In this manual, however, there is little dialog, 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 since 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 "`use myfile.`".

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 in the United Kingdom 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.

1.2.6 Vignettes

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 this release. A vignette could often appropriately go in several entries, so that G.E.P. Box deserves to be mentioned in entries other than [TS] **arima**, such as [R] **boxcox**. To save space, each is given once only, and an index of all vignettes is given in the *Quick Reference Manual and Index*.

All 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 (2006) and Everitt (2006) 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 previous Stata users. If you are new to Stata, you may as well skip to *What's more*, below.

As always, Stata is 100% compatible with the previous releases, but we remind programmers that it is important that you put `version 9`, or `version 8`, etc., at the top of your 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. Stata 10 has an interactive, point-and-click editor for your graphs. You do not need to type anything; you just right-click within the Graph window and select **Start Graph Editor**. You can do that any time, either when the graph is drawn or when you have `graph use`d it from disk. You can add text, lines, markers, titles, and annotations, outside the plot region or inside; you can move axes, titles, legends, etc.; you can change colors and sizes, number of tick marks, etc.; and you can even change scatters to lines or bars, or vice versa. See [G] **graph editor**.

2. You can now save estimation results to disk. After fitting a model, whether with `regress`, `logistic`, ..., or even a user-written command, you type `estimates save` *filename* to save it. You type `estimates use` *filename* to reload it later. See [R] **estimates**.

3. Stata now fits nested, hierarchical, and mixed models with binary and count responses; i.e., you can fit logistic and Poisson models with complex, nested error components. See [XT] **xtmelogit** and [XT] **xtmepoisson**.

4. Stata now has exact logistic and exact Poisson regression. In small samples, exact methods have better coverage than asymptotic methods, and exact methods are the only way to obtain point estimates, tests, and confidence intervals from covariates that perfectly predict the observed outcome. See [R] **exlogistic** and [R] **expoisson**.

5. Stata now supports LIML and GMM estimation in addition to 2SLS. Tests of instrumental relevance and tests of overidentifying restrictions are available. See [R] **ivregress** and [R] **ivregress postestimation**.

6. Stata now has more estimators for dynamic panel-data models, including the Arellano–Bover/Blundell–Bond system estimator. This estimator is an extension of the Arellano–Bond GMM estimator for dynamic panel models. It is more efficient and has smaller bias when the AR process is too persistent. These new estimators can also be used to fit models with serially correlated idiosyncratic errors and where the structure of the predetermined variables is complicated. These new estimators can compute the Windmeijer biased-corrected two-step robust VCE in addition to the standard one-step i.i.d., one-step robust, and two-step i.i.d. VCEs. See [XT] **xtabond**, [XT] **xtdpdsys**, and [XT] **xtdpd**.

7. New estimation command `nlsur` fits a system of nonlinear equations by feasible generalized least squares, allowing for covariances among the equations. See [R] **nlsur**.

8. Stata has new estimation commands for fitting categorical and ranked outcomes, often used for choice models. The new commands allow separate equations for each outcome and support the easy-to-use alternative-specific notation. Joining Stata's alternative-specific multinomial probit are alternative-specific conditional logit (McFadden's choice model) and alternative-specific rank-ordered probit (for modeling ordered outcomes). See [R] **asclogit** and [R] **asroprobit**.

9. Stata's `svy:` prefix now works with 48 estimators, 27 more than previously. Most importantly, `svy:` now works with Cox proportional hazards regression models (`stcox`) and parametric survival models (`streg`). See [SVY] **svy estimation**.

10. The new `stpower` command provides sample-size and power calculations for survival studies that use Cox proportional hazards regressions, log-rank tests for two groups, or differences in exponentially distributed hazards or log hazards. Available are (1) required sample size (given power), (2) power (given sample size), and (3) the minimal detectable effect (given power and sample size). `stpower` allows automated production of customizable tables and has options to assist with creating graphs and power curves. See [ST] **stpower**.

11. Stata 10 provides several discriminant analysis techniques, including linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), logistic discriminant analysis, and kth-nearest-neighbor (KNN) discrimination. See [MV] **discrim**.

12. Stata now provides multiple correspondence analysis (MCA) and joint correspondence analysis (JCA). See [MV] **mca**.

13. Stata now provides modern as well as classical multidimensional scaling (MDS), including metric and nonmetric MDS. Available loss functions include stress, normalized stress, squared stress, normalized squared stress, and Sammon. Available transformations include identity, power, and monotonic. See [MV] **mds**.

14. Stata 10 has time/date variables in addition to date variables, so now you can work with data that say an event happened on 12may2007 14:03:22.234 or events that happen every day at 14:03:22.234. Note the millisecond resolution. Time variables are available in two forms: adjusted for leap seconds and unadjusted. See [D] **dates and times**.

15. New Mata function `optimize()` performs minimization and maximization. You can code just the function, the function and its first derivatives, or the function and its first and second derivatives. Optimization techniques include Newton–Raphson, Davidon–Fletcher–Powell, Broyden–Fletcher–Goldfarb–Shanno, Berndt–Hall–Hall–Hausman, and the simplex method Nelder–Mead. See [M-5] **optimize( )**.
16. Stata's online help now provides saved results and examples that you can run.
17. Stata for Windows now supports Automation, formerly known as OLE Automation, which means that programmers can control Stata from other applications and retrieve results. See [P] **automation**.
18. Stata for Unix now supports unixODBC [*sic*], making it easier to connect to databases such as Oracle, MySQL, and PostgreSQL; see [D] **odbc**.

Another change is the introduction of Stata/MP, but that really happened during Stata 9. Stata/MP is the parallel version of Stata for multiple-CPU and multicore computers; see [U] **5 Flavors of Stata**. Stata/MP runs faster. In Stata 10, many more commands now exploit the multiple processors, which means that they run faster, too. This includes both survey and nonsurvey mean, total, ratio, and proportion estimators and the survey linearized variance estimator, which is available with many Stata estimation commands.

There is much more, and the important changes for you may not be what we list as highlights. Below are the details.

1.3.1 What's new in the GUI and command interface

1. The Review window has been redesigned. It now shows the return code of each previous command and highlights errors. From the window, you can now select multiple commands—not just single ones—to save or execute.
2. The Variables window has been redesigned. It nows shows the storage type and display format for each variable in addition to the variable's name and label. You can change any of these from the window, including the name; just right-click.
3. Stata's Viewer has been redesigned. In addition to an all-new look, it has a **Forward** button and the search capability is now built in rather than provided by a dialog box.
4. The Graph window has been redesigned. In addition to providing an interactive editor, under Windows, it now allows tabs. You can have either one window containing multiple graphs each on its own tab, or each graph in a separate window.
5. You can copy and paste from Stata's Results and Viewer windows in AS-IS mode, meaning that you can include Stata output in documents and slides looking exactly as it looked on your screen.
6. Multiple Do-file Editors can now be opened simultaneously; just click on the files in the Open dialog. (Unix users: you too can now open multiple Do-file Editors.)
7. Concerning dialogs,
    a. Stata now uses child dialogs, making dialogs for Stata commands easier to use.
    b. Programmers can program child dialogs, too; see `help dialog programming`.
    c. Graph dialogs have an all-new look that makes specifying the most important items and options easier, yet provides access to even more of `graph`'s capabilities.
    d. Dialogs that need matrices now allow the user to create the matrix via a matrix-input child dialog and to show that new matrix in the original dialog box after it has been created.
    e. Dialogs that need formats now allow the user to specify the format via a format builder.

f. Data from ODBC sources can now be accessed via a dialog.

g. Dialogs now scale with Microsoft Windows' DPI settings.

h. Dialogs now load faster.

8. Stata for Unix has an all new, more modern GUI.

9. Stata now executes `sysprofile.do`, if it exists, in addition to `profile.do` when Stata is launched. This allows system administrators to provide global customization. See [GS] **Appendix A: More on starting and exiting Stata**.

10. New command `adoupdate` automates the process of updating user-written ado-files; see [R] **adoupdate**.

11. New command `hsearch` searches the help files for specified words and presents a ranked, clickable list in the Viewer; see [R] **hsearch**.

12. Stata's help files are now named `*.sthlp` rather than `*.hlp`, meaning that user-written help files can be sent via email more easily. Many email filters flag `.hlp` files as potential virus carriers because Stata was not the only one to use the `.hlp` suffix. You need not rename your old help files. See [R] **help**.

13. There are now console versions of Stata/SE and Stata/MP for Macintosh, just as there are for Unix. They are included on the installation CD and installed automatically.

14. Stata's `in` command modifier now accepts `F` and `L` as synonyms for `f` and `l`, meaning first and last observations.

15. Multiple log files may be opened simultaneously; see [R] **log**.

16. Intercooled Stata has been renamed to Stata/IC.

1.3.2 What's new in data management

1. Stata 10 has new date/time variables, so you can now record values like 14jun2007 09:42:41.106 in one variable. They are called `%tc` and `%tC` variables. The first is unadjusted for leap seconds; the second is adjusted.

   What used to be called "daily variables" are now called `%td` variables. This is just a jargon change; daily (`%td`) variables continue to work as they did before—0 means 01jan1960, 1 means 02jan1960, and so on.

   `%tc` and `%tC` variables work similarly: 0 means 01jan1960 00:00:00. Here, however, 1 means 01jan1960 00:00:00.001, 1000 means 01jan1960 00:00:01.000, and 02jan1960 08:00:00 is 115,200,000. The underlying values are big—so it is important you store them as `doubles`—but the `%tc` and `%tC` formats make the values readable, just as the `%td` format makes daily (`%td`) values readable.

   There are many new functions to go along with this new value type. `clock()`, for instance, converts strings such as "02jan1960 08:00:00" (or even "8:00 a.m., 1/2/1960") to their numeric equivalents. `dofc()` converts a `%tc` value (such as 115,200,000, meaning 02jan1960 08:00:00) to its `%td` equivalent (namely, 1, meaning 02jan1960). `cofd()` does the reverse (the result would be 86,400,000, meaning 02jan1960 00:00:00).

   See [D] **dates and times**.

2. The previously existing `date()` function, which converts strings to `%td` values, is now smarter. In addition to being able to convert strings such as "21aug2005", "August 21, 2005", it can convert "082105", "08212005", "210805", and "21082005". See [D] **dates and times**.

3. New command `datasignature` allows you to sign datasets and later use that signature to determine whether the data have changed. An early version of the command was made available during the Stata 9 release. That command is now called `_datasignature` and was used as the building block for the new, improved `datasignature`. See [D] **datasignature** and [P] **_datasignature**.

4. Existing command `clear` now clears data and value labels only. Type `clear all` to clear everything. This change will bite you the first few times you type `clear` expecting it to `clear all`. The problem was that new users were surprised when `clear` by itself cleared everything, whereas `use` *filename*`, clear` loaded new data and value labels but left everything else in place. The new users were right.

   `clear` now has the following subcommands:

   a. `clear all` clears everything from memory.

   b. `clear ado` clears automatically loaded ado-file programs.

   d. `clear programs` clears all programs, automatically loaded or not.

   c. `clear results` clears saved results.

   d. `clear mata` clears Mata functions and objects from memory.

   See [D] **clear**.

5. Stata for Unix now supports unixODBC [*sic*], making it easier to connect to databases such as Oracle, MySQL, and PostgreSQL; see [D] **odbc**.

6. Existing command `describe` now allows option `varlist` that was previously allowed only by `describe using`. Existing command `describe using` *filename* now allows option `simple` that was previously allowed only by `describe`. Option `varlist` saves the variable names in `r(varlist)`, and option `simple` displays the variable names in a compact format. See [D] **describe**.

7. Existing command `collapse` now supports four additional *stat*s: `first`, the first value; `last`, the last value; `firstnm`, the first nonmissing value; and `lastnm`, the last nonmissing value. See [D] **collapse**.

8. Existing command `cf` (compare files) now provides a detailed listing of observations that differ when the `verbose` option is specified. Setting version to less than 10.0 restores the earlier behavior. See [D] **cf**.

9. Existing command `codebook` has new option `compact` that produces more compact output. See [D] **codebook**.

10. Existing command `insheet` has new option `case` that preserves the case of variable names when importing data; see [D] **insheet**.

11. Existing command `outsheet` has new option `delimiter()` that specifies an alternative delimiter; see [D] **outsheet**.

12. Existing commands `infile` and `infix` can now read up to 524,275 characters per line; the previous limit was 32,765. See [D] **infile** and [D] **infix (fixed format)**.

13. Existing commands `icd9` and `icd9p` have now been updated to use the V24 codes; see [D] **icd9**.

14. New function `itrim()` returns the string with consecutive, internal spaces collapsed to one space; see *String functions* in [D] **functions**.

15. New functions `lnnormal()` and `lnnormalden()` provide the natural logarithm of the cumulative standard normal distribution and of the standard normal density; see *Probability distributions and density functions* in [D] **functions**.

16. New functions for calculating cumulative densities are now available:

`binomial(`n`, `k`, `p`)`	lower tail of the binomial distribution
`ibetatail(`a`, `b`, `x`)`	reverse (upper tail) of the cumulative beta distribution
`gammaptail(`a`, `x`)`	reverse (upper tail) of the cumulative gamma distribution
`invgammaptail(`a`, `p`)`	inverse reverse of the cumulative gamma distribution
`invibetatail(`a`, `b`, `p`)`	inverse reverse of the cumulative beta distribution
`invbinomialtail(`n`, `k`, `p`)`	inverse of right cumulative binomial

    See *Probability distributions and density functions* in [D] **functions**.

17. Existing function `Binomial(`n`, `k`, `p`)` has been renamed `binomialtail(`n`, `k`, `p`)`, thus making its name consistent with the naming convention for probability functions. The accuracy of the function has also been improved for very large values of n. At the other end of the number line, the function now returns the appropriate 0 or 1 value when $n = 0$, rather than returning missing. `Binomial()` continues to work as a synonym for `binomialtail()`.

18. The behavior and accuracy of the following probability functions have been improved:

    a. `F(`n_1`, `n_2`, `f`)` and `Ftail(`n_1`, `n_2`, `f`)` are more accurate for small values of n_1 and large values of n_2. Also, `F()` is more accurate for large f where n_1 and n_2 are less than 1.

    b. `gammap(`a`, `x`)` is more accurate when a is large and x is near a.

    c. `ibeta(`a`, `b`, `x`)` now is more accurate when x is near $a/(a+b)$ and a or b is large.

    d. `invbinomial(`n`, `k`, `p`)`, `invchi2(`n`, `p`)`, `invchi2tail(`n`, `p`)`, `invF(`n_1`, `n_2`, `p`)`, and `invgammap(`a`, `p`)` are more accurate for small values of p or for returned values close to zero.

    e. `invFtail(`n_1`, `n_2`, `p`)` and `invibeta(`a`, `b`, `p`)` are more accurate for small values of p or for returned values close to zero.

    f. `invttail(`n`, `p`)` is more accurate for small values of p or for returned values close to zero.

    g. `ttail(`n`, `t`)` is more accurate for exceedingly large values of n.

19. Existing function `invbinomial(`n`, `k`, `p`)` now returns the probability of a success on one trial such that the probability of observing k or *fewer* successes in n trials is p. The previous behavior of `invbinomial()` is restored under version control.

20. New function `fmtwidth()` returns the display width of a *%fmt* string; see *Programming functions* in [D] **functions**.

21. The maximum length of a *%fmt* has increased from 12 to 48 characters; see [D] **format**. (This change was necessitated by the new date/time variables.)

22. Existing commands `corr2data` and `drawnorm` now allow singular correlation (or covariance) structures. New option `forcepsd` modifies a matrix to be positive semidefinite and thus to be a proper covariance matrix. See [D] **corr2data** and [D] **drawnorm**.

23. Existing command `hexdump, analyze` now saves the number of `\r\n` characters in `r(Windows)` rather than in `r(DOS)`. `r(DOS)` is still set when version is less than 10. See [D] **hexdump**.

(*Continued on next page*)

1.3.3 What's new in statistics (general)

1. As mentioned above, you can now save estimation results to disk. You type `estimates save` *filename* to save results and `estimates use` *filename* to reload them. In fact, the entire `estimates` command has been reworked. The new command `estimates notes` allows you to add notes to estimation results just as you add them to datasets. The new command `estimates esample` allows you to restore `e(sample)` after reloading estimates, should that be necessary (usually it is not). The maximum number of estimation results that can be held in memory (as opposed to saved on disk) is increased to 300 from 20. See [R] **estimates**.

2. Stata now has exact logistic and exact Poisson regression. Rather than having their inference based on asymptotic normality, exact estimators enumerate the conditional distribution of the sufficient statistics and then base inference upon that distribution. In small samples, exact methods have better coverage than asymptotic methods, and exact methods are the only way to obtain point estimates, tests, and confidence intervals from covariates that perfectly predict the observed outcome.

   Postestimation command `estat se` reports odds ratios and their asymptotic standard errors. `estat predict`, available only after `exlogistic`, computes predicted probabilities, asymptotic standard errors, and exact confidence intervals for single cases.

   See [R] **exlogistic** and [R] **expoisson**.

3. New estimation command `asclogit` performs alternative-specific conditional logistic regression, which includes McFadden's choice model. Postestimation command `estat alternatives` reports alternative-specific summary statistics. `estat mfx` reports marginal effects of regressors on probabilities of each alternative. See [R] **asclogit** and [R] **asclogit postestimation**.

4. New estimation command `asroprobit` performs alternative-specific rank-ordered probit regression. `asroprobit` is related to rank-ordered logistic regression (`rologit`) but allows modeling alternative-specific effects and modeling the covariance structure of the alternatives. Postestimation command `estat alternatives` provides summary statistics about the alternatives in the estimation sample. `estat covariance` displays the variance–covariance matrix of the alternatives. `estat correlation` displays the correlation matrix of the alternatives. `estat mfx` computes the marginal effects of regressors on the probability of the alternatives. See [R] **asroprobit** and [R] **asroprobit postestimation**.

5. New estimation command `ivregress` performs single-equation instrumental-variables regression by two-stage least squares, limited-information maximum likelihood, or generalized method of moments. Robust and HAC covariance matrices may be requested. Postestimation command `estat firststage` provides various descriptive statistics and tests of instrument relevance. `estat overid` tests overidentifying restrictions. `ivregress` replaces the previous `ivreg` command. See [R] **ivregress** and [R] **ivregress postestimation**.

6. New estimation command `nlsur` fits a system of nonlinear equations by feasible generalized least squares, allowing for covariances among the equations; see [R] **nlsur**.

7. Existing estimation command `nlogit` was rewritten and has new, better syntax and runs faster when there are more than two levels. Old syntax is available under version control. `nlogit` now fits the random utilities maximization (RUM) model by default as well as the nonnormalized model that was available previously. The new `nlogit` now allows unbalanced groups and allows groups to have different sets of alternatives. `nlogit` now excludes entire choice sets (cases) if any alternative (observation) has a missing value; use new option `altwise` to exclude just the alternatives (observations) with missing values. Finally, `vce(robust)` is allowed regardless of the number of nesting levels. See [R] **nlogit**.

8. Existing estimation command `asmprobit` has the following enhancements:

   a. The new default parameterization estimates the covariance of the alternatives differenced from the base alternative, making the estimates invariant to the choice of base. New option `structural` specifies that the previously structural (nondifferenced) covariance parameterization be used.

   b. `asmprobit` now permits estimation of the constant-only model.

   c. `asmprobit` now excludes entire choice sets (cases) if any alternative (observation) has a missing value; use new option `altwise` to exclude just the alternatives (observations) with missing values.

   d. New postestimation command `estat mfx` computes marginal effects after `asmprobit`.

   See [R] **asmprobit** and [R] **asmprobit postestimation**.

9. Existing estimation command `clogit` now accepts `pweight`s and may be used with the `svy:` prefix.

   Also, `clogit` used to be willing to produce cluster–robust VCEs when the groups were not nested within the clusters. Sometimes, this VCE was consistent, and other times it was not. You must now specify the new `nonest` option to obtain a cluster–robust VCE when the groups are not nested within panels.

   `predict` after `clogit` now accepts options that calculate the $\Delta\beta$ influence statistic, the $\Delta\chi^2$ lack-of-fit statistic, the Hosmer and Lemeshow leverage, the Pearson residuals, and the standardized Pearson residuals.

   See [R] **clogit** and [R] **clogit postestimation**.

10. Existing estimation command `cloglog` now accepts `pweight`s, may now be used with the `svy:` prefix, and has new option `eform` that requests that exponentiated coefficients be reported; see [R] **cloglog**.

11. Existing estimation command `cnreg` now accepts `pweight`s, may be used with the `svy:` prefix, and is now noticeably faster (up to five times faster) when used within loops, such as by `statsby`. See [R] **cnreg**.

12. Existing estimation commands `cnsreg` and `tobit` now accept `pweight`s, may be used with the `svy:` prefix, and are now noticeably faster (up to five times faster) when used within loops, such as by `statsby`. Also, `cnsreg` now has new advanced option `mse1` that sets the mean squared error to 1. See [R] **cnsreg** and [R] **tobit**.

13. Existing estimation command `regress` is now noticeably faster (up to five times faster) when used with loops, such as by `statsby`. Also,

    a. Postestimation command `estat hettest` has new option `iid` that specifies that an alternative version of the score test be performed that does not require the normality assumption. New option `fstat` specifies that an alternative F test be performed that also does not require the normality assumption.

    b. Existing postestimation command `estat vif` has new option `uncentered` that specifies that uncentered variance inflation factors be computed.

    See [R] **regress postestimation**.

14. Existing estimation commands `logit`, `mlogit`, `ologit`, `oprobit`, and `probit` are now noticeably faster (up to five times faster) when used within loops, such as by `statsby`.

15. For existing estimation command `probit`, `predict` now allows the `deviance` option; see [R] **probit postestimation**.

16. Existing estimation command `nl` has the following enhancements:

    a. Option `vce(`*vcetype*`)` is now allowed, with supported *vcetype*s that include types derived from asymptotic theory, that are robust to some kinds of misspecification, that allow for intragroup correlation, and that use bootstrap or jackknife methods. Also, three heteroskedastic- and autocorrelation-consistent variance estimators are available.

    b. `nl` no longer reports an overall model F test because the test that all parameters other than the constant are jointly zero may not be appropriate in arbitrary nonlinear models.

    c. The coefficient table now reports each parameter as its own equation, analogous to how `ml` reports single-parameter equations.

    d. `predict` after `nl` has new options that allow you to obtain the probability that the dependent variable lies within a given interval, the expected value of the dependent variable conditional on its being censored, and the expected value of the dependent variable conditional on its being truncated. These predictions assume that the error term is normally distributed.

    e. `mfx` can be used after `nl` to obtain marginal effects.

    f. `lrtest` can be used after `nl` to perform likelihood-ratio tests.

    See [R] **nl** and [R] **nl postestimation**.

17. Existing estimation command `mprobit` now allows `pweight`s, may now be used with the `svy:` prefix, and has new option `probitparam` that specifies that the probit variance parameterization, which fixes the variance of the differenced latent errors between the scale and the base alternatives to one, be used. See [R] **mprobit**.

18. Existing estimation command `rologit` now allows `vce(bootstrap)` and `vce(jackknife)`. See [R] **rologit**.

19. Existing estimation command `truncreg` now allows `pweight`s and now works with the `svy:` prefix. See [SVY] **svy estimation**.

20. After existing estimation command `ivprobit`, postestimation commands `estat classification`, `lroc`, and `lsens` are now available. Also, in `ivprobit`, the order of the ancillary parameters in the output has been changed to reflect the order in `e(b)`. See [R] **ivprobit** and [R] **ivprobit postestimation**.

21. All estimation commands that allowed options `robust` and `cluster()` now allow option `vce(`*vcetype*`)`. `vce()` specifies how the variance–covariance matrix of the estimators (and hence standard errors) are to be calculated. This syntax was introduced in Stata 9, with options such as `vce(bootstrap)`, `vce(jackknife)`, and `vce(oim)`.

    In Stata 10, option `vce()` is extended to encompass the robust (and optionally clustered) variance calculation. Where you previously typed

    . *estimation-command* ..., `robust`

    you are now to type

    . *estimation-command* ..., `vce(robust)`

    and where you previously typed

    . *estimation-command* ..., `robust cluster(`*clustervar*`)`

    with or without the `robust`, you are now to type

    . *estimation-command* ..., `vce(cluster` *clustervar*`)`

You can still type the old syntax, but it is undocumented. The new syntax emphasizes that the robust and cluster calculation affects standard errors, not coefficients. See [R] ***vce_option***.

Going along with this change, estimation commands now have a term for their default variance calculation. Thus, you will see things like `vce(ols)`, and `vce(gnr)`. Here is what they all mean:

a. `vce(ols)`. The variance estimator for ordinary least squares; an $s^2(X'X)^{-1}$-type calculation.

b. `vce(oim)`. The observed information matrix based on the likelihood function; a $(-H)^{-1}$-type calculation, where H is the Hessian matrix.

c. `vce(conventional)`. A generic term to identify the conventional variance estimator associated with the model. For instance, in the Heckman two-step estimator, `vce(conventional)` means the Heckman-derived variance matrix from an augmented regression. In two different contexts, `vce(conventional)` does not necessarily mean the same calculation.

d. `vce(analytic)`. The variance estimator derived from first principles of statistics for means, proportions, and totals.

e. `vce(gnr)`. The variance matrix based on an auxiliary regression, which is analogous to $s^2(X'X)^{-1}$ generalized to nonlinear regression. `gnr` stands for Gauss–Newton regression.

f. `vce(linearized)`. The variance matrix calculated by a first-order Taylor approximation of the statistic, otherwise known as the Taylor linearized variance estimator, the sandwich estimator, and the White estimator. This is identical to `vce(robust)` in other contexts.

The above are used for defaults. `vce()` may also be

g. `vce(robust)`. The variance matrix calculated by the sandwich estimator of variance, VDV-type calculation, where V is the conventional variance matrix and D is the outer product of the gradients, $\sum_i g_i g_i'$.

h. `vce(cluster` *varname*`)`. The cluster-based version of `vce(robust)` where sums are performed within the groups formed by *varname*, which is equivalent to assuming that the independence is between groups only, not between observations.

i. `vce(hc2)` and `vce(hc3)`. Calculated similarly as `vce(robust)` except that different scores are used in place of the gradient vectors g_i.

j. `vce(opg)`. The variance matrix calculated by the outer product of the gradients; a $(\sum_i g_i g_i')^{-1}$-type calculation.

k. `vce(jackknife)`. The variance matrix calculated by the jackknife, including delete one, delete n, and the cluster-based jackknife.

l. `vce(bootstrap)`. The variance matrix calculated by bootstrap resampling.

You do not need to memorize the above; the documentation for the individual commands, and their corresponding dialog boxes, make clear what is the default and what is available.

22. Estimation commands specified with option `vce(bootstrap)` or `vce(jackknife)` now report a note when a variable is dropped because of collinearity.

23. The new option `collinear`, which has been added to many estimation commands, specifies that the estimation command not remove collinear variables. Typically, you do not want to specify this option. It is for use when you specify constraints on the coefficients such that, even though the variables are collinear, the model is fully identified. See [R] **estimation options**.

24. Estimation commands having a model Wald test composed of more than just the first equation now save the number of equations in the model Wald test in `e(k_eq_model)`.

25. All estimation commands now save macro `e(cmdline)` containing the command line as originally typed.

26. Concerning existing estimation command `ml`;

    a. `ml` now saves the number of equations used to compute the model Wald test in `e(k_eq_model)`, even when option `lf0()` is specified.

    b. `ml score` has new option `missing` that specifies that observations containing variables with missing values not be eliminated from the estimation sample.

    c. `ml display` has new option `showeqns` that requests that equation names be displayed in the coefficient table.

    See [R] **ml**.

27. New command `lpoly` performs a kernel-weighted local polynomial regression and displays a graph of the smoothed values with optional confidence bands; see [R] **lpoly**.

28. New prefix command `nestreg:` reports comparison tests of nested models; see [R] **nestreg**.

29. Existing commands `fracpoly`, `fracgen`, and `mfp` have new features:

    a. `fracpoly` and `mfp` now support `cnreg`, `mlogit`, `nbreg`, `ologit`, and `oprobit`.

    b. `fracpoly` and `mfp` have new option `all` that specifies that out-of-sample observations be included in the generated variables.

    c. `fracpoly, compare` now reports a closed-test comparison between fractional polynomial models by using deviance differences rather than reporting the gain; see [R] **fracpoly**.

    d. `fracgen` has new option `restrict()` that computes adjustments and scaling on a specified subsample.

    See [R] **fracpoly** and [R] **mfp**.

30. For existing postestimation command `hausman`, options `sigmaless` and `sigmamore` may now be used after `xtreg`. These options improve results when comparing fixed- and random-effects regressions based on small to moderate samples because they ensure that the differenced covariance matrix will be positive definite. See [R] **hausman**.

31. Existing postestimation command `testnl` now allows expressions that are bound in parentheses or brackets to have commas. For example, `testnl _b[x] = M[1,3]` is now allowed. See [R] **testnl**.

32. Existing postestimation command `nlcom` has a new option `noheader` that suppresses the output header; see [R] **nlcom**.

33. Existing command `statsby` now works with more commands, including postestimation commands. `statsby` also has new option `forcedrop` for use with commands that do not allow `if` or `in`. `forcedrop` specifies that observations outside the `by()` group be temporarily dropped before the command is called. See [D] **statsby**.

34. Existing command `mkspline` will now create restricted cubic splines as well as linear splines. New option `displayknots` will display the location of the knots. See [R] **mkspline**.

35. In existing command `kdensity`, `kernel(`*kernelname*`)` is now the preferred way to specify the kernel, but the previous method of simply specifying *kernelname* still works. See [R] **kdensity**.

36. Existing command `ktau`'s computations are now faster; see [R] **spearman**.

37. In existing command `ladder`, the names of the transformations in the output have been renamed to match those used by `gladder` and `qladder`. Also, the returned results `r(raw)` and `r(P_raw)` have been renamed to `r(ident)` and `r(P_ident)`, respectively. See [R] **ladder**.

38. Existing command `ranksum` now allows the *groupvar* in option `by(`*groupvar*`)` to be a string; see [R] **ranksum**.

39. Existing command `tabulate, exact` now allows exact computations on larger tables. Also, new option `nolog` suppresses the enumeration log. See [R] **tabulate twoway**.

40. Existing command `tetrachoric`'s default algorithm for computing tetrachoric correlations has been changed from the Edwards and Edwards estimator to a maximum likelihood estimator. Also, standard errors and two-sided significance tests are produced. The Edwards and Edwards estimator is still available by specifying the new `edwards` option. A new `zeroadjust` option requests that frequencies be adjusted when one cell has a zero count. See [R] **tetrachoric**.

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

1. New command `xtset` declares a dataset to be panel data and designates the variable that identifies the panels. In previous versions of Stata, you specified options `i(`*groupvar*`)` and sometimes `t(`*timevar*`)` to identify the panels. You specified the `i()` and `t()` options on the `xt` command you wanted to use. Now you `xtset` *groupvar* or `xtset` *groupvar timevar* first. The values you set will be remembered from one session to the next if you save your dataset.

   `xtset` also provides a new feature. `xtset` allows option `delta()` to specify the frequency of the time-series data, something you will need to do if you are using Stata's new date/time variables.

   Finally, you can still specify old options `i()` and `t()`, but they are no longer documented. Similarly, old commands `iis` and `tis` continue to work but are no longer documented. See [XT] **xtset**.

2. New estimation commands `xtmelogit` and `xtmepoisson` fit nested, hierarchical, and mixed models with binary and count responses; i.e., you can fit logistic and Poisson models with complex, nested error components. Syntax is the same as for Stata's linear mixed model estimator, `xtmixed`. To fit a model of graduation with a fixed coefficient on `x1` and random coefficient on `x2` at the school level, and with random intercepts at both the school and class-within-school level, you type

   ```
   . xtmelogit graduate x1 x2 || school: x2 || class:
   ```

   `predict` after `xtmelogit` and `xtmepoisson` will calculate predicted random effects. See [XT] **xtmelogit**, [XT] **xtmelogit postestimation**, [XT] **xtmepoisson**, and [XT] **xtmepoisson postestimation**.

3. New estimation commands are available for fitting dynamic panel-data models:

   a. Existing estimation command `xtabond` fits dynamic panel-data models by using the Arellano–Bond estimator but now reports results in levels rather than differences. Also, `xtabond` will now compute the Windmeijer biased-corrected two-step robust VCE. See [XT] **xtabond**.

   b. New estimation command `xtdpdsys` fits dynamic panel-data models by using the Arellano–Bover/Blundell–Bond system estimator. `xtdpdsys` is an extension of `xtabond` and produces estimates with smaller bias when the AR process is too persistent. `xtpdsys` is also more efficient than `xtabond`. Whereas `xtabond` uses moment conditions based on the differenced errors in producing results, `xtpdsys` uses moment conditions based on differences and levels. See [XT] **xtdpdsys**.

   c. New estimation command `xtdpd` fits dynamic panel-data models extending the Arellano–Bond or the Arellano–Bover/Blundell–Bond system estimator and allows a richer syntax for specifying models and so will fit a broader class of models then either `xtabond` or `xtdpdsys`. `xtdpd` can be used to fit models with serially correlated idiosyncratic errors, whereas `xtdpdsys` and `xtabond` assume no serial correlation. `xtdpd` can be used with

models where the structure of the predetermined variables is more complicated than that assumed by `xtdpdsys` or `xtabond`. See [XT] **xtdpd**.

d. New postestimation command `estat abond` tests for serial correlation in the first-differenced errors. See [XT] **xtabond postestimation**, [XT] **xtdpdsys postestimation**, and [XT] **xtdpd postestimation**.

e. New postestimation command `estat sargan` performs the Sargan test of overidentifying restrictions. See [XT] **xtabond postestimation**, [XT] **xtdpdsys postestimation**, and [XT] **xtdpd postestimation**.

4. Existing estimation command `xtreg, fe` now accepts `aweights`, `fweights`, and `pweights`. Also, new option `dfadj` specifies that the cluster–robust VCE be adjusted for the within transform. This was previously the default behavior. See [XT] **xtreg**.

5. Existing estimation commands `xtreg, fe` and `xtreg, re` used to be willing to produce cluster–robust VCEs when the panels were not nested within the clusters. Sometimes this VCE is consistent and other times it is not. You must now specify the new `nonest` option to obtain a cluster–robust VCE when the panels are not nested within the clusters.

6. The numerical method used to evaluate distributions, known as quadrature, has been improved. This method is used by the `xt` random-effects estimation commands `xtlogit`, `xtprobit`, `xtcloglog`, `xtintreg`, `xttobit`, and `xtpoisson, re normal`.

   a. For the estimation commands, the default method is now `intmethod(mvaghermite)`. The old default was `intmethod(aghermite)`.

   b. Option `intpoints(#)` for the commands now allows up to 195 quadrature points. The default is 12, and the old upper limit was 30. (Models with large random effects often require more quadrature points.)

   c. The estimation commands may now be used with constraints regardless of the quadrature method chosen.

   d. Command `quadchk`, for use after estimation to verify that the quadrature approximation was sufficiently accurate, now produces a more informative comparison table. Before, four fewer and four more quadrature points were used, and that was reasonable if the number of quadrature points was, say, $n_q = 12$. Now you may specify significantly larger n_q and the ± 4 is not useful. Now `quadchk` uses $n_q - \texttt{int}(n_q/3)$ and $n_q + \texttt{int}(n_q/3)$.

   e. `quadchk` has new option `nofrom` that forces refitted models to start from scratch rather than starting from the previous estimation results. This is important if you use the old `intmethod(aghermite)`, which is sensitive to starting values, but not important if you are using the new default `intmethod(mvaghermite)`.

   See [XT] **quadchk**.

7. All `xt` estimation commands now accept option `vce(`*vcetype*`)`. As mentioned in the *What's new in statistics (general)*, `vce(robust)` and `vce(cluster` *varname*`)` are the right ways to specify the old `robust` and `cluster()` options, and new option `vce()` allows other VCE calculations as well.

8. Existing estimation command `xtcloglog` has new option `eform` that requests exponentiated coefficients be reported; see [XT] **xtcloglog**.

9. Existing estimation command `xthtaylor` now allows users to specify only endogenous time-invariant variables, only endogenous time-varying variables, or both. Previously, both were required. See [XT] **xthtaylor**.

10. All `xt` estimation commands have new option `collinear`, which specifies that collinear variables are not to be removed. Typically, you do not want to specify this option. It is for use when you specify constraints on the coefficients such that, even though the variables are collinear, the model is fully identified. See [XT] **estimation options**.
11. Existing command `xtdes` has been renamed to `xtdescribe`. `xtdes` continues to work as a synonym for `xtdescribe`. See [XT] **xtdescribe**.
12. The [XT] manual has an expanded glossary.

1.3.5 What's new in statistics (time series)

1. All time-series analysis commands now support data with frequencies as high as 1 millisecond (ms), corresponding to Stata's new date/time variables. Since your data are probably not recorded at the millisecond level, existing command `tsset` has new option `delta()` that allows you to specify the frequency of your data. Previously, time was recorded as t_0, $t_0 + 1$, $t_0 + 2$, ..., and if time $= t$ in some observation, then the corresponding lagged observation was the observation for which time $= t - 1$. That is still the default. When you specify `delta()`, time is assumed to be recorded as t_0, $t_0 + \delta$, $t_0 + 2\delta$, and if time $= t$ in some observation, then the corresponding lagged observation is the observation for which time $= t - \delta$. Say that you are analyzing hourly data and time is recorded using Stata's new `%tc` values. One hour corresponds to 3,600,000 ms, and you would want to specify `tsset t, delta(3600000)`. Option `delta()` is smart; you can specify `tsset t, delta(1 hour)`. See [TS] **tsset**.
2. `tsset` now reports whether panels are balanced when an optional panel variable is specified.
3. Many `ts` estimation commands now accept option `vce(`*vcetype*`)`. As mentioned in *What's new in statistics (general)*, `vce(robust)` and `vce(cluster` *varname*`)` are the right ways to specify the old `robust` and `cluster()` options, and option `vce()` allows other VCE calculations as well.
4. Options `vce(hc2)` and `vce(hc3)` are now the preferred way to request alternative bias corrections for the robust variance calculation for existing estimation command `prais`. See [TS] **prais**.
5. Existing estimation commands `arch` and `arima` have new option `collinear` that specifies that the estimation command not remove collinear variables. Typically, you do not want to specify this option. It is for use when you specify constraints on the coefficients such that, even though the variables are collinear, the model is fully identified. See [TS] **estimation options**.
6. Existing command `irf` now estimates and reports dynamic–multiplier functions and cumulative dynamic–multiplier functions, as well as their standard errors. See [TS] **irf**.
7. The [TS] manual has an expanded glossary.

(*Continued on next page*)

1.3.6 What's new in statistics (survey)

1. Stata's `svy:` prefix now works with 48 estimators, 28 more than previously. Most importantly, `svy:` now works with Cox regression (`stcox`) and parametric survival models (`streg`). Other commands with which `svy:` now works include

`biprobit`	bivariate probit regression
`clogit`	conditional (fixed effects) logistic regression
`cloglog`	complementary log-log regression
`cnreg`	censored-normal regression
`cnsreg`	constrained linear regression
`glm`	generalized linear models
`hetprob`	heteroskedastic probit model
`ivregress`	instrumental-variables regression
`ivprobit`	probit model with endogenous regressors
`ivtobit`	tobit model with endogenous regressors
`mprobit`	multinomial probit regression
`nl`	nonlinear least-squares estimation
`scobit`	skewed logistic regression
`slogit`	stereotype logistic regression
`stcox`	Cox proportional hazards regression
`streg`	parametric survival models (5 estimators)
`tobit`	tobit regression
`treatreg`	treatment-effects model
`truncreg`	truncated regression
`zinb`	zero-inflated negative binomial regression
`zip`	zero-inflated Poisson regression
`ztnb`	zero-truncated negative binomial regression
`ztp`	zero-truncated Poisson regression

   See [SVY] **svy estimation**.

2. `svy:` prefix now calculates the linearized variance estimator 2 to 100 times faster, the larger multiplier applying to large datasets with many sampling units; see [SVY] **svy**.

3. `svy: mean`, `svy: proportion`, `svy: ratio`, and `svy: total` are considerably faster when the `over()` option identifies many subpopulations.

4. `svy:`, `svy: mean`, `svy: proportion`, `svy: ratio`, and `svy: total` now take advantage of multiple processors in Stata/MP, making them even faster in that case.

5. Concerning `svyset`,

   a. New option `singleunit(`*method*`)` provides three methods for handling strata with one sampling unit. If not specified, the default in such cases is to report standard errors as missing value.

   b. New option `fay(#)` specifies that Fay's adjustment be made to the BRR weights.

   See [SVY] **svyset**.

6. `estat` has two new subcommands for use with `svy` estimation results:

   a. `estat sd`, used after `svy: mean`, reports subpopulation standard deviations.

   b. `estat strata` reports the number of singleton and certainty strata within each sampling stage.

   See [SVY] **estat**.

7. `svy: tabulate` now allows string variables. See [SVY] **svy: tabulate oneway** and [SVY] **svy: tabulate twoway**.

8. Existing command `svydes` has been renamed `svydescribe`; `svydes` continues to work. `svydescribe` now puts missing values in the `generate(`*newvar*`)` variable for observations outside the specified estimation sample. Previously, the variable would contain a zero for observations outside the estimation sample. See [SVY] **svydescribe**.

9. The [SVY] manual has been reorganized. Stata's survey estimation commands are now documented in [SVY] **svy estimation**. All model-specific information is now documented in the manual entry for the corresponding estimation command.

1.3.7 What's new in statistics (survival analysis)

1. Existing estimation commands `stcox` and `streg` may now be used with the `svy:` prefix and so can fit models for complex survey data; see [ST] **stcox** and [ST] **streg**.

2. New command `stpower` provides sample-size and power calculations for survival studies that use Cox proportional hazards regressions, log-rank tests for two groups, or differences in exponentially distributed hazards or log hazards.

    a. `stpower cox` estimates required sample size (given power) or power (given sample size) or the minimal detectable coefficient (given power and sample size) for models with multiple covariates. The command provides options to account for possible correlation between the covariate of interest and other predictors and for withdrawal of subjects from the study. See [ST] **stpower cox**.

    b. `stpower logrank` estimates required sample size (given power) or power (given sample size) or the minimal detectable hazard ratio (given power and sample size) for studies comparing survivor functions of two groups by using the log-rank test. Both the Freedman (1982) and the Schoenfeld (1981) methods are provided. The command allows for unequal allocation of subjects between the groups and possible withdrawal of subjects. Estimates can be adjusted for uniform accrual. See [ST] **stpower logrank**.

    c. `stpower exponential` estimates sample size (given power) or power (given sample size) of tests of the difference between hazards or log hazards of two groups under the assumption of exponential survivor functions (also known as the exponential test). Both the Lachin–Foulkes (1986) and Rubinstein–Gail–Santner(1981) methods are provided. Unequal group allocation, uniform or truncated exponential accrual, and different exponential losses due to follow-up in each group are allowed. See [ST] **stpower exponential**.

    The `stpower` commands allow automated production of customizable tables and have options to assist with creating graphs of power curves. See [ST] **stpower**.

3. Concerning existing command `sts graph`,

    a. New option `risktable()` places a subjects-at-risk table underneath and aligned with the survivor or hazard plot.

    b. New option `ci` replaces old options `gwood`, `cna`, and `cihazard`. `sts graph` will choose the appropriate confidence interval on the basis of the function being graphed.

    c. Confidence intervals are now graphed using shaded areas and new options `plotopts()` and `ciopts()` allow you to control how plots and confidence intervals look.

    d. Overlaid confidence intervals are now allowed and are produced when new option `ci` is combined with existing option `by(`*varlist*`)`.

e. New option `censopts()` controls the appearance of ticks and markers produced by existing option `censored()`.

f. Boundary computations for smoothing hazards have been improved. New option `noboundary` specifies that no boundary correction be done.

g. The lower bound of the range to plot the hazard function now extends to zero.

h. Option `na` has been renamed `cumhaz`. `na` may still be used.

See [ST] **sts graph**. Setting `version` to less than 10 restores previous behavior.

4. For `sts list`, option `na` has been renamed `cumhaz`. `na` may be used as a synonym for `cumhaz`. See [ST] **sts list**.

5. Improvements to `stcurve` analogous to those of `sts graph` have been made.

   a. Boundary computations for smoothing hazards have been improved. New option `noboundary` specifies that no boundary correction be done.

   b. The lower bound of the range to plot the hazard function now extends to zero.

   See [ST] **stcurve**.

6. All `st` estimation commands accept option `vce(`*vcetype*`)`. As mentioned in the *What's new in statistics (general)*, `vce(robust)` and `vce(cluster` *varname*`)` are the right ways to specify the old `robust` and `cluster()` options, and option `vce()` now allows other VCE calculations as well.

7. Existing command `predict` after `stcox` has a new option, `scores`, that allows generating variables with the partial efficient score residuals; see [ST] **stcox postestimation**.

8. Existing command `ltable` has new options `byopts()`, `plotopts()`, `plot#opts()`, and `ci#opts()` that allow for more customization of the graph. New option `ci` adds confidence intervals to the graph. See [ST] **ltable**.

9. Existing command `stphplot` has a new option `plot#opts()` that allows for more customization of the graph. See [ST] **stcox diagnostics**.

10. Existing command `stcoxkm` has new options `byopts()`, `obsopts()`, `obs#opts()`, `predopts()`, and `pred#opts()` that allow for more customization of the graph. See [ST] **stcox diagnostics**.

11. Existing command `cc` has new option `tarone` that produces Tarone's (1985) adjustment of the Breslow–Day test for homogeneity of odds ratios. See [ST] **epitab**.

12. Existing command `stdes` has been renamed to `stdescribe`. `stdes` continues to work. See [ST] **stdescribe**.

13. The [ST] manual has an expanded glossary.

1.3.8 What's new in statistics (multivariate)

1. New estimation commands `discrim` and `candisc` provide several discriminant analysis techniques, including linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), logistic discriminant analysis, and kth nearest neighbor discrimination. See [MV] **discrim**, [MV] **discrim estat**, and [MV] **candisc**.

2. Existing estimation commands `mds`, `mdslong`, and `mdsmat` now provide modern as well as classical multidimensional scaling (MDS), including metric and nonmetric MDS. Available loss functions include stress, normalized stress, squared stress, normalized squared stress, and Sammon. Available transformations include identity, power, and monotonic. `mdslong` also now allows `aweights` and `fweights`, and `mdsmat` has a `weight()` option. See [MV] **mds**, [MV] **mdslong**, and [MV] **mdsmat**.

3. New estimation command `mca` provides multiple correspondence analysis (MCA) and joint correspondence analysis (JCA); see [MV] **mca** and [MV] **mca postestimation**. You can use existing command `screeplot` afterward to graph principal inertias; see [MV] **screeplot**.
4. Concerning existing estimation command `ca` (correspondence analysis),
   a. `ca` now allows crossed (stacked) variables. This provides a way to automatically combine two or more categorical variables into one crossed variable and perform correspondence analysis with it.
   b. `ca`'s existing option `normalize()` now allows `normalize(standard)` to provide normalization of the coordinates by singular vectors divided by the square root of the mass.
   c. `ca`'s new option `length()` allows you to customize the length of labels with crossed variables in output.
   d. New postestimation command `estat loadings`, used after `ca` and `camat`, displays correlations of profiles and axes.
   e. Existing postestimation command `cabiplot` has new option `origin` that displays the origin within the plot. `cabiplot` also now accepts `originlopts(`*line_options*`)` to customize the appearance of the origin on the graph.
   f. Existing postestimation commands `cabiplot` and `caprojection` now allow row and column marker labels to be specified using the `mlabel()` suboption of `rowopts()` and `colopts()`.

   See [MV] **ca** and [MV] **ca postestimation**.
5. Existing commands `cluster`, `matrix dissimilarity`, and `mds` now allow the Gower measure for a mix of binary and continuous data; see [MV] ***measure_option***.
6. Existing command `biplot` has new options. `dim()` specifies the dimensions to be displayed. `negcol` specifies that negative column (variable) arrows be plotted. `negcolopts(`*col_options*`)` provides graph options for the negative column arrows. `norow` and `nocolumn` suppress the row points or column arrows. See [MV] **biplot**.
7. New postestimation command `estat rotate` after `canon` performs orthogonal varimax rotation of the raw coefficients, standard coefficients, or canonical loadings. After `estat rotate`, new postestimation command `estat rotatecompare` displays the rotated and unrotated coefficients or loadings and the most recently rotated coefficients or loadings. See [MV] **canon postestimation**.
8. Existing commands `pcamat` and `factormat` now allow singular correlation or covariance structures. New option `forcepsd` modifies a matrix to be positive semidefinite and thus to be a proper covariance matrix. See [MV] **pca** and [MV] **factor**.
9. Existing commands `rotate` and `rotatemat` now refer to "Kaiser normalization" rather than "Horst normalization". A search of the literature indicates that Kaiser normalization is the preferred terminology. Previously option `horst` was a synonym for `normalize`. Now option `horst` is not documented. See [MV] **rotate** and [MV] **rotatemat**.
10. Existing command `procrustes` now saves the number of y variables in scalar `e(ny)`; see [MV] **procrustes**.

1.3.9 What's new in graphics

1. Stata 10 has an interactive, point-and-click editor for your graphs. You do not need to type anything; you just right-click within the Graph window and select **Start Graph Editor**. You can do that any time, either when the graph is drawn or when you have `graph use`d it from disk. You can add text, lines, markers, titles, and annotations, outside the plot region or inside; you can move

axes, titles, legends, etc.; you can change colors and sizes, number of tick marks, etc.; and you can even change scatters to lines or bars, or vice versa. See [G] **graph editor**.

2. New command `graph twoway lpoly` plots a local polynomial smooth; see [G] **graph twoway lpoly**. New command `graph twoway lpolyci` plots a local polynomial smooth along with a confidence interval; see [G] **graph twoway lpolyci**.
3. Concerning command `graph twoway`,
   a. `graph twoway` now allows more than 100 variables to be plotted.
   b. New suboption `custom` of *axis_label_options* allows you to create custom axis ticks and labels that have a different color, size, tick length, etc., from the standard ticks and labels on the axis. Such custom ticks can be used to emphasize points in the scale, such as important dates, physical constants, or other special values. See the `custom` suboption in [G] ***axis_label_options***.
   c. New suboption `norescale` of *axis_label_options* specifies that added ticks and labels be placed directly on the graph without rescaling the axis or associated plot region for the new values; see [G] ***axis_label_options***.
   d. New advanced options `yoverhangs` and `xoverhangs` adjust the graph region margins to prevent long labels on the y or x axis from extending off the edges of the graphs; see [G] ***advanced_options***.
4. `graph twoway pcarrow` and `graph twoway pcbarrow` may now be drawn on plot regions with log scales or reversed scales; see [G] **graph twoway pcarrow**.
5. `graph bar` and `graph dot` no longer require user-provided names when a variable is repeated with more than one statistic; see [G] **graph bar** and [G] **graph dot**.
6. `graph twoway lfit` and `graph twoway qfit` now use value labels to annotate the x axis when using the existing suboption `valuelabel` of `xmlabel()`; see [G] ***axis_label_options***.
7. `graph export` has new options `width(#)` and `height(#)` that specify the width and height of the graph when exporting to PNG or TIFF, thus allowing the resolution to be greater than screen resolution; see [G] ***png_options*** and [G] ***tif_options***.
8. `graph twoway area` and `graph twoway rarea` now allow option `cmissing()` to control whether missing values produce breaks in the areas or are ignored; see [G] **graph twoway area** and [G] **graph twoway rarea**.
9. Typing `help graph` *option* now displays the help file for the specified option of the `graph` command. See [R] **help**.

1.3.10 What's new in programming

1. First, a warning for time-series programmers: Stata's new date/time values, which contain the number of milliseconds since 01jan1960 00:00:00, result in large numbers. 21apr2007 12:14:07.123 corresponds to 1,492,776,847,123. Date/time values must be stored as `doubles`. Programmers should use scalars to store these values whenever possible. If you must use a macro, exercise caution that the value is not rounded. It would not do at all for 1,492,776,847,123 to be recorded as "`1.493e+12`" (which would be 24apr2007 02:13:20). If you must use macros, our recommendations are
   a. If a date/time value is stored in one macro and you need it in another, code

```
local new `old'
```

b. If a date/time value is the result of an expression, and you must store it as a macro, code

```
local new = string(exp, "%21x")
```

or

```
local new : display %21x (exp)
```

Now we will continue with What's new.

2. Stata for Windows now supports Automation, formerly known as OLE Automation, which means that programmers can control Stata from other applications and retrieve results. See [P] **automation**.

3. New command `confirm {numeric | string | date} format` verifies that the format is of the specified type; see [P] **confirm**.

4. New function `fmtwidth(`*s*`)` returns the display width of a *%fmt* string, including date formats; see *Programming functions* in [D] **functions**.

5. Expression limits have been increased in Stata/MP, Stata/SE, and Stata/IC. The limit on the number of dyadic operators has increased from 500 to 800, and the limit on the number of numeric literals has increased from 150 to 300. See `help limits`.

6. Intercooled Stata has been renamed to Stata/IC. `c(flavor)` now contains "`IC`" rather than "`Intercooled`" if version $\geq$ 10. Backward-compatibility old global macro `$S_FLAVOR` continues to contain "`Intercooled`". See [P] **creturn** and [P] **macro**.

7. `c()` now contains values associated with Stata/MP: `c(MP)` (1 or 0 depending on whether you are running Stata/MP), `c(processors)` (the number of processors Stata will use), `c(processors_mach)` (the number of processors on the computer), `c(processors_lic)` (the maximum number of processors the license will allow you to use), and `c(processors_max)` (the maximum number of processors that could be used on this computer with this license).

8. New command `include` is a variation on `do` and `run`. `include` executes the commands stored in a file just as if they were entered from the keyboard or the current do-file. It is most often used in advanced Mata situations to create the equivalent of `#include` files. See [P] **include**.

9. New commands `signestimationsample` and `checkestimationsample` are useful in writing estimation/postestimation commands that need to identify the estimation sample; see [P] **signestimationsample**.

10. New command `_datasignature` is the building block for Stata's `datasignature` command and the programming commands `signestimationsample` and `checkestimationsample`. In advanced situations, you may wish to call it directly. See [P] **_datasignature**.

11. New extended macro function `:copy` copies one macro to another and is faster when the macro being copied is long. That is, coding

```
local new : copy local old
```

is faster than

```
local new `old'
```

See [P] **macro**.

12. New command `timer` times sections of code; see [P] **timer**.

13. Existing command `matrix accum` is now faster when some observations contain zeros; see [P] **matrix accum**.

14. Existing command `ml display` has new option `showeqns` that requests that equation names be displayed in the coefficient table; see [R] **ml**.

15. Existing command `mkmat` has new options `rownames()`, `roweq()`, `rowprefix()`, `obs`, and `nchar()` that specify the row names of the created matrix; see [P] **matrix mkmat**.

16. Existing command `_rmdcoll`'s `nocollin` option has been renamed to `normcoll`. `nocollin` will continue to work as a synonym for `normcoll`. See [P] **_rmcoll**.

17. Existing command `describe`'s option `simple` no longer saves the names of the variables in `r(varlist)`; you must specify option `varlist` if you want that. Also, existing command `describe using` *filename* now allows options `simple` and `varlist`. See [D] **describe**.

18. New extended macro function `adosubdir` returns the subdirectory in which Stata would search for a file along the ado-path. Determining the subdirectory in which Stata stores files is now a function of the file's extension. `adosubdir` returns the subdirectory in which to look. See [P] **macro**.

19. Existing command `syntax` { [*optionname*`(real ...)`] } now returns the number specified in `%18.0g` format if `version` is set to 10.0 or higher. For `version` less than 10, the number is returned in `%9.0g` format. See [P] **syntax**.

20. New functions `_byn1()` and `_byn2()`, available within a `byable(recall)` program, return the beginning and ending observation numbers of the by-group currently being executed; see [P] **byable**.

21. Existing command `program drop` may now specify `program drop _allado` to drop programs that were automatically loaded from ado-files; see [P] **program**.

22. Concerning SMCL,

    a. Existing directive `{synoptset}` has new optional argument `notes` that specifies that some of the table entries will have footnotes and results in a larger indentation of the first column.

    b. Existing directive `{p}` now has an optional fourth argument specifying the paragraph's width.

    See [P] **smcl**.

23. Concerning classes, you can now define an `oncopy` member program to perform operations when a copy of an object is being created. See [P] **class**.

24. Concerning programmable menus, the maximum number of menu items that can be added to Stata has increased to 1,250 from 1,000; see `help window programming`.

25. Concerning programmable dialogs,

    a. Child dialogs can now be created.

    b. New control `TEXTBOX` allows displaying multiline text.

    c. In the dialog programming language, (1) `if` now allows `else` and (2) new command `close` closes the dialog programmatically.

    d. Messages can be passed to dialogs when they are launched; see `help db`.

    e. Dialogs can now be designated as modal, meaning that this dialog must be dealt with by the user before new dialogs (other than children) can be launched.

    f. Several controls have new options and new member programs. For instance, `FILE` and `LISTBOX` now have option `multiselect`, which lets the user pick more than one item.

    See `help dialog programming`.

26. Stata's help files are now named `*.sthlp` rather than `*.hlp`, meaning that user-written help files can be sent via email more easily. Many email filters flag `.hlp` files as potential virus carriers because Stata was not the only one to use the `.hlp` suffix. You need not rename your old help files. See [R] **help**.

27. Two new C functions have been exposed from Stata for use by plugins: `sstore()` and `sdata()`. `sstore()` stores string data in the Stata dataset and `sdata()` reads them. See http://www.stata.com/plugins/.

1.3.11 What's new in Mata

1. New Stata command `include` is a variation on `do` and `run` and is useful for implementing `#include` for header files in advanced programming situations. See [P] **include** and type `viewsource optimize.mata` for an example of use.
2. Mata now has structures, which will be of special interest to those writing large systems. See [M-2] **struct** and [M-5] **liststruct( )**.
3. Mata now engages in more thorough type checking, and produces better code, for those who explicitly declare arguments and variables.
4. Mata inherits all of Stata's formats and functions for dealing with the new date/time variables and values; see [M-5] **date( )** and [M-5] **fmtwidth( )**.
5. New functions `inbase()` and `frombase()` perform base conversions; see [M-5] **inbase( )**.
6. New function `floatround()` returns values rounded to float precision. This is Mata's equivalent of Stata's `float()` function. See [M-5] **floatround( )**.
7. New function `nameexternal()` returns the name of an external; see [M-5] **findexternal( )**.
8. Concerning matrix manipulation,
   a. Matrix multiplication is now faster when one of the matrices contains many zeros, as is function `cross()`.
   b. Appending rows or columns to a matrix using `,` and `\` is now faster.
   c. New function `_diag()` replaces the principal diagonal of a matrix with a specified vector or scalar; see [M-5] **_diag( )**.
   d. New functions `select()` and `st_select()` select rows or columns of a matrix on the basis of a criterion; see [M-5] **select( )**.
   e. Existing functions `rowsum()`, `colsum()`, `sum()`, `quadrowsum()`, `quadcolsum()`, and `quadsum()` now allow an optional second argument that determines how missing values are handled; see [M-5] **sum( )**.
   f. New functions `runningsum()`, `quadrunningsum()`, `_runningsum()`, and `_quadrunningsum()` return the running sum of a vector; see [M-5] **runningsum( )**.
   g. New functions `minindex()` and `maxindex()` return the indices of minimums and maximums (including tied values) of a vector; see [M-5] **minindex( )**.
9. Concerning statistics,
   a. New Mata function `optimize()` performs minimization and maximization. You can code just the function, the function and its first derivatives, or the function and its first and second derivatives. Optimization techniques include Newton–Raphson, Davidon–Fletcher–Powell, Broyden–Fletcher–Goldfarb–Shanno, Berndt–Hall–Hall–Hausman, and the simplex method Nelder–Mead. See [M-5] **optimize( )**.
   b. New function `cvpermute()` forms permutations; see [M-5] **cvpermute( )**.
   c. New function `ghk()` provides the Geweke–Hajivassiliou–Keane multivariate normal simulator; see [M-5] **ghk( )**. New function `ghkfast()` is faster but a little more difficult to use; see [M-5] **ghkfast( )**.

d. New functions `halton()`, `_halton()`, and `ghalton()` compute Halton and Hammersley sets; see [M-5] **halton()**.

e. The new density and probability functions available in Stata are also available in Mata, including `binomial()`, `binomialtail()`, `gammaptail()`, `invgammaptail()`, `invbinomialtail()`, `ibetatail()`, `invibetatail()`, `lnnormal()`, and `lnnormalden()`; see [M-5] **normal()**. Also, as in Stata, convergence and accuracy of many of the cumulatives, reverse cumulatives, and density functions have been improved.

f. Existing Mata functions `mean()`, `variance()`, `quadvariance()`, `meanvariance()`, `quadmeanvariance()`, `correlation()`, and `quadcorrelation()` now make the weight argument optional. If not specified, unweighted estimates are returned. See [M-5] **mean()**.

10. Concerning string processing,

a. New function `stritrim()` replaces multiple, consecutive internal spaces with one space; see [M-5] **strtrim()**.

b. New functions `strtoreal()` and `_strtoreal()` convert strings to numeric values; see [M-5] **strtoreal()**.

c. New function `_substr()` substitutes a substring into an existing string; see [M-5] **_substr()**.

d. New function `invtokens()` is the inverse of the existing function `tokens()`; see [M-5] **invtokens()**.

e. New function `tokenget()` performs advanced parsing; see [M-5] **tokenget()**.

11. Concerning I/O,

a. New function `adosubdir()` returns the subdirectory in which Stata would search for a file; see [M-5] **adosubdir()**. New function `pathsearchlist(`*fn*`)` returns a row vector of full paths/filenames specifying all the locations, in order, where Stata would look for the specified *fn* along the official Stata ado-path; see [M-5] **pathjoin()**.

b. New function `byteorder()` returns 1 if the computer is HILO and returns 2 if the computer is LOHI; see [M-5] **byteorder()**.

c. New functions `bufput()` and `bufget()` copy elements into and out of buffers; see [M-5] **bufio()**.

d. Existing function `cat()` now allows optional second and third arguments that specify the beginning and ending lines of the file to read; see [M-5] **cat()**.

12. New functions `ferrortext()` and `freturncode()` obtain error messages and return codes following an I/O error; see [M-5] **ferrortext()**.

13. Concerning the Stata interface,

a. New function `stataversion()` returns the version of Stata that you are running, and new function `statasetversion()` allows setting it. See [M-5] **stataversion()**.

b. New function `setmore()` allows turning `more` on and off. New function `setmoreonexit()` allows restoring `more` to its original setting when execution ends. See [M-5] **more()**.

1.3.12 What's more

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

Stata is continually being updated, and those updates are available for free over the Internet. All you have to do is type

```
. update query
```

and follow the instructions.

To learn what has been added since this manual was printed, select **Help > What's new?** or type

```
. help whatsnew
```

We hope that you enjoy Stata 10.

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. 2006. *The Cambridge Dictionary of Statistics.* 3rd ed. Cambridge: Cambridge University Press.

Freedman, L. S. 1982. Tables of the number of patients required in clinical trials using the logrank test. *Statistics in Medicine* 1: 121–129.

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.* New York: Wiley.

Lachin, J. M., and M. A. Foulkes. 1986. Evaluation of sample size and power for analysis of survival with allowance for nonuniform patient entry, losses to follow-up, noncompliance, and stratification. *Biometrics* 42: 507–519.

Rubinstein, L. V., M. H. Gail, and T. J. Santner. 1981. Planning the duration of a comparative clinical trial with loss to follow-up and a period of continued observation. *Journal of Chronic Diseases* 34: 469–479.

Schoenfeld, D. 1981. The asymptotic properties of nonparametric tests for comparing survival distributions. *Biometrika* 68: 316–319.

Tarone, R. E. 1985. On heterogeneity tests based on efficient scores. *Biometrika* 72: 91–95.

Upton, G., and I. Cook. 2006. *A Dictionary of Statistics.* 2nd ed. Oxford: Oxford University Press.