The Stata Journal

The *Stata Journal*, electronic version (ISSN 1536-8734) is a publication of Stata Press. Stata and Mata are registered trademarks of StataCorp LP.

The Stata Journal (2007)
7, Number 2, pp. 227–244

Making regression tables simplified

Ben Jann
ETH Zurich
Zurich, Switzerland
jann@soz.gess.ethz.ch

Abstract. `estout`, introduced by Jann (*Stata Journal* 5: 288–308), is a useful tool for producing regression tables from stored estimates. However, its syntax is relatively complex and commands may turn out long even for simple tables. Furthermore, having to store the estimates beforehand can be cumbersome. To facilitate the production of regression tables, I therefore present here two new commands called `eststo` and `esttab`. `eststo` is a wrapper for official Stata's `estimates store` and simplifies the storing of estimation results for tabulation. `esttab`, on the other hand, is a wrapper for `estout` and simplifies compiling nice-looking tables from the stored estimates without much typing. I also provide updates to `estout` and `estadd`.

Keywords: st0085_1, csv, estadd, estimates, estout, eststo, esttab, excel, html, latex, regression table, rtf, word

1 Introduction

Statistical software packages such as Stata provide a rich variety of statistical estimation routines, all producing some kind of output. Although the outputs from single commands are important for scientific research, they are usually not well designed for presentation. One must often select and rearrange results from different outputs to get an overview of the results and present a clear and concise analysis. Therefore, not only are statistical routines necessary but also tools to efficiently process results for presentation.

Jann (2005) provided such a tool called `estout`. `estout` compiles regression tables containing results from one or more estimation commands for use in, say, LaTeX documents, spreadsheet programs, or word processors. The initial motivation behind `estout` was to provide a generic program to compile a table from several sets of regression estimates and write the table to disk for use with other software. Developmental efforts were directed more toward functionality—to be able to flexibly arrange and format the table—than toward ease of use. Furthermore, since there are different needs and conventions concerning the contents and look of a regression table, the basic approach was to provide a *clean desk* for users from which they could start building up their full-fledged end product. Clean desk means here that `estout` was designed to produce a plain, essentially unformatted table containing only point estimates by default.

Although `estout` is powerful in functionality, the motivational orientation just outlined brought with it some limitations. These limitations may be summarized as follows:

1. `estout` tables are usually not suitable for display in Stata's Results window. For example, by default, `estout` uses the tab character to separate the table's columns. However, tab characters are expanded to a fixed number of blanks in the Results window, causing the table's columns to appear misaligned. This appearance is unfavorable because users often need to produce regression tables on the fly, for quick result inspection on screen.

2. `estout`'s syntax is not as intuitive and user friendly as it could be. For example, there are nested options, which do their job but are hard to handle and remember. Also even experienced users are often forced to consult the command's documentation while producing an `estout` table.

3. The amount of typing required to compile even a simple table can be considerable. Users generally have to specify many options to determine the content and formatting of the table according to their needs. `estout` provides the possibility to specify options via so-called default files (similar to scheme files for Stata graphics). However, maintaining such default files does not appear beneficial unless one is working on large reports containing many similar tables.

A consequence of these limitations is that the use of `estout` may be somewhat clumsy in daily work. Smooth application of `estout` is also compromised by the fact that the estimation sets have to be stored using official Stata's `estimates store` before they can be tabulated (at least if there is more than one set of estimates). A drawback of `estimates store` is that it requires the user to specify names under which to store the estimation sets. Having to provide such names, although sensible in some contexts, can be distracting.

In summary, there appears to be a need for (1) an easy-to-use version of `estout` that produces fully formatted tables immediately and is suitable for interactive work and (2) a simplified procedure to hold on to estimates for tabulation. In the rest of this article, I will address these two points (in reverse order) and present in section 2 a command called `eststo` to overcome the limitations of `estimates store`. Section 3 then introduces a user-friendly `estout` wrapper called `esttab` and illustrates its application by examples. The appendix (section 4) contains syntax overviews for the two commands and provides updates to `estout` and `estadd`.

Space limitations do not allow an extended treatment of the new commands. For details and more examples, consult the online help or visit the `estout` web site at http://fmwww.bc.edu/repec/bocode/e/estout/.

2 eststo: Storing estimates simplified

The new `eststo` command stores a copy of the active estimation results for later tabulation. It is an alternative to official Stata's `estimates store`. The basic syntax of `eststo` is

`eststo` [*name*] [`,` *options*] [`:` *estimation command*]

Store estimates without providing a name

A main advantage of `eststo` over `estimates store` is that `eststo` does not require the user to specify a name for the stored estimates. If *name* is provided, then, naturally, the estimates are stored under this name. However, if no name is provided, `eststo` makes up its own name. `eststo` keeps track of the names of the stored estimation sets via global macros from where the names can be picked up by, say, `estout`. Here is an example:

```
. sysuse auto
(1978 Automobile Data)
. regress price weight mpg
  (output omitted)
. eststo
(est1 stored)
. regress price weight mpg foreign
  (output omitted)
. eststo
(est2 stored)
. estout, style(fixed)
                      est1         est2
                         b            b
weight           1.746559     3.464706
mpg             -49.51222      21.8536
foreign                         3673.06
_cons            1946.069    -5853.696
```

To erase the estimation sets that have been stored by `eststo`, type

```
. eststo clear
```

Use eststo as a prefix command

As shown above, a model's results are stored by applying `eststo` after the model has been fitted. This is also how official `estimates store` works. `eststo`, however, may also be used as a prefix command (see [U] **11.1.10 Prefix commands**). For example,

```
. eststo: regress price weight mpg
  (output omitted)
. eststo: regress price weight mpg foreign
  (output omitted)
. estout, style(fixed)
                      est1         est2
                         b            b
weight           1.746559     3.464706
mpg             -49.51222      21.8536
foreign                        3673.06
_cons            1946.069    -5853.696
```

`by` (see [D] **by**) is allowed with `eststo`, if `eststo` is used as a prefix command. Furthermore, `eststo` has an option to drop the `e(sample)` from the stored estimation sets to preserve memory.[1]

3 esttab: Tabulating estimates simplified

The new `esttab` command is a wrapper for `estout`. Its syntax is much simpler than that of `estout` and, by default, it produces publication-style tables that display nicely in Stata's Results window. `esttab` is more than just a simplified version of `estout`. On the one hand, `esttab` provides full `estout` functionality since all `estout` options are allowed in `esttab`. On the other hand, `esttab` also extends functionality. For example, `esttab` adds support for Word rich-text format (RTF) and Excel comma-separated value (CSV) files and provides improved functionality for LaTeX and HTML.

In what follows I will introduce `esttab` and show some of its applications, although I must be brief. See the *Appendix* for a syntax overview and consult the online help for more detailed information. You can find other examples at http://fmwww.bc.edu/repec/bocode/e/estout/.

Basic syntax and usage

`esttab`'s syntax is

`esttab` [*namelist*] [`using` *filename*] [`,` *options* *estout_options*]

where *namelist* is a list of names of stored estimation sets. *namelist* may be `*` to tabulate all stored estimation sets. If *namelist* is omitted, `esttab` tabulates the currently active estimates or, if present, the estimation sets stored by `eststo`. Specifying `using` causes the regression table to be written to a file on disk instead of being displayed in Stata's Results window.

1. Stored estimates consume a great deal of memory. To preserve full functionality of postestimation commands (see [U] **20 Estimation and postestimation commands**), an estimation sample indicator variable (i.e., a copy of the `e(sample)` function) is stored for each estimation set. Even though the `byte` storage type is used for these variables, they may greatly enlarge the dataset if it contains many observations or if many estimation sets are stored. Also, storing the `e(sample)` information has the side effect of slowing down cycling through the stored estimation sets in large datasets, which also slows down tabulation programs such as `estout` or official Stata's `estimates table`.

As with `estout` (or official Stata's `estimates table`), the basic procedure is to first store several models and then apply `esttab` to these stored estimation sets to compose a regression table. The main difference between `esttab` and `estout`, however, is that, if applied without options, `esttab` produces a fully formatted table. Consider the following example and compare it to the examples in section 2, which used `estout`:

```
. eststo clear
. sysuse auto
(1978 Automobile Data)
. eststo: regress price weight mpg
  (output omitted)
. eststo: regress price weight mpg foreign
  (output omitted)
. esttab
```

	(1) price	(2) price
weight	1.747**	3.465***
	(2.72)	(5.49)
mpg	-49.51	21.85
	(-0.57)	(0.29)
foreign		3673.1***
		(5.37)
_cons	1946.1	-5853.7
	(0.54)	(-1.73)
N	74	74

t statistics in parentheses
* p<0.05, ** p<0.01, *** p<0.001

Change table contents and add summary statistics

The default in `esttab` is to display raw point estimates along with t statistics and to print the number of observations in the table footer. Furthermore, the output displays asterisks denoting the statistical significance of coefficients. All this can be changed. To replace the t statistics with standard errors, add the adjusted R-squared, and omit the significance asterisks; for example, type

(*Continued on next page*)

```
. esttab, se ar2 nostar
```

	(1) price	(2) price
weight	1.747	3.465
	(0.641)	(0.631)
mpg	-49.51	21.85
	(86.16)	(74.22)
foreign		3673.1
		(684.0)
_cons	1946.1	-5853.7
	(3597.0)	(3377.0)
N	74	74
adj. R-sq	0.273	0.478

Standard errors in parentheses

The t statistics can also be replaced with p-values (`p`), confidence intervals (`ci`), or any parameter statistics contained in the estimates (see the `aux()` option). Further summary statistics options are, for example, `pr2` for the pseudo-R-squared and `bic` for Schwarz's Bayesian information criterion. Moreover, there is a generic `scalars()` option to include any other scalar statistics contained in the stored estimates. For instance, `scalars(F df_m df_r)` would add the overall F statistic and information on the degrees of freedom.

Also, the point estimates may be replaced with other statistics. Here is an example in which beta coefficients are printed and the t statistics are suppressed:

```
. esttab, beta not
```

	(1) price	(2) price
weight	0.460**	0.913***
mpg	-0.097	0.043
foreign		0.573***
N	74	74

Standardized beta coefficients
* p<0.05, ** p<0.01, *** p<0.001

More possibilities are provided by the `main()` option (replace the point estimates with any other stored parameter statistics) or by `estout` options such as `eform` or `margin`.

Layout, labels, and titles

There are many options for changing the table's design and adding labels, titles, or notes. For example, the `wide` option arranges point estimates and t statistics beside one another instead of beneath one another:

```
. esttab, wide
```

	(1) price		(2) price	
weight	1.747**	(2.72)	3.465***	(5.49)
mpg	-49.51	(-0.57)	21.85	(0.29)
foreign			3673.1***	(5.37)
_cons	1946.1	(0.54)	-5853.7	(-1.73)
N	74		74	

```
t statistics in parentheses
* p<0.05, ** p<0.01, *** p<0.001
```

Furthermore, the `plain` option produces a minimally formatted table with all display formats set to Stata's `%9.0g` quasistandard, and `compress` reduces horizontal spacing to fit more models on screen without line breaking. Other options are, for example, `label`, which causes variable labels to be used, and `mtitles()`, which specifies that model titles be printed in the header above the model columns. For example,

```
. esttab, se ar2 nostar brackets label title(This is a regression table)
> nonumbers mtitles("Model A" "Model B") addnote("Source: auto.dta")
This is a regression table
```

	Model A	Model B
Weight (lbs.)	1.747	3.465
	[0.641]	[0.631]
Mileage (mpg)	-49.51	21.85
	[86.16]	[74.22]
Car type		3673.1
		[684.0]
Constant	1946.1	-5853.7
	[3597.0]	[3377.0]
Observations	74	74
Adjusted R-squared	0.273	0.478

```
Standard errors in brackets
Source: auto.dta
```

Numerical formats

`esttab` has sensible default settings for numerical display formats. For example, t statistics are printed using two decimal places and R-squared measures are printed using three decimal places. For point estimates and, for example, standard errors, an adaptive display format is used where the number of displayed decimal places depends on the scale of the statistic to be printed (the default format is `a3`; see below).

You can change the format applied to a certain statistic by adding the appropriate display format specification in parentheses. For example, to display p-values and the R-squared with four decimal places and display point estimates with the `%9.0g` format, type

```
. esttab, b(%9.0g) p(4) r2(4) nostar wide
```

	(1) price		(2) price	
weight	1.746559	(0.0081)	3.464706	(0.0000)
mpg	-49.51222	(0.5673)	21.8536	(0.7693)
foreign			3673.06	(0.0000)
_cons	1946.069	(0.5902)	-5853.696	(0.0874)
N	74		74	
R-sq	0.2934		0.4996	

p-values in parentheses

Available formats are official Stata's display formats, such as `%9.0g` or `%8.2f` (see [D] **format**). Or, as shown above, you can request a fixed format by specifying an integer indicating the desired number of decimal places. Furthermore, you may specify an adaptive format, `a`#, where # in {1, . . . , 9} determines the minimum number of significant digits to be printed.

Document formats

Output format options in `esttab` allow the user to quickly switch between different document formats depending on the table's intended use. Available formats include the following:

`smcl` to produce a Stata markup and control language (SMCL)–formatted table. `smcl` is the default (unless `using` is specified) and is used to display the table in Stata's Results window.

`fixed` to produce a fixed-format ASCII table. This font is suitable, for example, if the table is to be displayed in a fixed-width font text editor.

`tab` to produce a tab-delimited ASCII table. This is a general format that can be used as an input format for many programs.

`csv` to produce a CSV-format table for use with Excel. The delimiter is a comma. See below for details on using `esttab` with Excel.

`scsv` to produce a CSV table using a semicolon as the delimiter. This format is appropriate for some non-English versions of Excel, such as the German version.

`rtf` to produce an RTF table for use with word processors. The code follows the guidelines in Burke (2003) and should work with almost any RTF viewer. See below for details on using `esttab` with Word.

`html` to produce a simple HTML-formatted table that can be displayed in a web browser.

`tex` to produce a table to be included in a LaTeX 2$_\varepsilon$ document. See below for details on using `esttab` with LaTeX.

`booktabs` to produce a LaTeX 2ε–formatted table for use with LaTeX's `booktabs` package.[2]

You can choose a specific document format by specifying the format's name as an option to `esttab`. For example,

```
. esttab, tab
  (output omitted)
```

produces a tab-delimited table. As indicated above, the `smcl` mode is the default. However, if `using` *filename* is specified, the default format depends on *filename*'s suffix (e.g., `rtf` for ".rtf", `html` for ".htm" or ".html"). Furthermore, if *filename* is specified without a suffix, a default suffix is added depending on the specified document format (e.g., ".tex" for `tex` or `booktabs`).

Use with Excel

To produce a table for use with Excel, apply the `csv` format (or the `scsv` format depending on the language version of Excel). For example, type

```
. esttab using example.csv
(output written to example.csv)
```

and then click on "example.csv" in Stata's Results window to launch Excel and display the file's content.

Depending on whether the `plain` option is specified, `esttab` uses two different variants of the CSV format. By default—that is, if `plain` is omitted—the contents of the table cells are enclosed in double quotes preceded by an equal sign (i.e., `="..."`). This convention prevents Excel from trying to interpret the contents of the cells and, therefore, preserves formatting elements such as parentheses around t statistics. One drawback of this approach is, however, that the displayed numbers cannot be used directly for further calculations in Excel. Hence, if the purpose of exporting the estimates is to do additional computations in Excel, specify the `plain` option. Here the table cells are enclosed in double quotes without the equal sign, and Excel will interpret the contents as numbers.

Use with Word

To produce a table for use with Word, apply the `rtf` format. For example, type

```
. esttab using example.rtf
(output written to example.rtf)
```

and then click on "example.rtf" in Stata's Results window to launch Word (or another RTF reader, depending on your operating system settings) and display the table. You may use `estout`'s `varwidth(#)` and `modelwidth(#)` options to change the column

2. See http://www.ctan.org/tex-archive/macros/latex/contrib/booktabs/.

widths in the RTF table (# = 12 corresponds to a column width of 1 inch, save cell padding). Furthermore, you may use the `append` option to include several tables in one RTF document.

Use with LaTeX

Using `esttab` together with LaTeX can be effective. For example, I produced table 1 in this article by running the command

```
. esttab using example1.tex, label nostar title(Regression table\label{tab1})
(output written to example1.tex)
```

and including

```
\input{example1.tex}
```

in the LaTeX code.

Table 1: Regression table

	(1) Price	(2) Price
Weight (lbs.)	1.747	3.465
	(2.72)	(5.49)
Mileage (mpg)	-49.51	21.85
	(-0.57)	(0.29)
Car type		3673.1
		(5.37)
Constant	1946.1	-5853.7
	(0.54)	(-1.73)
Observations	74	74

t statistics in parentheses

`esttab` automatically initializes the tabular environment and, if `title()` is specified, sets the table as a float object. Use the `fragment` option if you prefer to hard-code the table's environment and have `esttab` just produce the table rows.

Table 1 looks all right, but there is room for improvement. For example, the vertical spaces after the horizontal lines seem too small, the vertical gaps between the coefficients appear too large, and the use of double lines is debatable. One remedy for these problems is to load LaTeX's `booktabs` package in the document preamble and choose the `booktabs` format in `esttab`.

Other improvements to table 1 would be to use a typographical minus sign and to align the numbers on the decimal point. These improvements can be implemented, for

example, by loading LaTeX's dcolumn package[3] and using esttab's alignment() option to set a D column specifier. For instance, you might attain a good result by

```
. esttab using example2.tex, booktabs alignment(D{.}{.}{-1})
(output written to example2.tex)
```

Finally, spacing out the table columns to a certain total table width can be desirable. This is achieved using esttab's width() option. For example, type

```
. esttab using example3.tex, width(0.6\hsize)
(output written to example3.tex)
```

to set the table width to 60% of the width of the text body and add white space between the columns.

Viewing the internal estout call

Sometimes you cannot produce a desired table with standard esttab options right away. One approach is then to use esttab to assemble a table that comes close and then hand-edit and rerun the estout call that has been compiled by esttab. The estout call can be made visible by the noisily option and is returned in r(estout). For example,

```
. esttab, noisily
estout ,
 cells(b(fmt(a3) star) t(fmt(2) par("{ralign 12:{txt:(}" "{txt:)}}")))
 stats(N, fmt(%18.0g) labels(`"N"'))
 starlevels(* 0.05 ** 0.01 *** 0.001)
 varwidth(12)
 modelwidth(12)
 abbrev
 delimiter(" ")
 smcltags
 prehead(`"{hline @width}"')
 posthead("{hline @width}")
 prefoot("{hline @width}")
 postfoot(`"{hline @width}"' `"t statistics in parentheses"' @starlegend)
 varlabels(, end("" "") nolast)
 mlabels(, depvar)
 numbers
 collabels(, none)
 eqlabels(, begin("{hline @width}" "") nofirst)
 level(95)
  (output omitted)
```

3. See http://www.ctan.org/tex-archive/macros/latex/required/tools/.

4 Appendix

4.1 Syntax of eststo

`eststo`'s syntax is

[_]`eststo` [*name*] [`,` *options*] [`:` *command*]

[_]`eststo drop` { # | *name* } [{ # | *name* } ...]

[_]`eststo clear`

where *name* must not be `drop` or `clear` and *command* is any command returning its results in `e()` (see [U] **26 Overview of Stata estimation commands**). `by` is allowed if `eststo` is used as a prefix command (see [D] **by**). Below, I give a brief list of `eststo`'s options. See the online help for details.

options	description
[no]esample	do not/do store `e(sample)` (default is `esample` in `eststo` and `noesample` in `_eststo`)
title(*string*)	specify a title for the stored estimation set
addscalars(*name exp* [...] [, replace])	add scalars to the stored estimation set
refresh[(#)]	overwrite a previously stored estimation set
nocopy	clear `e()` after storing the estimation set
missing	use missing values in the `by` groups

4.2 Syntax of esttab

`esttab`'s syntax is

`esttab` [*namelist*] [`using` *filename*] [`,` *options*]

where *namelist* is either `_all` or *name* [*name* ...], and *name* is the name of a stored estimation set. The `*` and `?` wildcards may be used in *namelist*, and the results estimated last may be indicated by a period (`.`) even if they have not yet been stored. Below, I give a brief list of `esttab`'s options. See the online help for details.

options	description
Main	
`b(`*fmt*`)`	specify format for point estimates
`beta[(`*fmt*`)]`	display beta coefficients instead of point estimates
`main(`*name* `[`*fmt*`])`	display statistics contained in `e(`*name*`)` instead of point estimates
`t(`*fmt*`)`	specify format for t statistics
`abs`	use absolute value of t statistics
`not`	suppress t statistics
`se[(`*fmt*`)]`	display standard errors instead of t statistics
`p[(`*fmt*`)]`	display p-values instead of t statistics
`ci[(`*fmt*`)]`	display confidence intervals instead of t statistics
`aux(`*name* `[`*fmt*`])`	display statistics contained in `e(`*name*`)` instead of t statistics
`[no]cons`tant	do not/do report the intercept
Significance asterisks	
`[no]star[(`*sym* # `[...])]`	do not/do report significance asterisks and, optionally, redefine significance symbols and thresholds
`staraux`	attach asterisks to t statistics instead of point estimates
Summary statistics	
`r2[.]`, `ar2[.]`, `pr2[(`*fmt*`)]`	display raw, adjusted, or pseudo-R-squared
`aic[(`*fmt*`)]`, `bic[(`*fmt*`)]`	display Akaike's information criterion or Schwarz's Bayesian information criterion
`sca`lars`(`*scalarlist*`)`	display any other scalars contained in `e()`
`sfmt(`*fmtlist*`)`	set format(s) for *scalarlist*
`noobs`	do not display the number of observations
`obslast`	place the number of observations last
Layout	
`wide`	place point estimates and t statistics beside one another
`[no]parentheses`	do not/do print parentheses around t statistics
`br`ackets	use brackets instead of parentheses
`[no]gaps`	suppress/add vertical spacing
`[no]li`nes	suppress/add horizontal lines
`noeqlines`	suppress lines between equations
`compress`	reduce horizontal spacing
`plain`	produce a minimally formatted table

Labeling	
`label`	use variable labels
`title(`*string*`)`	specify a title for the table
`mtitles(`*strlist*`)`	specify model titles to appear in table header
`nomtitles`	disable model titles
`[no]depvars`	do not/do print dependent variables in header
`[no]numbers`	do not/do print model numbers in table header
`coeflabels(`*strlist*`)`	specify labels for coefficients
`[no]notes`	suppress/add notes in the table footer
`addnotes(`*strlist*`)`	add lines at the end of the table
Document format	
`smcl` | `fixed` | `tab` | `csv` | `scsv` | `rtf` | `html` | `tex` | `booktabs`	set the document format
`fragment`	suppress table opening and closing (LaTeX, HTML)
`page[(`*packages*`)]`	add page opening and closing (LaTeX, HTML)
`alignment(`*string*`)`	set alignment within columns (LaTeX, HTML, RTF)
`width(`*string*`)`	set width of table (LaTeX, HTML)
Output	
`replace`	overwrite an existing file
`append`	append the output to an existing file
`type`	force printing the table in the Results window
`noisily`	display the executed `estout` command
Advanced	
`drop(`*droplist*`)`	drop individual coefficients or equations
`keep(`*keeplist*`)`	keep individual coefficients or equations
`order(`*orderlist*`)`	change order of coefficients and equations
`equations(`*matchlist*`)`	match the models' equations
`eform`	report exponentiated coefficients
`margin`	report marginal effects or elasticities
`unstack`	place multiple equations in separate columns
other_estout_options	any other `estout` options[4]

4. All `estout` options are allowed in `esttab`. However, if specified, `estout` options supersede `esttab` options. For example, using `estout`'s `cells()` option will disable a whole series of `esttab` options (`b()`, `t()`, `abs`, `not`, `se()`, `p()`, `ci()`, `aux()`, `beta()`, `star`, `staraux`, `parentheses`, and `brackets`, to be precise). Furthermore, `estout`'s `stats()` option disables `r2()`, `ar2()`, `pr2()`, `aic()`, `bic()`, `scalars()`, `sfmt()`, `noobs`, and `obslast`. Other `estout` options that should be used with care are `begin()`, `delimiter()`, `end()`, `prehead()`, `posthead()`, `prefoot()`, `postfoot()`, `mlabels()`, and `varlabels()`.

4.3 Changes to estout

I have made many changes to `estout` since its first publication in Jann (2005). Some of the more important changes are

- `estout`'s `fmt()` suboption (within `cells()` and `stats()`) now provides two alternatives to official Stata's hard-to-type display formats. A fixed display format may now be specified as an integer indicating the number of decimal places to be displayed. Furthermore, formats may now also be specified as `a`#, where # is in $\{1, 2, \ldots, 9\}$. Doing so causes `estout` to choose a reasonable format for each number depending on its scale. # sets the minimum number of significant digits to be displayed (see `help estout, marker(fmt)` for details).

- `style(smcl)` now produces SMCL-formatted tables for display in Stata's Results window.

- The `*` and `?` wildcards are now allowed in coefficient and equation specifications within options such as `drop()` and `keep()`, and there is a new `order()` option to change the order of the coefficients in the table. In turn, `keep()` does not alter the order of the coefficients.

- The new `indicate()` option indicates for each model whether certain variables are present in the model. For example, if some of the models contain a set of year dummies, say, `y1`, `y2`, and `y3`, you may specify

  ```
  . estout ..., indicate(year effects = y1 y2 y3)
  ```

  to drop the dummies from the table and add a row indicating for each model whether the year dummies are part of it.

- The new `refcat()` option inserts information on the reference category of a categorical variable in the model.

- The new `transform()` option applies transformations to coefficients, standard errors, and confidence intervals. `transform()` is a generalization of the `eform` option and allows you to, for example, apply different kinds of transformations to different coefficients (say, apply exponentiation to the random-effects part of a `xtmixed` model but leave the rest unchanged).

- `estout` now takes action to clean up the table if equation names differ from model to model. In Stata 9, many commands return results by using multiple equations, which often disarranges the table. `estout` now automatically matches first equations if the equation names are different.

- Specifying `eqlabels(, none)` now causes `_cons` to be replaced by the equation name if `_cons` is the only parameter in an equation. Doing so is useful, for example, for tabulating `ologit` or `oprobit` results in Stata 9, which return the cut values as single equations containing just a constant.

4.4 Revision of estadd

`estadd`, also introduced in Jann (2005), has a new and simplified syntax, and its functionality has been extended. The syntax, now similar to official Stata's `estat` command (available since Stata 9), is

`estadd` *subcommand* [`,` *options*] [`:` *namelist*]

where *namelist* consists of names of stored estimation sets. If *namelist* is empty, `estadd` is applied to the last (i.e., currently active) estimates. The options are

options	description
`replace`	permit `estadd` to overwrite existing `e()` results
`prefix(`*string*`)`	specify a common prefix for names of added results
subcmdopts	specific options associated with *subcommand* (see the online documentation)

`estadd` has three kinds of subcommands. First, there are elementary functions to add a simple macro, scalar, or matrix to the `e()` returns. For example, use the `scalar` subcommand to add results from `test`:

```
. regress price weight mpg
  (output omitted)
. test weight = mpg
  (output omitted)
. estadd scalar p_diff = r(p)
```

The second type comprises subcommands that compute and add auxiliary statistics for each coefficient in the model. For example, the `beta` subcommand adds a vector containing standardized beta coefficients, and the `mean` subcommand adds a vector containing the means of the regressors. Once added, these statistics can be tabulated in `estout` by using the `cells()` option. For example,

```
. regress price weight mpg
  (output omitted)
. estadd mean
. estout, cells("b mean") style(fixed)
                          b         mean
weight            1.746559     3019.459
mpg              -49.51222      21.2973
_cons             1946.069
```

The third type contains subcommands to compute and add scalar summary statistics that can then be tabulated in `estout` by using the `stats()` option.

Below, I give a brief list of the available subcommands. See the online help for details.

subcommands	description
Elementary	
`local` *name* `...`	add a macro
`scalar` *name* `=` *exp*	add a scalar
`matrix` *name* `=` *mat* [`, copy`]	add a matrix
Statistics for each coefficient	
`beta`	standardized coefficients
`vif` [`, tolerance sqrvif`]	variance inflation factors (after `regress`)
`pcorr` [`, semi`]	partial (and semipartial) correlations
`expb` [`, noconstant`]	exponentiated coefficients
`ebsd`	standardized factor change coefficients
`mean`	means of regressors
`sd` [`, nobinary`]	standard deviations of regressors
`summ` [, *stats*]	various descriptives of the regressors; the default *stats* are `mean`, `sd`, `min`, and `max`; further *stats* are `sum`, `range`, `var`, `cv`, `semean`, `skewness`, `kurtosis`, `p1`, `p5`, `p10`, `p25`, `p50`, `p75`, `p90`, `p95`, `p99`, `iqr`, `all`, `median`, and `q`
Summary statistics	
`coxsnell`	Cox and Snell's pseudo-R-squared
`nagelkerke`	Nagelkerke's pseudo-R-squared
`lrtest` *model* [, *options*]	likelihood-ratio test; *options* are `name(`*string*`)` and *lrtest_options*
`ysumm` [, *stats*]	descriptives of the dependent variable; *stats* are as for the `summ` subcommand

5 Acknowledgments

`esttab` and `estout` owe much to John Luke Gallup's `outreg` (Gallup 1998) and official Stata's `estimates table` (see [R] **estimates**). Furthermore, Roy Wada's `outreg2` command motivated the idea to provide the adaptive display format.

Many people commented on `estout` and reported bugs. Christopher F. Baum, Debra Hevenstone, and J. Scott Long made comments on this article. I thank them all.

6 References

Burke, S. M. 2003. *RTF Pocket Guide*. Beijing: O'Reilly Media.

Gallup, J. L. 1998. sg97: Formatting regression output for published tables. *Stata Technical Bulletin* 46: 28–30. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 200–202. College Station, TX: Stata Press.

Jann, B. 2005. Making regression tables from stored estimates. *Stata Journal* 5: 288–308.

About the author

Ben Jann is research assistant and a Ph.D. candidate at the Department of Humanities, Social, and Political Sciences of the ETH Zurich in Switzerland.