrt{3}$.

    `invlogistic(`p`)` computes the inverse cumulative logistic distribution: if `logistic(`x`)` $= p$, then `invlogistic(`p`)` $= x$.

    `invlogistic(`s`,`p`)` computes the inverse cumulative logistic distribution: if `logistic(`s`,`x`)` $= p$, then `invlogistic(`s`,`p`)` $= x$.

    `invlogistic(`m`,`s`,`p`)` computes the inverse cumulative logistic distribution: if `logistic(`m`,`s`,`x`)` $= p$, then `invlogistic(`m`,`s`,`p`)` $= x$.

    `invlogistictail(`p`)` computes the inverse reverse cumulative logistic distribution: if `logistictail(`x`)` $= p$, then `invlogistictail(`p`)` $= x$.

invlogistictail(s,p) computes the inverse reverse cumulative logistic distribution: if logistictail(s,x) $= p$, then invlogistictail(s,p) $= x$.

invlogistictail(m,s,p) computes the inverse reverse cumulative logistic distribution: if logistictail(m,s,x) $= p$, then invlogistictail(m,s,p) $= x$.

rlogistic() computes logistic variates with mean 0 and standard deviation $\pi/\sqrt{3}$.

rlogistic(s) computes logistic variates with mean 0, scale s, and standard deviation $s\pi/\sqrt{3}$.

rlogistic(m,s) computes logistic variates with mean m, scale s, and standard deviation $s\pi/\sqrt{3}$.

See [FN] **Statistical functions** and [FN] **Random-number functions**.

79. A new family of functions computes probabilities and other quantities of the Weibull distribution.

weibull(a,b,x) computes the cumulative distribution function of a Weibull distribution with shape a and scale b. weibull(a,b,x) = weibull(a, b, 0, x).

weibull(a,b,g,x) computes the cumulative distribution function of a Weibull distribution with shape a, scale b, and location g.

weibullden(a,b,x) computes the density of the Weibull distribution with shape a and scale b. weibullden(a,b,x) = weibullden(a, b, 0, x).

weibullden(a,b,g,x) computes the density of the Weibull distribution with shape a, scale b, and location g.

weibulltail(a,b,x) computes the reverse cumulative Weibull distribution with shape a and scale b. weibulltail(a,b,x) = weibulltail(a, b, 0, x).

weibulltail(a,b,g,x) computes the reverse cumulative Weibull distribution with shape a, scale b, and location g.

invweibull(a,b,p) computes the inverse cumulative Weibull distribution: if weibull(a,b,x) $= p$, then invweibull(a, b, p) $= x$.

invweibull(a,b,g,p) computes the inverse cumulative Weibull distribution: if weibull(a,b,g, x) $= p$, then invweibull(a, b, g, p) $= x$.

invweibulltail(a,b,p) computes the inverse reverse cumulative Weibull distribution: if weibulltail(a,b,x) $= p$, then invweibulltail(a, b, p) $= x$.

invweibulltail(a,b,g,p) computes the inverse reverse cumulative Weibull distribution: if weibulltail(a,b,g,x) $= p$, then invweibulltail(a, b, g, p) $= x$.

rweibull(a,b) computes Weibull variates with shape a and scale b.

rweibull(a,b,g) computes Weibull variates with shape a, scale b, and location g.

See [FN] **Statistical functions** and [FN] **Random-number functions**.

80. A new family of functions computes probabilities and other quantities of the Weibull distribution (proportional hazards).

weibullph(a,b,x) computes the cumulative distribution function of a Weibull distribution (proportional hazards) with shape a and scale b. weibullph(a,b,x) = weibullph(a, b, 0, x).

weibullph(a,b,g,x) computes the cumulative distribution function of a Weibull distribution (proportional hazards) with shape a, scale b, and location g.

weibullphden(a,b,x) computes the density of the Weibull distribution (proportional hazards) with shape a and scale b. weibullphden(a,b,x) = weibullphden(a, b, 0, x).

`weibullphden(`a`,`b`,`g`,`x`)` computes the density of the Weibull distribution (proportional hazards) with shape a, scale b, and location g.

`weibullphtail(`a`,`b`,`x`)` computes the reverse cumulative Weibull distribution (proportional hazards) with shape a and scale b. `weibullphtail(`a`,`b`,`x`)` $=$ `weibullphtail(`a`,`b`,0,`x`)`.

`weibullphtail(`a`,`b`,`g`,`x`)` computes the reverse cumulative Weibull distribution (proportional hazards) with shape a, scale b, and location g.

`invweibullph(`a`,`b`,`p`)` computes the inverse cumulative Weibull distribution (proportional hazards): if `weibullph(`a`,`b`,`x`)` $= p$, then `invweibullph(`a`,`b`,`p`)` $= x$.

`invweibullph(`a`,`b`,`g`,`p`)` computes the inverse cumulative Weibull distribution (proportional hazards): if `weibullph(`a`,`b`,`g`,`x`)` $= p$, then `invweibullph(`a`,`b`,`g`,`p`)` $= x$.

`invweibullphtail(`a`,`b`,`p`)` computes the inverse reverse cumulative Weibull distribution (proportional hazards): if `weibullphtail(`a`,`b`,`x`)` $= p$, then `invweibullphtail(`a`,`b`,`p`)` $= x$.

`invweibullphtail(`a`,`b`,`g`,`p`)` computes the inverse reverse cumulative Weibull distribution (proportional hazards): if `weibullphtail(`a`,`b`,`g`,`x`)` $= p$, then `invweibullphtail(`a`,`b`,`g`,`p`)` $= x$.

`rweibullph(`a`,`b`)` computes Weibull (proportional hazards) variates with shape a and scale b.

`rweibullph(`a`,`b`,`g`)` computes Weibull (proportional hazards) variates with shape a, scale b, and location g.

See [FN] **Statistical functions** and [FN] **Random-number functions**.

81. A new family of functions computes probabilities and other quantities of the exponential distribution.

`exponential(`b`,`x`)` computes the cumulative distribution function of an exponential distribution with scale b.

`exponentialden(`b`,`x`)` computes the density of the exponential distribution with scale b.

`exponentialtail(`b`,`x`)` computes the reverse cumulative exponential distribution with scale b.

`invexponential(`b`,`p`)` computes the inverse cumulative exponential distribution: if `exponential(`b`,`x`)` $= p$, then `invexponential(`b`,`p`)` $= x$.

`invexponentialtail(`b`,`p`)` computes the inverse reverse cumulative exponential distribution: if `exponentialtail(`b`,`x`)` $= p$, then `invexponentialtail(`b`,`p`)` $= x$.

`rexponential(`b`)` computes exponential variates with scale b.

See [FN] **Statistical functions** and [FN] **Random-number functions**.

82. The following statistical functions are added:

`invnt(`df`,`np`,`p`)` computes inverse cumulative noncentral Student's t distribution.

`invnF(`df_1`,`df_2`,`np`,`p`)` computes inverse cumulative noncentral F distribution.

`lnwishartden(`df`,`V`,`X`)` computes natural logarithm of the density of the Wishart distribution.

`lniwishartden(`df`,`V`,`X`)` computes natural logarithm of the density of the inverse Wishart distribution.

`lnmvnormalden(`M`,`V`,`X`)` computes natural logarithm of the multivariate normal density.

`lnigammaden(`a`,`b`,`x`)` computes natural logarithm of the inverse gamma density.

See [FN] **Statistical functions**.

83. The string functions, random-number functions and statistical functions added to Stata were also added to Mata.

    See [M-4] **string**, [M-5] **runiform( )**, and [M-5] **normal( )**.

1.3.15 What's new in graphics

84. New commands `graph replay` and `graph close` join improved existing command `graph drop` to form a useful suite. All accept a graph name, a list of graph names, `_all`, and graph names with wildcards.

    `graph replay` redisplays graphs.

    `graph close` closes graph windows.

    `graph drop` drops graphs (and closes their window if they have one open).

    See [G-2] **graph replay**, [G-2] **graph close**, and [G-2] **graph drop**.

85. Existing command `histogram` now allows bin size to be recalculated in each category when `by` is specified; specify the new option `binrescale`. See [R] **histogram**.

86. Existing command `twoway kdensity` can now estimate the density one bandwidth beyond the maximum and minimum values of the dependent variable; specify new option `boundary`.

87. New graph commands for the new IRT models are provided:

    `irtgraph icc` graphs item characteristic curves.

    `irtgraph tcc` graphs test characteristic curves.

    `irtgraph iif` graphs the item information function.

    `irtgraph tif` graphs the test information function.

    See [IRT] **irtgraph icc**, [IRT] **irtgraph tcc**, [IRT] **irtgraph iif**, and [IRT] **irtgraph tif**.

88. `bayesgraph` graphs summaries and convergence diagnostics for simulated posterior distributions (MCMC samples) of model parameters and functions of model parameters obtained from new command `bayesmh`. Graphical summaries include trace plots, autocorrelation plots, and various distributional plots. See [BAYES] **bayesgraph**.

1.3.16 What's new in Mata

89. All the new functions added to Stata have also been added to Mata. See *What's new in functions* above.

90. Mata's file commands can now read, write, and seek with files that are longer than 2GB on 64-bit systems. See [M-5] **fopen( )**.

91. The default size of the Mata cache has been increased from 400 to 2,000 kilobytes. See [M-3] **mata set**.

92. Existing `xl()` Excel file I/O class has been extended beyond inserting text to include formatting of text, alignment, boldface, color, italics, and the like; inserting Stata graphs; specifying Excel formats including date formats, currency formats, etc.; cell spanning and table border formatting; and inserting Excel formulas. See the highlight *Excel reports get better* and see [M-5] **xl( )**.

93. New `PdfDocument()` class has creates PDF files from scratch in a programmatic way. See [M-5] **Pdf*( )**.

1.3.17 What's new in programming

94. Command `version` has new option `user`. This option causes `version` to backdate the random-number generators (RNGs). The new RNGs are a highlight of Stata 14. As we explained, the `version` command has become more sophisticated. Seemingly like magic, Stata chooses an RNG according to what the user has specified and ignores the version numbers specified in intermediary ado-files. If the user is running under version 14, it does not matter if the ado-file was written for Stata 12. Any `runiform()` function in it is given the same interpretation as in Stata 14.

    This is because Stata is tracking a second version number along with the first. The second is known as the user version and is set only when the `version` command is given interactively or in do-files. `version` given here sets both version numbers. In your ado-file, `version` sets only the first version number. If you want to set the second version number in your ado-file, you add a second `version` line with option `user`.

    Official guidelines are that you should not do this, or at least, not do this without warning the user that your command does not honor the implicit RNG setting via the user setting the version number.

95. `creturn()` reports two new system settings, `c(rng)` and `(rng_current)`.

    `c(rng)` corresponds to the setting of `set rng`, which will usually be the string `default` but could be `mt64` or `kiss32`. If it is `default`, the RNGs in effect on based on `version`, as described above.

    `c(rng_current)` reports the RNGs in effect—the RNGs that would be used by `runiform()` if it were given now—regardless of how that was set or determined. Its values are `mt64` or `kiss32`.

96. Stata's dialog programming language now provides a `TREEVIEW` input control. See *3.6.17 TREEVIEW tree input control* in [P] **dialog programming**.

97. New command `set locale_functions` *localename* resets the default locale used by new Unicode string functions, which take an optional locale argument. `set locale_functions` is automatically set to `default`, which means the operating system's recorded locale. See [P] **set locale_functions**.

98. Two new extended macro functions are provided.

    `:ustrlen` is the macro equivalent of the new `ustrlen()` function.

    `:udstrlen` is the macro equivalent of the new `udstrlen()` function.

    See [P] **macro**.

1.3.18 What's new in the Stata interface

Already mentioned as highlights of the release were the following:

Postestimation made easy
Manual entries now have Quick starts
Stata in Spanish and Japanese

The following are also new:

99. The Data Editor has three new features.

    You can now use Find in the Data Editor to search.

    You can now insert variables or observations in the middle of your data.

    You can print the entire dataset or a selection.

100. The Variable Manager now supports printing. You can print the entire list of variables or a selection.
101. You can now export graphs and results to PDF files in Stata for Unix. You could always do this with Windows or Mac.
102. The PDF export engine in Stata for Windows is all new and provides support for the new Unicode features in Stata.

1.3.19 What's more

We have not listed all the changes, but we have listed the important ones.

Stata is continually being updated. Those between-release updates are available for free over the Internet.

Type `update query` and follow the instructions.

We hope that you enjoy Stata 14.

1.4 References

Chambers, J. M., W. S. Cleveland, B. Kleiner, and P. A. Tukey. 1983. *Graphical Methods for Data Analysis*. Belmont, CA: Wadsworth.

Everitt, B. S., and A. Skrondal. 2010. *The Cambridge Dictionary of Statistics*. 4th ed. Cambridge: Cambridge University Press.

Heyde, C. C., and E. Seneta, ed. 2001. *Statisticians of the Centuries*. New York: Springer.

Johnson, N. L., and S. Kotz, ed. 1997. *Leading Personalities in Statistical Sciences: From the Seventeenth Century to the Present*. New York: Wiley.

Upton, G. J. G., and I. T. Cook. 2014. *A Dictionary of Statistics*. 3rd ed. Oxford: Oxford University Press.