The Stata Journal

The *Stata Journal* publishes reviewed papers together with shorter notes or comments, regular columns, book reviews, and other material of interest to Stata users. Examples of the types of papers include 1) expository papers that link the use of Stata commands or programs to associated principles, such as those that will serve as tutorials for users first encountering a new field of statistics or a major new technique; 2) papers that go "beyond the Stata manual" in explaining key features or uses of Stata that are of interest to intermediate or advanced users of Stata; 3) papers that discuss new commands or Stata programs of interest either to a wide spectrum of users (e.g., in data management or graphics) or to some large segment of Stata users (e.g., in survey statistics, survival analysis, panel analysis, or limited dependent variable modeling); 4) papers analyzing the statistical properties of new or existing estimators and tests in Stata; 5) papers that could be of interest or usefulness to researchers, especially in fields that are of practical importance but are not often included in texts or other journals, such as the use of Stata in managing datasets, especially large datasets, with advice from hard-won experience; and 6) papers of interest to those who teach, including Stata with topics such as extended examples of techniques and interpretation of results, simulations of statistical concepts, and overviews of subject areas.

The *Stata Journal* is indexed and abstracted by *CompuMath Citation Index*, *Current Contents/Social and Behavioral Sciences*, *RePEc: Research Papers in Economics*, *Science Citation Index Expanded* (also known as *SciSearch*, *Scopus*, and *Social Sciences Citation Index*.

For more information on the *Stata Journal*, including information for authors, see the webpage

http://www.stata-journal.com

Subscriptions are available from StataCorp, 4905 Lakeway Drive, College Station, Texas 77845, telephone 979-696-4600 or 800-STATA-PC, fax 979-696-4601, or online at

http://www.stata.com/bookstore/sj.html

Subscription rates listed below include both a printed and an electronic copy unless otherwise mentioned.

U.S. and Canada		Elsewhere	
1-year subscription	$ 79	1-year subscription	$115
2-year subscription	$155	2-year subscription	$225
3-year subscription	$225	3-year subscription	$329
3-year subscription (electronic only)	$210	3-year subscription (electronic only)	$210
1-year student subscription	$ 48	1-year student subscription	$ 79
1-year university library subscription	$ 99	1-year university library subscription	$135
2-year university library subscription	$195	2-year university library subscription	$265
3-year university library subscription	$289	3-year university library subscription	$395
1-year institutional subscription	$225	1-year institutional subscription	$259
2-year institutional subscription	$445	2-year institutional subscription	$510
3-year institutional subscription	$650	3-year institutional subscription	$750

Back issues of the *Stata Journal* may be ordered online at

http://www.stata.com/bookstore/sjj.html

Individual articles three or more years old may be accessed online without charge. More recent articles may be ordered online.

http://www.stata-journal.com/archives.html

The *Stata Journal* is published quarterly by the Stata Press, College Station, Texas, USA.

Address changes should be sent to the *Stata Journal*, StataCorp, 4905 Lakeway Drive, College Station, TX 77845, USA, or emailed to sj@stata.com.

The Stata Journal (2012)
12, Number 4, pp. 655–673

A programmer's command to build formatted statistical tables

John Luke Gallup
Portland State University
Portland, OR
jlgallup@pdx.edu

Abstract. The `frmttable` command is a tool for experienced users and programmers to create formatted tables from statistics and write them to Word or LaTeX files. My objective is to provide as much control over the layout and formatting of the statistical tables as possible in both file formats while keeping the syntax simple. Users can create rectangular tables with any configuration of data and text; specify numeric formats, font sizes, and font types at the table cell level; specify row spacing; and place lines in or around the table. A complex table can be built by merging or appending new statistics to an existing table, and multiple tables can be included in the same document, making it possible to create a fully formatted statistical appendix from a single do-file. In this article, I provide examples of the ways in which programmers call `frmttable` to create formatted tables of statistics.

Keywords: sg97_5, frmttable, outreg, tables, statistics, formatting, Word, TeX, programming command

1 Introduction

My goal in writing `frmttable` is to provide a tool to create fully formatted statistical tables simply and easily but with almost complete control of the format of a table written to a word processor file. `frmttable` can create a whole appendix of formatted tables in a Word or LaTeX document from a single do-file that would not require any modification after it was created.

`frmttable` makes available these formatting capabilities to experienced users who can generate a matrix of the statistics they want to put in a formatted table. The command is also useful for programmers who write programs generating statistics and want the results written to a formatted table.[1] `frmttable` is distributed as part of the `outreg` (Gallup 2012) program.

`frmttable` carries out four main tasks:

- takes a matrix of statistics, combines it with titles, headings, and other additions, and creates a Mata `struct` of string matrices holding the table contents

1. If the programmer creates an e-class estimation command that stores results in the `e(b)` and the `e(V)` matrices, then `outreg` (based on `frmttable`) can already write the results to a formatted table; it is not necessary to use `frmttable` directly.

- displays the table in the Stata Results window
- writes the table, using extensive formatting options, to a Word or a TEX[2] document file
- merges or appends the table with subsequent statistics to create a more complex table

A programmer who writes an ado-file to generate statistics can call the `frmttable` command to create a fully formatted table out of the statistics. The programmer passes the statistics as a Stata matrix to `frmttable`. The user of the program can then apply the dozens of `frmttable` options to modify the format (numerical, font, etc.) and layout of the table similar to the way the layout of a graph can be modified in Stata.

The matrix of statistics passed to `frmttable` can be reorganized in several ways by using options. First, missing values in the matrix produce blank cells in the formatted table. Row and column names of the Stata matrix can be used as table row and column headings in the `frmttable` table; if the names are variable names, their variable labels can be substituted with the `varlabels` option.

Second, when the programmer's code generates "substatistics" (such as t statistics to be placed below regression coefficients), these statistics can be passed as separate columns in the statistics matrix; with the `substat()` option, they will be positioned below the primary statistics in the formatted table. By default, the first substatistic is placed in parentheses.

Third, `frmttable` conveniently handles "double statistics" such as the lower and upper bounds of confidence intervals or minimum–maximum ranges. The lower and upper bounds of double statistics are passed to `frmttable` in separate columns of the statistics matrix. With the `doubles()` option, the lower and upper statistics will be combined into a single range in the formatted table.

Fourth, a statistics matrix organized by equation can be split into separate columns of the formatted table with the `eq_merge` option.

Once a formatted table is created by the `frmttable` routine, it remains persistent in memory (until replaced or until the end of the Stata session) similar to estimation results after Stata estimation commands. The table is available for merging or appending with subsequent results through additional `frmttable` calls, making it possible to construct quite complex tables by parts. The commands for merging and appending results accept table names so that users can juggle more than one formatted table simultaneously.

2. From here on, I will refer to LATEX documents simply as TEX documents.

The formatted table information is held in a Mata `struct` (see [M-2] **struct**) of matrices. The body of the table is held in a Mata string matrix, with separate string matrices for the title, the notes below the table, and regular text before and after the table. The efficient memory structure of Mata matrices makes them suitable for storing very large tables. The `frmttable struct` also contains several vectors designating the internal dimensions of sections within the body of the table (such as the number of header rows above the statistics).

`frmttable` executes very fast because almost all the code is in the Mata language, which is faster and more concise than the Stata macro language.

`frmttable` has a large number of options to enable users to make fine adjustments to all aspects of the layout and formatting (analogous, on a smaller scale, to the plethora of `graph` options in Stata). Because `frmttable`'s options are numerous (59 in total), the detailed description of the options is left to the `frmttable` online help file.

In this article, I present a series of examples that demonstrate how to integrate `frmttable` into user-written programs. I emphasize the programming aspects rather than how to use the many options that modify the format of the tables.

`frmttable` is the formatting engine responsible for most of the capabilities of `outreg`, a program that formats estimation tables. Most formatting options for `frmttable` have already been described in the *Stata Journal* article about `outreg` (Gallup 2012), which inherits these capabilities from `frmttable`. Many examples of using `frmttable` formatting options can also be found in the `outreg` online help file.

1.1 Related commands

Stata displays statistics in the Results window. Stata does not have any commands to write tables of statistics to Word or TEX files. User-written commands have filled the void. Estimation tables can be written to formatted Word and TEX tables with `outreg` (which uses `frmttable` for formatting), `outreg2` (Wada 2010), or `estout` (Jann 2005, 2007). The `xml_tab` program (Lokshin and Sajaia 2008) writes estimation tables to a formatted Excel file. Summary statistics can be written to TEX or plain text files by `tabout` (Watson 2011), `outreg2`, and `estpost` (distributed with the `estout` program).

None of these commands do exactly what `frmttable` does: write a Stata matrix of statistics to a fully formatted Word or TEX table. All the above user-written commands take output from specific Stata commands (especially regressions) rather than from an arbitrary matrix. None of the above commands can control the appearance of the tables to the same degree as `frmttable` (except for `outreg`, which uses `frmttable`), nor can they build up a complex table by merging and appending simpler pieces of the table.

The most similar command to `frmttable` is the user-written `outtable` (Baum and Azevedo 2001), which writes a Stata matrix to a TEX table. However, it has much more limited formatting capabilities and cannot merge or append to a previously created table.

2 Syntax and description

`frmttable` [`using` *filename*] [`,` *options*]

Specifying the optional `using` *filename* statement ensures that the `frmttable` table is written to a document file instead of just being displayed in the Results window. `using` *filename* is left out of the `frmttable` statement mainly when the user will merge additional statistics before the finished table needs to be written to a document.

`frmttable using` *filename* creates a Microsoft Word file[3] by default or a TeX file with the `tex` option. The table is also displayed in the Results window (minus some of the finer formatting destined for the Word or TeX file) unless the `nodisplay` option is specified.

`frmttable` normally has a `statmat()` option specifying a Stata matrix containing the statistics that make up the body of the table. The main alternative is the `replay` option, which can redisplay and write to a file a formatted table previously created by `frmttable`.

The syntax of the many `frmttable` options is not included in this article because of length. The syntax can be found in the online help. After a technical note about the memory structure of the `frmttable` table, the rest of the article provides examples of the most important ways to integrate `frmttable` into a user-written program.

❏ Technical note

This note explains how to access `frmttable` tables directly in memory. `frmttable` already has several options for manipulating tables in memory (notably, the `replay`, `merge`, `append`, `store()`, and `clear` options). Accessing `frmttable` tables with these options does not require understanding the memory structure explained here. This note explains the structure and naming conventions of `frmttable` tables for those programmers who nonetheless want to access the tables directly in memory, outside of `frmttable`.

`frmttable` creates a `struct` of Mata matrices. The default variable name of this `struct` is "_FrmtT". Hence, a pointer to this `struct` (if it exists in memory) can be obtained with the command

```
mata: pFrmtT = findexternal("_FrmtT")
```

If the `frmttable` table has been given a *tblname*, this name is added to the default name of "_FrmtT". So for a *tblname* of "`tbl1`" (for example, using the option `store(tbl1)`), the `struct` will have the name "_FrmtTtbl1".

3. The Word file format is rich text format, an ASCII file format readable by any software that can open Word files.

The struct of frmttable tables has the following components:

```
struct FrmtTabl {                 // structure holding table data
   string colvector Pretext      // plain text preceding the table
   string colvector Title        // table title
   string matrix Body            // table body including column headings and
                                 // row titles
   real rowvector SectRows       // # of rows in each row section of Body
   real rowvector SectSubstats   // # of substats in row sections
   real scalar ConsSect          // has a row section for "_cons" statistics
   real rowvector SectCols       // # of columns in each column section of Body
   string colvector Note         // note text at bottom of table
   string colvector Posttext     // plain text that comes below table
}
```

A few of the FrmtTabl components need explanation:

SectRows keeps track of the different sections of Body to enable merging of complex tables, particularly estimation results from using the outreg command. The table has designated row sections to ensure that rows with each section are merged appropriately. For example, in the case of estimation tables, outreg designates separate row sections for the regular coefficients and the constant coefficients. This ensures that rows for new variables from subsequently merged estimation results will be inserted above the constant coefficient estimates.

The number of elements in SectRows indicates how many separate row sections are in the table. The first element of SectRows always specifies the number of column header rows (possibly zero). The second element is the number of primary statistics rows. The third element in tables created by outreg is the number of rows of _cons coefficient estimates. The fourth element in tables created by outreg indicates the number of rows devoted to summary statistics such as the number of observations.

SectSubstats indicates the number of substatistics (from the substat() option) in each of the row sections. For instance, tables created by outreg have by default one substatistic for the second and third row sections (the t statistics for the regular and constant coefficients), but no substatistics for the summary statistics in the fourth row section. The information in SectSubstats enables correct merging of a new set of results that have a different number of substatistics than the original statistics.

The scalar ConsSect is a Boolean variable indicating whether there is a separate row section for _cons coefficient estimates, which are handled specially by merge.

SectCols has only two elements: the first is the number of row title columns (possibly zero) and the second is the number of statistics columns.

❏

3 Examples of frmttable in use

This section contains a set of examples applying frmttable in a programming context to create formatted tables from previously created statistics.

Before we look at the example of the basic usage of `frmttable`, here is a list of the coming examples with the `frmttable` options each example uses.

3.1 Basic usage

`statmat()` names the Stata matrix containing the statistics that make up the body of the table.

`tex` specifies that a TeX rather than a Word file be written.

`sdec()` specifies the decimal places for the statistics in `statmat()`.

`title()` places a title or titles above the table.

`ctitles()` places column titles above the statistics in the table.

`rtitles()` places row titles in the leftmost column (or columns) of the table.

3.2 Merge and append

`merge` merges statistics to an existing table as new columns.

`append` appends statistics below an existing table as new rows.

`addrows()` adds new rows of text to the bottom of the table.

3.3 Multicolumn titles, border lines, fonts

`multicol()` combines table cells into a single cell that spans multiple columns.

`vlines()` draws vertical lines between columns.

`vlstyle()` changes the style of vertical lines.

`basefont()` changes the base font style and size for all text in the table.

3.4 Add stars for significance: Substatistics and annotate

`substat()` indicates the number of substatistics to be placed in separate rows below the principal statistics (for example, t statistics below means).

`annotate()` passes a matrix of annotation locations.

`asymbol()` provides symbols for each annotation location in `annotate()`.

`varlabels` replaces variable names with variable labels in row and column titles.

3.5 Make a table of summary statistics and merge it with a regression table

Uses the previously described options `statmat()`, `substat()`, `varlabels`, `merge`, and `ctitles()`.

3.6 Create complex tables using merge( ) and append( )

`clear(tbl1)` removes the table named `tbl1` from memory.

`merge(tbl2)` merges statistics from table `tbl2` (when used with `replay`).

`append(tbl3)` appends statistics from table `tbl3` as new rows below the existing statistics (when used with `replay`).

`replay(tbl1)` reuses an existing table named `tbl1`.

3.7 Double statistics

`doubles()` indicates which statistics in `statmat()` are double statistics.

3.1 Basic usage

The basic role of `frmttable` is to take statistics in a Stata matrix and organize them in a table that is displayed in the Results window and can be written to a file as a Word table or a TeX table.

First, we create a 2 × 2 Stata matrix named **A**:

```
. matrix A = (100,50\0,50)
. matrix list A
A[2,2]
     c1   c2
r1  100   50
r2    0   50
```

The simplest usage of the `frmttable` command is to display the matrix **A**:

```
. frmttable, statmat(A)
                                   ─────────────────
                                    100.00  50.00
                                     0.00   50.00
                                   ─────────────────
```

This does not get us very far. The reason `frmttable` is useful is that it can make extensive adjustments to the formatting of the table and write the result to a Word or TeX document.

The `frmttable` command below has a `using` statement followed by a filename (`xmpl1`). This causes the table to be written to a Word document called `xmpl1.doc`. Word documents are the default; to specify that the table be written as a TeX document, include the `tex` option.

The `frmttable` statement below adds a number of options. The first, `sdec()`, sets the number of decimal places displayed for the statistics in `statmat()` to 0. The next three options, `title()`, `ctitle()`, and `rtitle()`, add an overall title to the table, titles above each column of the table, and titles on the left of each row of the table, respectively. The column and row titles are designated with the syntax used for matrices: commas separate columns and backslashes separate rows.

```
. frmttable using xmpl1, statmat(A) sdec(0) title("Payoffs")
> ctitle("","Game 1","Game 2") rtitle("Player 1"\"Player 2")
                           Payoffs
                    ----------------------------
                              Game 1  Game 2
                    ----------------------------
                     Player 1   100     50
                     Player 2    0      50
                    ----------------------------
```

The table in the Word document `xmpl1.doc` looks like this:

Payoffs

	Game 1	Game 2
Player 1	100	50
Player 2	0	50

3.2 Merge and append

Once `frmttable` is run, the table created stays in memory (as a `struct` of Mata string matrices). Subsequent statistical results can be `merge`d as new columns of the table or `append`ed as new rows. The merged columns are arranged so that the new row titles are matched with the existing table's row titles, and rows with unmatched titles are placed below the other statistics (similar to the way the Stata `merge` command matches observations of the merged dataset).

The `frmttable` command below merges a new column of statistics for players 1 and 3 to the existing `frmttable` table, created in the previous example.

```
. matrix B = (25\75)
. frmttable, statmat(B) sdec(0)
> ctitle("","Game 3") rtitle("Player 1"\"Player 3") merge
                           Payoffs
                ------------------------------------
                          Game 1  Game 2  Game 3
                ------------------------------------
                 Player 1   100     50      25
                 Player 2    0      50
                 Player 3                   75
                ------------------------------------
```

In this case, the new statistics in matrix **B** are arranged according to the row titles in the `rtitles()` option. The statistics in the first row of **B** for player 1 are lined up with the statistics for `Player 1` in the existing table; the statistics for player 3 are placed below those for player 2 because `Player 3` is a new row title.

The text of the row titles must match exactly for the merged results to be placed in the same row. Row titles of `Player 1` and `player 1` do not match, so they would be placed in different rows.

Next we add another column to the table for new game results.

```
. matrix C = (90\10)
. frmttable, statmat(C) sdec(0)
> ctitle("","Game 4") rtitle("Player 2"\"Player 4") merge
                          Payoffs
          ----------------------------------------
                     Game 1  Game 2  Game 3  Game 4
          ----------------------------------------
           Player 1   100      50      25
           Player 2    0       50              90
           Player 3                    75
           Player 4                            10
          ----------------------------------------
```

The statistics for player 2 and for player 4 are merged: the statistics for player 2 are lined up with previous results for `Player 2`, and the statistics for the new row title, `Player 4`, are placed below the other rows.

Finally, we `append` new rows to the table for the total payoffs.

```
. matrix D = (100,100,100,100)
. frmttable, statmat(D) sdec(0) rtitle("Total") append
                          Payoffs
          ----------------------------------------
                     Game 1  Game 2  Game 3  Game 4
          ----------------------------------------
           Player 1   100      50      25
           Player 2    0       50              90
           Player 3                    75
           Player 4                            10
           Total      100     100     100     100
          ----------------------------------------
```

Whereas the `merge` option creates new columns, the `append` option creates new rows, placing the new statistics below the existing table. If matrix **D** had more than or fewer than four columns, it would still be appended below the existing results but with a warning message. The arrangement of the `append`ed results does not depend on the column titles (unlike the way `merge` depends on the row titles). In fact, the `ctitles()` of the appended data are ignored if they are specified.

An alternative way of adding rows and columns is with the options `addrows()` and `addcols()`. `merge` and `append` add matrices of numbers to a previously created table; `addrows()` and `addcols()` add on rows and columns of text (which can include numbers) to the table currently being created.

The following set of commands will create the same table as above, but we use the `addrows()` option to attach the column totals as text instead of `append`ing a Stata matrix:

```
. matrix E = (100,50,25,. \ 0,50,.,90 \ .,.,75,. \ .,.,.,10)
. frmttable, statmat(E) sdec(0) addrows("Total", "100", "100", "100", "100")
> rtitles("Player 1" \ "Player 2" \ "Player 3" \ "Player 4")
> ctitles("", "Game 1", "Game 2", "Game 3", "Game 4")
> title("Payoffs")
                             Payoffs
                  ----------------------------------------------
                             Game 1  Game 2  Game 3  Game 4
                  ----------------------------------------------
                   Player 1   100      50      25
                   Player 2    0       50              90
                   Player 3                    75
                   Player 4                            10
                   Total      100     100     100     100
                  ----------------------------------------------
```

3.3 Multicolumn titles, border lines, fonts

Many formatting options are available in `frmttable`. This example makes some of the column titles span multiple columns, places a vertical line in the table, and changes the font size and typeface.

This gives just a taste of the many user options in `frmttable` because this article is focused on its use by programmers. `frmttable` can change many other aspects of the tables it creates, such as footnotes and other annotations among the statistics, justification of columns, and spacing above and below table cells (as detailed in the online help). You can find additional examples of many `frmttable` formatting options in the online help for the `outreg` command: example 11 focuses on fonts, example 12 on special characters (foreign or mathematical), example 13 on multiple tables in the same document, and example 14 on placing footnotes among the statistics.

In the Stata code below, the `frmttable` table is created from data in the matrix **F**. Where **F** contains missing values, the table cells will be blank.

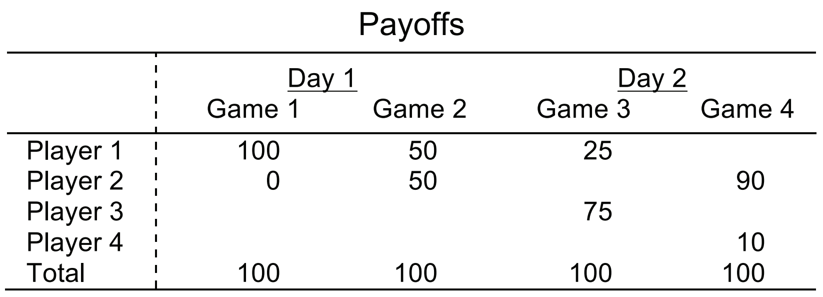
The table's column titles in the `ctitles()` option have two rows, and the titles in the first row are meant to span two columns each. They are made to span multiple columns with the `multicol(1,2,2;1,4,2)` option. The two triples of numbers—`1,2,2` and `1,4,2`—indicate which table cells span more than one column. `1,2,2` indicates that the first row, second column of the table should span two columns; `1,4,2` indicates that the first row, fourth column of the table should span two columns.

A dashed vertical line is placed in the table separating the row titles from the statistics. The `vlines(010)` option specifies where the vertical line or lines are placed. A `0` indicates no line, and a `1` indicates a line. The `010` means no line to the left of the first column, a vertical line between the first and second column, and no line between the second and third column. Because the table has more than two columns, the "no line" specification is extended to the rest of the columns. The `vlstyle(a)` option changes the line style from the default solid line to a dashed line.

The last option, `basefont(arial fs10)`, changes the font of the Word table to Arial, with a base font size of 10 points. The base font size applies to most of the table, but by default, the table's title has larger text and the notes below the table, if any, have smaller text.

```
. matrix F = (100,50,25,. \ 0,50,.,90 \ .,.,75,. \ .,.,.,10 \ 100,100,100,100)
. frmttable using xmpl2, statmat(F) sdec(0) title("Payoffs") replace
> ctitles("", "{\ul Day 1}", "", "{\ul Day 2}" ,"" \
> "", "Game 1", "Game 2", "Game 3", "Game 4")
> multicol(1,2,2;1,4,2)
> rtitles("Player 1" \ "Player 2" \ "Player 3" \ "Player 4" \ "Total")
> vlines(010) vlstyle(a) basefont(arial fs10)
(note: file xmpl2.doc not found)
                                Payoffs
                  ------------------------------------------------------
                             {\ul Day 1}          {\ul Day 2}
                               Game 1     Game 2   Game 3     Game 4
                  ------------------------------------------------------
                   Player 1     100         50       25
                   Player 2      0          50                  90
                   Player 3                          75
                   Player 4                                     10
                   Total        100        100      100        100
                  ------------------------------------------------------
```

The resulting table appears in Word like this:



Payoffs

	Day 1		Day 2	
	Game 1	Game 2	Game 3	Game 4
Player 1	100	50	25	
Player 2	0	50		90
Player 3			75	
Player 4				10
Total	100	100	100	100

3.4 Add stars for significance to regression output: Substatistics and annotate

The following Stata commands create a matrix, `b_se`, containing regression coefficients in the first column and standard errors of estimates in the second column:

```
. sysuse auto, clear
(1978 Automobile Data)
. regress mpg length weight headroom
  (output omitted)
. matrix b_se = get(_b)´, vecdiag(cholesky(diag(vecdiag(get(VCE)))))´
. matrix colnames b_se = mpg mpg_se
. matrix list b_se
b_se[4,2]
                mpg      mpg_se
  length  -.07849725   .05699153
  weight  -.00385412   .00159743
headroom  -.05143046   .55543717
   _cons   47.840789   6.1492834
```

`frmttable` will convert this matrix into a formatted table. The `substat(1)` option informs `frmttable` that the second column of statistics, the standard errors, should be interweaved below the first column of statistics, the coefficients, in the table. If the option were `substat(2)`, the second and third columns of statistics would be interweaved below the statistics in the first column of `statmat()`.

In the absence of `rtitles()` and `ctitles()`, `frmttable` uses the matrix row names and column names of `b_se` as the row and column titles for the table.

```
. frmttable, statmat(b_se) substat(1) sdec(3)
```

	mpg
length	-0.078
	(0.057)
weight	-0.004
	(0.002)
headroom	-0.051
	(0.555)
_cons	47.841
	(6.149)

Stars indicating significance levels can be placed next to the standard errors by using the `annotate()` option. First, you need to create a Stata matrix indicating the cells to which the stars should be added. The matrix, named `stars` below, has a 1 in the second row, second column, and a 2 in the fourth row, second column, because the second and fourth coefficients are statistically significant at the stated levels.

```
. local bc = rowsof(b_se)
. matrix stars = J(`bc´,2,0)
. forvalues k = 1/`bc´ {
  2.          matrix stars[`k´,2] =
>                  (abs(b_se[`k´,1]/b_se[`k´,2]) > invttail(`e(df_r)´,0.05/2)) +
>                  (abs(b_se[`k´,1]/b_se[`k´,2]) > invttail(`e(df_r)´,0.01/2))
  3. }
. matrix list stars
stars[4,2]
    c1  c2
r1   0   0
r2   0   1
r3   0   0
r4   0   2
```

The entries of 1 and 2 in the `stars` matrix correspond to the first and second entry of the `asymbol(*,**)` option, which adds a single star in the cell where the 1 is and a double star in the cell where the 2 is. All the elements of `stars` equal to 0 will have no symbols added. The dimensions of `stars` (4×2) correspond to the dimensions of the `statmat()` matrix, not the dimensions of the statistics in the final table (8×1), which has a single statistics column because of the `substat(1)` option.

The option `varlabels` causes the variable labels for the variables `mpg`, `length`, `weight`, and `headroom` to be substituted for their names.

```
. frmttable, statmat(b_se) substat(1) sdec(3)
> annotate(stars) asymbol(*,**) varlabels
                          ──────────────────────────────────

                                    Mileage (mpg)
                          ──────────────────────────────────
                           Length (in.)        -0.078
                                               (0.057)
                           Weight (lbs.)       -0.004
                                               (0.002)*
                           Headroom (in.)      -0.051
                                               (0.555)
                           Constant            47.841
                                               (6.149)**
                          ──────────────────────────────────
```

The code above implements the most basic capabilities of `outreg`, so the same table can more easily be created by

```
. outreg, se varlabels
```

3.5 Make a table of summary statistics and merge it with a regression table

First, we create a Stata matrix containing summary statistics for four variables: `length`, `weight`, `headroom`, and `mpg`. The first column of the matrix `mean_sd` contains the means of the variables, and the second column contains the standard deviations. The statistics are calculated by the `summarize` command, looping over the variables through the `foreach` command.

```
. matrix mean_sd = J(4,2,.)
. local i = 1
. foreach v in length weight headroom mpg {
  2.         summ `v´
  3.         matrix mean_sd[`i´,1] = r(mean)
  4.         matrix mean_sd[`i´,2] = r(sd)
  5.         local i = `i´ + 1
  6. }
  (output omitted)
. matrix rownames mean_sd = length weight headroom mpg
. matrix list mean_sd
mean_sd[4,2]
                 c1          c2
  length  187.93243   22.26634
  weight  3019.4595  777.19357
headroom  2.9932432  .84599477
     mpg  21.297297  5.7855032
```

We can create a formatted table with this matrix of statistics, and we can also merge these statistics into any other table created by `frmttable` (or by commands that call `frmttable`, such as `outreg`). The command below `merge`s the summary statistics with the table of regression coefficients created in the previous example. Note that `merge` can line up the substatistics properly even though they do not have row titles. The merged rows of the substatistics are aligned according to the row titles of the primary statistics above them.

```
. frmttable, statmat(mean_sd) substat(1) varlabels
> ctitles("", Summary statistics) merge
(note: tables being merged have different numbers of row sections)
              ─────────────────────────────────────────────────────
                                 Mileage (mpg)  Summary statistics
              ─────────────────────────────────────────────────────
               Length (in.)       -0.078            187.93
                                  (0.057)          (22.27)
               Weight (lbs.)      -0.004           3,019.46
                                  (0.002)*         (777.19)
               Headroom (in.)     -0.051             2.99
                                  (0.555)           (0.85)
               Mileage (mpg)                        21.30
                                                    (5.79)
               Constant           47.841
                                  (6.149)**
               R2                  0.66
               N                    74
              ─────────────────────────────────────────────────────
                                 * p<0.05; ** p<0.01
```

This example demonstrates how `frmttable` works, but in the real world, a user can more easily create a table like this with the following commands:

```
. regress mpg length weight headroom
. outreg, se varlabels
. mean length weight headroom mpg
. outreg, se varlabels merge
```

3.6 Create complex tables using merge( ) and append( )

If you use `frmttable` to handle the output from a new command you write, users of your command can build quite complex tables by repeatedly executing your command along with `frmttable`'s `merge()` and `append()` options. In this example, we use `outreg` as an instance of a `frmttable`-based command to build a table from parts.

Let's say we want to create a table showing how a car's weight affects its mileage. We want the table to be broken down by foreign versus domestic cars and by three categories of headroom. It is easy with a `frmttable`-based command such as `outreg` to create six separate tables with all of these results. It is also easy to create a single table with six columns (or six rows) of coefficients by using straightforward application of the `merge()` (or `append()`) option.

It is more complicated to create a table with three columns of foreign estimates above three columns of domestic estimates. The results for foreign cars must be merged across the headroom categories into one table, and the results for the domestic cars must be merged into a separate table. The two tables must then be appended one below the other. Working with two separate `frmttable` tables simultaneously requires the use of table names.

The statistics for the table in this example are created in a double `foreach` loop, iterating first over foreign versus domestic and then over three categories of headroom. Using data from `auto.dta`, we first recode the variable `headroom` into a new variable, `hroom`, with just three levels.

```
. sysuse auto, clear
(1978 Automobile Data)

. recode headroom (1.5=2) (3.5/5=3), gen(hroom)
(34 differences between headroom and hroom)
```

Type in the following `outreg, clear()` commands; I explain them in just a moment:

```
. outreg, clear(row0)

. outreg, clear(row1)
```

The heart of the table building occurs with a double `foreach` loop, iterating over `foreign` and `hroom` values. We estimate the correlation of `mpg` with `weight` by using the `regress` command for each category in the double loop. The formatted table is built up by using the `outreg, merge()` command repeatedly.

```
. foreach f in 0 1 {
     foreach h in 2 2.5 3 {
        reg mpg weight if foreign==`f´ & hroom==`h´
        outreg, nocons noauto merge(row`f´)
     }
  }
  (output omitted)
```

The `outreg` command has the options `nocons` to suppress the constant coefficient and `noautosumm` to suppress the summary statistics, R^2, and the number of observations. The `merge(row‘f’)` option is the interesting part. This merges the coefficients into two separate tables, named `row0` and `row1`, each of which contains three columns of estimates for the categories of `hroom`.

The first time the `merge()` option is invoked for each table, we want it to create the first column, basically, to merge the results into a table that does not yet exist. The `merge()` option allows merging into a nonexistent table for convenience in loops like this. However, to make this work, it is important to clear out any preexisting table of the same name. Particularly, we want to clear out the table from the previous time we ran the same do-file. Otherwise, the table would get larger and larger each time the do-file is run. That is the reason for the two `outreg, clear()` commands just before the loop. An alternative to the `outreg, clear()` command, especially if one needs to clear many `frmttable` tables, is `mata: mata clear`, which clears all Mata memory structures, including tables.

Now that the tables for each row are created, we want to append them one below the other. We do this with a combination of the `replay()` and `append()` options. The `replay(row0)` option displays the named table (and can write it to a word processing document if `using` *filename* is specified). The `append(row1)` option appends the table `row1`, containing the second row, below the `row0` results. The rest of the `outreg` command below adds titles to the final table. The combined table now has the table name `row0`.

```
. outreg, replay(row0) append(row1) replace rtitle(Domestic \ "" \ Foreign)
> ctitle("", "", "Headroom", "" \ "Origin", "<=2.0", "2.5", ">=3.0")
> title(Effect of weight on MPG by origin and headroom)
                   Effect of weight on MPG by origin and headroom
                    ─────────────────────────────────────────────
                                          Headroom
                    Origin        <=2.0      2.5       >=3.0
                    ─────────────────────────────────────────────
                    Domestic      -0.006    -0.007     -0.005
                                  (3.62)**  (7.63)*    (8.00)**
                    Foreign       -0.018    -0.008     -0.011
                                  (1.87)    (2.14)     (2.64)*
                    ─────────────────────────────────────────────
```

This example shows how elaborate tables can be created by combining smaller parts by using the `merge()` and `append()` options and table names. Other useful options for this purpose are `addrows()` and `addcols()`, which add columns of text (which may include numbers) rather than statistics from Stata matrices.

3.7 Double statistics

Double statistics are statistics showing two numbers, such as a minimum–maximum range or a confidence interval. `frmttable` has the `doubles()` option to make it easy to display double statistics. The lower and upper values of double statistics are held in neighboring columns of `statmat()`, and the `doubles()` option indicates which columns are the second numbers of double statistics. `frmttable` automatically combines the two numbers into a single cell of the formatted table, with a dash between them.

The following code creates a Stata matrix, `conf_int`, containing the mean of several variables in the first column and the lower and upper confidence intervals in the second and third columns, respectively. The details are similar to the code in section 3.5 above.

```
. matrix conf_int = J(4,3,.)

. local i = 1

. foreach v in length weight headroom mpg {
  2.         summ `v´
  3.         matrix conf_int[`i´,1] = r(mean)
  4.         matrix conf_int[`i´,2] = r(mean) - invttail(r(N)-1, .05/2)
> *sqrt(r(Var)/r(N))
  5.         matrix conf_int[`i´,3] = r(mean) + invttail(r(N)-1, .05/2)
> *sqrt(r(Var)/r(N))
  6.         local i = `i´ + 1
  7. }
  (output omitted)

. matrix rownames conf_int = length weight headroom mpg

. matrix list conf_int

conf_int[4,3]
                 c1          c2          c3
  length  187.93243   182.77374   193.09113
  weight  3019.4595   2839.3983   3199.5206
headroom  2.9932432   2.7972422   3.1892443
     mpg  21.297297   19.956905    22.63769
```

For `frmttable` to display double statistics, it requires a row vector indicating which columns contain the second numbers of double statistics. In this example, there are second values of a double statistic in column 3, so the third element of the `doubles()` option matrix `dcols` is set to 1. The `doubles(dcols)` option causes the numbers in the second and third columns of `conf_int` (the lower and upper confidence limits) to be combined in the final table.

```
. matrix dcols = (0,0,1)

. frmttable, statmat(conf_int) substat(1) doubles(dcols) varlabels
> ctitles("",Summary statistics) sdec(0 \ 0 \ 0 \ 0 \ 2 \ 2 \ 1 \ 1)
```

```
 ----------------------------------------
                    Summary statistics
 ----------------------------------------
 Length (in.)              188
                       (183 - 193)
 Weight (lbs.)            3,019
                    (2,839 - 3,200)
 Headroom (in.)            2.99
                      (2.80 - 3.19)
 Mileage (mpg)             21.3
                      (20.0 - 22.6)
 ----------------------------------------
```

There is only one substatistic in `substat(1)` because the second statistic is a double statistic.

Confidence intervals have been integrated into the `outreg` command, making use of `frmttable`'s `doubles()` option, so the particular table in this example is easily created with the following commands:

```
. mean length weight headroom mpg
. outreg, stats(b,ci) nostars varlabels ctitles("",Summary statistics)
> bdec(0 0 2 1)
```

4 References

Baum, C. F., and J. P. Azevedo. 2001. outtable: Stata module to write matrix to LaTeX table. Statistical Software Components S419501, Department of Economics, Boston College. http://ideas.repec.org/c/boc/bocode/s419501.html.

Gallup, J. L. 2012. A new system for formatting estimation tables. *Stata Journal* 12: 3–28.

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

———. 2007. Making regression tables simplified. *Stata Journal* 7: 227–244.

Lokshin, M., and Z. Sajaia. 2008. Creating print-ready tables in Stata. *Stata Journal* 8: 374–389.

Wada, R. 2010. outreg2: Stata module to arrange regression outputs into an illustrative table. Statistical Software Components S456416, Department of Economics, Boston College. http://ideas.repec.org/c/boc/bocode/s456416.html.

Watson, I. 2011. tabout: Stata module to export publication quality cross-tabulations. Statistical Software Components S447101, Department of Economics, Boston College. http://ideas.repec.org/c/boc/bocode/s447101.html.

About the author

John Luke Gallup is an assistant professor in the Department of Economics at Portland State University.