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 teaching, including Stata with topics such as extended examples of techniques and interpretation of results, simulations of statistical concepts, and overviews of subject areas.

For more information on the Stata Journal, including information for authors, see the web page

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:

Subscriptions mailed to US and Canadian addresses:

- 3-year subscription (includes printed and electronic copy) $153
- 2-year subscription (includes printed and electronic copy) $110
- 1-year subscription (includes printed and electronic copy) $ 59
- 1-year student subscription (includes printed and electronic copy) $ 35

Subscriptions mailed to other countries:

- 3-year subscription (includes printed and electronic copy) $225
- 2-year subscription (includes printed and electronic copy) $158
- 1-year subscription (includes printed and electronic copy) $ 83
- 1-year student subscription (includes printed and electronic copy) $ 59
- 3-year subscription (electronic only) $153

Back issues of the Stata Journal may be ordered online at

http://www.stata.com/bookstore/sj.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 email sj@stata.com.
Using the file command to produce formatted output for other applications

Emma Slaymaker
Centre for Population Studies, London School of Hygiene and Tropical Medicine, 49-51 Bedford Square, London, WC1B 3DP UK
emma.slaymaker@lshtm.ac.uk

Abstract. The file command provides a way to produce tables for use in other application software. It can be especially useful for combining descriptive results (such as means and percentages) and results from significance tests. Extracting and manipulating the results directly from Stata matrices gives more control over arrangement, while other Stata functions may be used to control numeric formats. This tutorial includes examples based on survey data of both plain text and HTML output.

Keywords: dm0015, file, presentation of results, tables, HTML, spreadsheets, word processors, browsers

1 Introduction

The file command can be used to write text files; see [P] file. These can contain nothing but data or other ordinary text, or they can include mark-up or code for other applications to use. For example, it is easy within Stata to produce a text file or an HTML file describing a web page, which can be opened in other applications (for example, a text editor or a browser). There are advantages to this approach over log files or copy and paste: results from different calculations can be combined and arranged to suit, and the output of many similar tables can be automated to produce a series of results in a format (e.g., tab-delimited) easily read by other programs.

2 Example 1

Using the NHANES2 survey data, imagine trying to create a table giving the means of several variables for two groups of respondents, those with normal and those with high blood pressure. Given that variables of interest are age, lead, tcrresults, and tgrresults and that the blood pressure categories are identified by highbp, Stata commands (Stata 8) for this are

```
. webuse nhanes2, clear
. svyset [pw=finalwgt], psu(psu) strata(strata)
   pweight is finalwgt
   strata is strata
   psu is psu
```
. svymean age lead tcresult tgresult, by(highbp)
Survey mean estimation

pweight: finalwgt
Strata: strata
PSU: psu

Number of obs(*) = 10351
Number of strata = 31
Number of PSUs = 62
Population size = 1.172e+08

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Mean among respondents</th>
<th>Mean among respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>with normal blood pressure</td>
<td>with high blood pressure</td>
</tr>
<tr>
<td>age</td>
<td></td>
<td></td>
</tr>
<tr>
<td>highbp==0</td>
<td>41.03829</td>
<td>40.42351</td>
</tr>
<tr>
<td>highbp==1</td>
<td>52.5146</td>
<td>51.59932</td>
</tr>
<tr>
<td>lead</td>
<td></td>
<td></td>
</tr>
<tr>
<td>highbp==0</td>
<td>14.21601</td>
<td>13.65942</td>
</tr>
<tr>
<td>highbp==1</td>
<td>15.49412</td>
<td>14.79971</td>
</tr>
<tr>
<td>tcresult</td>
<td></td>
<td></td>
</tr>
<tr>
<td>highbp==0</td>
<td>210.4118</td>
<td>208.0773</td>
</tr>
<tr>
<td>highbp==1</td>
<td>235.7946</td>
<td>231.2528</td>
</tr>
<tr>
<td>tgresult</td>
<td></td>
<td></td>
</tr>
<tr>
<td>highbp==0</td>
<td>133.7581</td>
<td>128.9335</td>
</tr>
<tr>
<td>highbp==1</td>
<td>183.3632</td>
<td>170.8109</td>
</tr>
</tbody>
</table>

(*) Some variables contain missing values.

The output from this command is not convenient for final presentation. It would be preferable to have the means for each blood pressure category given side by side for easy comparison. The labeling of the results could also be improved. Using file, it is possible to produce a text file containing the results, as set out in table 1.

Table 1: Layout for results in example 1

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Mean among respondents</th>
<th>Mean among respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>with normal blood pressure</td>
<td>with high blood pressure</td>
</tr>
<tr>
<td>First characteristic</td>
<td>Mean value 1</td>
<td>Mean value 2</td>
</tr>
<tr>
<td>Second characteristic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>etc.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Stata stores the means calculated by the svymean command in a matrix called `e(est)`. A list of all the stored results is obtained with the ereturn command (see `[P] ereturn`). The following command lists the matrix on screen:
This matrix can be manipulated. The syntax becomes easier if a copy of this matrix is created first, which also makes it straightforward to extract matrix elements.

Capture these results in a new matrix:

```
matrix define myresults = e(est)
```

Get the value in the first row and second column of the matrix:

```
display myresults[1,2]
```

Using file, it is possible to create a tab-delimited text file that contains these results, arranged in any order. The first step is to tell Stata where it should put the results: it needs a name (strictly, a handle) by which to refer to the file. Here we use a temporary name so that there is no conflict with any name already in use.

Make a temporary filename for Stata to use:

```
tempname meansfile
```

Open a file, referred to by this temporary name, and call this file `example1.txt`:

```
file open 'meansfile' using "example1.txt", write replace
svymean age lead tcresult tgresult, by(highbp)
matrix define myresults = e(est)
local myvarnames = e(varlist)
```

We now need to know how many means were calculated, which can be read off from the size of the matrix. Stata stores the results from `svymean` in a matrix with only one row, so we need to find the number of columns.

Make a new local macro that contains the number of columns in the matrix, and show this on screen:

```
local matrixcols = colsof(myresults)
disp "Matrix has " "\"matrixcols\" columns"
```

The first column of the table will contain a description of the variable; here we use the variable label. If the variable does not have a label, this will leave a blank in the table. It would be possible to add code that would use the variable name if the variable is not labeled; however, as variable names are rarely suitable for final presentation, it is better to ensure beforehand that all the variables are labeled.
The next stage requires a loop. The first part of the loop gets the variable name from another piece of information stored by Stata, e(varlist). Having selected a variable name, we use a local macro to hold the text of the variable label, which can then be inserted into the final table.

The second part of the loop picks out the relevant results from the matrix of means. In this example, there are two categories for the by() variable, normal and high blood pressure. We therefore extract two means for each line of the table. To facilitate this, the loop starts with a value of 1 for \( j \) and increments by 2 each time. Each run of the loop finds the values in the matrix at positions \([1,j]\) and \([1,j+1]\). These values are then added into the text file after the description of the variable.

Create a local macro to determine the position in e(varlist) from which to extract the variable name:

```stata
local i = 1
```

Go through the matrix, two columns at a time:

```stata
forvalues j = 1(2)matrixcols{
```

Get the variable names one at a time:

```stata
local myvar: word `i' of `myvarnames'
disp "The current variable is " "myvar"
```

Get the variable label to use instead of the name in the results table:

```stata
local name: variable label 'myvar'
```

Increment local macro ‘\( i \)’ so that on the next loop the next variable name is selected:

```stata
local i = 'i' + 1
```

Pick out pairs of values for normal BP and high BP:

```stata
local mean_in_lowbp = myresults[1,'j']
local mean_in_highbp = myresults[1,'j'+1]
```

Write these results to the file:

```stata
file write 'meansfile' ("'name'") _tab ('mean_in_lowbp') _tab ('mean_in_highbp') _n
```

The last instruction adds the extracted information into the text file (which Stata refers to as ‘meansfile’). The content is enclosed in parentheses and separated by tab characters.

- _tab tells Stata to put a tab character in the file.
- _n tells Stata to go to a new line.
The text from the label of the variable being used is put into the file using the macroname ‘name’. Anything that Stata should evaluate before putting into the file must be included within parentheses.

- Text can be included in the file between double quotes. In this example, the name of the local macro that holds the variable label text is placed between double quotes.

- The means are contained in local macros `mean_in_lowbp` and `mean_in_highbp`.

The curly bracket closes the loop so that this set of instructions is repeated for each variable. After the loop, all that remains is to close the file.

```stata
close 'meansfile'
```

This produces a table with columns separated by tab characters and similar to table 2.

Table 2: Content of text file generated by example 1

<table>
<thead>
<tr>
<th>Variable</th>
<th>Lowbp</th>
<th>Highbp</th>
</tr>
</thead>
<tbody>
<tr>
<td>age in years</td>
<td>41.038287</td>
<td>52.514597</td>
</tr>
<tr>
<td>lead (mcg/dL)</td>
<td>14.216012</td>
<td>15.494117</td>
</tr>
<tr>
<td>serum cholesterol (mg/dL)</td>
<td>210.41181</td>
<td>235.79456</td>
</tr>
<tr>
<td>serum triglycerides (mg/dL)</td>
<td>133.75813</td>
<td>183.36324</td>
</tr>
</tbody>
</table>

### Example 2

Additional text can be added to the file: in this example, a caption for the table and a header row. Formats can be applied to the results. Mean estimates are rounded to two decimal places using the `round()` function. We also include `p`-values for the difference between the means in each group by first using the `test` command and then putting the resulting `p`-value into a local macro.

The format extended macro function may be used to present the `p`-value to four decimal places. Using format merely alters the appearance of the number; Stata allows enough space for the unformatted number when writing to file. Stata therefore writes blank spaces before or, depending on the choice of format, after the number, which is undesirable for the means, as the table columns would then not align properly. However, `p`-values are a special case; rounding would render 0.000 as 0, which is also unsuitable. Thus for `p`-values, format is more useful. Using a left-justified format ensures that there is a 0 before the decimal point and that any surplus spaces come after the number.

(Continued on next page)
Using the file command to produce formatted output

tempname meansfile
file open `meansfile' using "d:\emma\stata\example2.txt", write replace
file write `meansfile' "Table 3. Selected background characteristics of */
   */ "NHANES2 respondents with normal and with high blood pressure" _n
file write `meansfile' ("Characteristic") _tab ("Normal BP") _tab ("High BP") */
   */ _tab ("p-value") _n _n
svymean age lead tcresult tgresult, by(highbp) complete
matrix define myresults = e(est)
local myvarnames = e(varlist)
llocal matrixcols = colsof(myresults)
disp "'matrixcols'
local i=1
forvalues j = 1(2)'matrixcols'{
   local myvar: word 'i' of 'myvarnames'
disp "'myvar'"
   local name: variable label 'myvar'
   local i = 'i' + 1
Round the results to two decimal places:
   local mean_in_lowbp = round(myresults[1,'j'],0.01)
   local mean_in_highbp = round(myresults[1,'j'+1],0.01)
Use test to get the p-value for the significance of the difference:
   cap test ['myvar']0 = ['myvar']1
Create a local macro equal to the first four decimal places of the p-value and with a
leading 0:
   local pvalue : display %-9.4f r(p)
   file write `meansfile' ("'name'") _tab ("mean_in_lowbp") _tab /*
   */ ("mean_in_highbp") _tab ("pvalue") _n
} file close `meansfile'

Again, the resulting table can be opened using a spreadsheet or word processor. If
you want to edit in, for example, Microsoft Word, you can take advantage of built-in
features for managing tables to produce one similar to table 3.

Table 3: Selected background characteristics of NHANES2 respondents with normal and
with high blood pressure

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Normal BP</th>
<th>High BP</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>age in years</td>
<td>41.1</td>
<td>52.4</td>
<td>0.0000</td>
</tr>
<tr>
<td>lead (mcg/dL)</td>
<td>14.16</td>
<td>15.5</td>
<td>0.0007</td>
</tr>
<tr>
<td>serum cholesterol (mg/dL)</td>
<td>208.56</td>
<td>239.91</td>
<td>0.0000</td>
</tr>
<tr>
<td>serum triglycerides (mg/dL)</td>
<td>131.61</td>
<td>202.86</td>
<td>0.0000</td>
</tr>
</tbody>
</table>
4  Example 3

It is possible to add HTML tags to a file and make a web page. Let us expand the previous example to include additional variables and output in HTML. The estimation commands are unchanged, but more information is now written to the file, which has the extension .htm. Some extra commands put the essential HTML mark-up at the beginning and end of the file. In the middle of the file, where the results table is written, there are no _tabs to separate the columns, but instead the HTML tags (<tr>, <td>, etc.) are written into the file. A screenshot of the resulting HTML document is shown in table 4.

```
tempname meansfile
file open "meansfile" using "d:\emma\stata\example3.htm", write replace

Start the HTML document:

    file write 'meansfile' "<HTML><HEAD></HEAD><BODY>" _n

Write a caption for the table:

    file write 'meansfile' "<p><b>Table 4.</b>Selected background characteristics" /*
    */ "of NHANES2 respondents with normal and with high blood pressure</p>" _n

Add a horizontal line at the top of the table:

    file write 'meansfile' "<hr width="70%" align="left">" _n

Write the header row for the table:

    file write 'meansfile' "<table width="70%"><tr><td>Characteristic</td> /*
    */ "Normal BP</td><td>High BP</td><td>p-value</td></tr>" _n

Add a horizontal line under the header row:

    file write 'meansfile' "<tr><td colspan="4"><hr width="100%" align="left">"</td></tr>" _n

svymean age lead tcresult tgresult hgb hct tibc iron,by(highbp) complete
This is exactly the same as before:

matrix define myresults = e(est)
local myvarnames = e(varlist)
local matrixcols = colsof(myresults)
disp ""matrixcols"
local i = 1
forvalues j = 1(2)matrixcols'{
    local mean_in_lowbp = round(myresults[1,'j'],0.01)
    local mean_in_highbp = round(myresults[1,'j'+1],0.01)
    local myvar: word 'i' of 'myvarnames'
disp ""myvar"
    local name: variable label 'myvar'
cap test ["myvar"]0 = ["myvar"]1
local pvalue : display %-9.4f r(p)
```
Using the `file` command to produce formatted output

Write these results to the HTML file:

```stata
file write 'meansfile' "<tr><td>" ('name') "</td><td>" ('mean_in_lowbp') /*
+ "</td><td>" ('mean_in_highbp') "</td><td>" ('pvalue') "</td></tr>
local i = 'i' + 1
```

Finish the table, and end the HTML document:

```stata
file write 'meansfile' "</table></BODY></HTML>"
file write 'meansfile' "<hr width="70%" align="left">
file close 'meansfile'
```

Table 4: Selected background characteristics of NHANES2 respondents with normal and high blood pressure

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Normal BP</th>
<th>High BP</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>age in years (years)</td>
<td>43.1</td>
<td>72.4</td>
<td>0.0000</td>
</tr>
<tr>
<td>blood (mg/dL)</td>
<td>14.36</td>
<td>15.5</td>
<td>0.0007</td>
</tr>
<tr>
<td>serum cholesterol (mg/dL)</td>
<td>205.56</td>
<td>299.91</td>
<td>0.0000</td>
</tr>
<tr>
<td>serum triglyceride (mg/dL)</td>
<td>195.61</td>
<td>320.86</td>
<td>0.0000</td>
</tr>
<tr>
<td>hemoglobin (g/dL)</td>
<td>14.32</td>
<td>14.54</td>
<td>0.0557</td>
</tr>
<tr>
<td>hematocrit (%)</td>
<td>42.04</td>
<td>42.69</td>
<td>0.0229</td>
</tr>
<tr>
<td>total iron bind. cap. (mg/dL)</td>
<td>386.17</td>
<td>375.15</td>
<td>0.0037</td>
</tr>
<tr>
<td>serum iron (mg/dL)</td>
<td>106.98</td>
<td>100.88</td>
<td>0.0032</td>
</tr>
</tbody>
</table>

5 Summary

Output of results using `file` is a versatile way to produce tables for use in other application software. It can be especially useful for combining descriptive results (such as means and percents) and results from significance tests. Extracting and manipulating the results directly from Stata matrices gives more control over arrangement, and other Stata functions may be used to control numeric formats. If `file` is used to create a text file, the more superficial formatting, such as rules and spacing, can easily be applied using a word processor or spreadsheet. If the results are output with additional code or mark-up (e.g., HTML), it is possible to produce a finished table directly from Stata.
About the Author

Emma Slaymaker is a Research Fellow in the Centre for Population Studies, London School of Hygiene and Tropical Medicine, UK. Research interests include the analysis of population surveys to estimate the prevalence of sexual behaviors that place individuals at risk of sexually transmitted infections and the application of new methods to improve these estimates.