The Stata Journal

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

The Stata Journal (2005)
5, Number 2, pp. 239–247

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`, `tcresult`, and `tgresult` 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                              Number of obs(*) =     10351
Strata:   strata                                Number of strata =        31
PSU:      psu                                   Number of PSUs   =        62
                                                Population size  = 1.172e+08
```

Mean	Subpop.	Estimate	Std. Err.	[95% Conf.	Interval]	Deff
age						
	highbp==0	41.03829	.3014329	40.42351	41.65306	3.583525
	highbp==1	52.5146	.44877	51.59932	53.42987	1.295673
lead						
	highbp==0	14.21601	.2729028	13.65942	14.7726	8.34335
	highbp==1	15.49412	.3404777	14.79971	16.18853	1.811026
tcresult						
	highbp==0	210.4118	1.144654	208.0773	212.7463	5.352918
	highbp==1	235.7946	2.226881	231.2528	240.3363	2.202582
tgresult						
	highbp==0	133.7581	2.365556	128.9335	138.5827	3.247626
	highbp==1	183.3632	7.574167	167.9156	198.8109	1.446885

```
(*) 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

Characteristic	Mean among respondents with normal blood pressure	Mean among respondents with high blood pressure
First characteristic	Mean value 1	Mean value 2
Second characteristic		
etc.		

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:

```
. matrix list e(est)
e(est)[1,8]
          age:       age:      lead:      lead:  tcresult:  tcresult:  tgresult:
            0          1          0          1          0          1          0
r1  41.038287  52.514597  14.216012  15.494117  210.41181  235.79456  133.75813
     tgresult:
             1
r1  183.36324
```

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]
52.514597
```

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:

```
local i = 1
```

Go through the matrix, two columns at a time:

```
forvalues j = 1(2)‘matrixcols’{
```

Get the variable names one at a time:

```
        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:

```
        local name: variable label ‘myvar’
```

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

```
        local i = ‘i’ + 1
```

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

```
        local mean_in_lowbp = myresults[1,‘j’]
        local mean_in_highbp = myresults[1,‘j’+1]
```

Write these results to the file:

```
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.

```
file 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

age in years	41.038287	52.514597
lead (mcg/dL)	14.216012	15.494117
serum cholesterol (mg/dL)	210.41181	235.79456
serum triglycerides (mg/dL)	133.75813	183.36324

3 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*)

```
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)
local 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

Characteristic	Normal BP	High BP	p-value
age in years	41.1	52.4	0.0000
lead (mcg/dL)	14.16	15.5	0.0007
serum cholesterol (mg/dL)	208.56	239.91	0.0000
serum triglycerides (mg/dL)	131.61	202.86	0.0000

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 `_tab`s 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>"
```

Add a horizontal line at the top of the table:

```
file write ‘meansfile’ ‘"<hr width="70%" align="left">"’
```

Write the header row for the table:

```
file write ‘meansfile’ ‘"<table width="70%"><tr><td>Characteristic</td>"’ /*
    */ ‘"<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>"’
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)
```

Write these results to the HTML file:

```
file write ‘meansfile’  "<tr><td>" ("‘name’") "</td><td>" (‘mean_in_lowbp’)/*
     */ "</td><td>" (‘mean_in_highbp’) "</td><td>" ("‘pvalue’") "</td></tr>" _n
        local i = ‘i’ + 1
}
```

Finish the table, and end the HTML document:

```
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

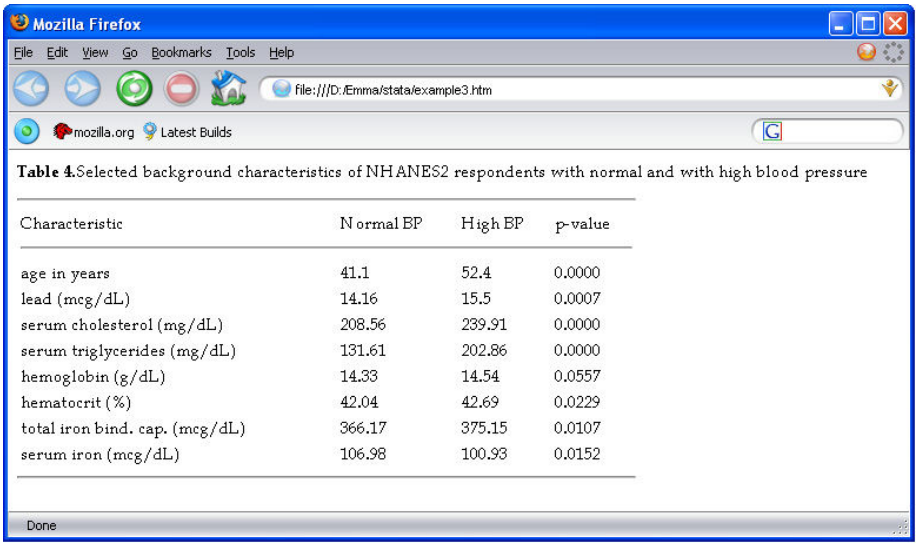

Table 4. Selected background characteristics of NHANES2 respondents with normal and with high blood pressure

Characteristic	Normal BP	High BP	p-value
age in years	41.1	52.4	0.0000
lead (mcg/dL)	14.16	15.5	0.0007
serum cholesterol (mg/dL)	208.56	239.91	0.0000
serum triglycerides (mg/dL)	131.61	202.86	0.0000
hemoglobin (g/dL)	14.33	14.54	0.0557
hematocrit (%)	42.04	42.69	0.0229
total iron bind. cap. (mcg/dL)	366.17	375.15	0.0107
serum iron (mcg/dL)	106.98	100.93	0.0152

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.