Processing Results in Stata

Most Stata functions (such as **reg**) display their results in the Stata results window. Sometimes this is not quite sufficient: we might want to either preserve some of the output and use it in future computations, or produce tables (e.g. for publication) in format different from that in the display window.

In this handout we'll go over several examples of such results processing, and give an example of how to install packages from the public repositories.

In this exercise we'll go over these examples and a few more common Stata commands. The examples were produced using Stata for linux, running on Skew; but in most cases the instructions apply cross-platform.

Downloading packages

As before, all commands in the text are given in bold font; UNIX commands are indicated by '**$**' and Stata commands by '**.**' (you do not have to enter these signs).

We'll use several user-contributed packages that are not a part of Stata distribution for this exercise. In particular, **statsmat**, **outtable**, **tabstatmat**, **estout** and **esttab** will be introduced. These and other packages are available from the Internet and can be found with **net search,** e.g.:

.net search regression tables

```
. net search regression tables
(contacting http://www.stata.com)

12 packages found (Stata Journal and STB listed first)
---------------------------------------------------

st0085_1 from http://www.stata-journal.com/software/sj7-2
    SJ7-2 st0085_1.  Update: Making regression tables simplified / Update:
    Making regression tables simplified / by Ben Jann, ETH Zurich / Support:
    jann@soz.gess.ethz.ch / After installation, type help estout, esttab,
    eststo, and estadd

st0085 from http://www.stata-journal.com/software/sj5-3
    SJ5-3 st0085.  Making regression tables from stored... / Making regression
    tables from stored estimates / by Ben Jann, ETH Zurich / Support:
    jann@soz.gess.ethz.ch / After installation, type help estout, estoutdef,
    / and estadd

sg97 from http://www.stata.com/stb/stb46
    STB-46 sg97.  Formatting regression output for published tables. / STB
    insert by John Luke Gallup, developIT.org / Support:
    john_gallup@alum.swarthmore.edu / After installation, see help outreg.

metamodbias from http://www.epi.bris.ac.uk/user/rogerh
--    metamodbias.  Modified test for bias in meta-analysis. / Roger Harbord,
    Dept. of Social Medicine, University of Bristol. /
    roger.harbordbristol.ac.uk / A modified regression test for bias in
    meta-analysis of 2x2 tabl
--Break--
```

If you are using Stata GUI, click on the link to open a help window that offers an option of installing the package. Alternatively, you can install the commands with **net install** or **ssc install.** Let us make sure the commands we are going to be using are actually installed:

. ssc install estout, replace
. ssc install statsmat, replace
. ssc install matsave, replace
. ssc install outtable, replace
. ssc install tabstatmat, replace

Saving output of functions

Let us load the Stata sample dataset **auto.dta**:

```
. sysuse auto
(1978 Automobile Data)

. d

Contains data from /usr/local/stata10/ado/base/a/auto.dta
  obs:            74                          1978 Automobile Data
 vars:            12                          13 Apr 2007 17:45
 size:         3,478 (99.9% of memory free)   (_dta has notes)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
make            str18  %-18s                  Make and Model
price           int    %8.0gc                 Price
mpg             int    %8.0g                  Mileage (mpg)
rep78           int    %8.0g                  Repair Record 1978
headroom        float  %6.1f                  Headroom (in.)
trunk           int    %8.0g                  Trunk space (cu. ft.)
weight          int    %8.0gc                 Weight (lbs.)
length          int    %8.0g                  Length (in.)
turn            int    %8.0g                  Turn Circle (ft.)
displacement    int    %8.0g                  Displacement (cu. in.)
gear_ratio      float  %6.2f                  Gear Ratio
foreign         byte   %8.0g       origin     Car type
-------------------------------------------------------------------------------
Sorted by:  foreign
```

Most Stata functions, in addition to producing output such as above, return values that are accessible for other Stata commands. This is typically done via either a **return** (for summary statistics and similar commands) or via **ereturn** (for estimation commands). To see the values returned, use **return list** or **ereturn list**.

```
. return list

scalars:
             r(changed) =  0
            r(widthmax) =  60000
               r(k_max) =  5000
               r(N_max) =  201647
               r(width) =  43
                   r(k) =  12
                   r(N) =  74
```

As we can see, even **describe** command is not an exception, although the values returned are probably not the most useful statistics.

Let's see what a more useful command like **summarize** gives us:

```
. summarize price

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       price |         74    6165.257    2949.496       3291      15906

. return list

scalars:
                    r(N) =  74
                r(sum_w) =  74
               r(mean) =  6165.256756756757
                r(Var) =  8699525.974268788
                 r(sd) =  2949.495884768919
               r(min) =  3291
               r(max) =  15906
               r(sum) =  456229
```

Now we see some useful statistics, most (although not all) of which are also displayed on screen. We can access these by typing **r(N)**, etc. When another command is issued, they will be overwritten, so we might want to save them into variables that we can access later:

```
. scalar meanprice=r(mean)

. display meanprice
6384.6818
```

In this case, we created a scalar variable equal to the mean of price.

Now we can, for example, group all observations into two groups – one with price above the mean, and one below:

```
. gen hiprice=price>meanprice if price~=.

. tab hiprice, missing

    hiprice |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |         56       75.68       75.68
          1 |         18       24.32      100.00
------------+-----------------------------------
      Total |         74      100.00
```

Note the "**if price~=.**" statement above. Unlike some other statistical software, Stata does not treat missing values in a very standard way. In particular, a missing value (denoted as '.') is always considered the largest possible number in existence – and were there any cars without price data, they would be automatically assigned to the "high" group by a simple "**>**" operation, instead of generating a missing value for ***hiprice***. In this case, there was no reason to worry because there are no missing price values, as we see from the **tab** output.

In the previous section we saw an example of using **tabstat** to generate a summary table. Let us create one again:

```
. tabstat price mpg, s(mean) by(foreign) save

Summary statistics: mean
  by categories of: foreign (Car type)

 foreign |     price       mpg
---------+--------------------
Domestic |  6072.423  19.82692
 Foreign |  6384.682  24.77273
---------+--------------------
   Total |  6165.257   21.2973
------------------------------
```

Note 'save' option – it instructs **tabstat** to save the results so that we can access them:

```
. return list

macros:
              r(name2) : "Foreign"
              r(name1) : "Domestic"

matrices:
              r(Stat2) :  1 x 2
              r(Stat1) :  1 x 2
           r(StatTotal) :  1 x 2
```

The matrices in question refer to rows of the matrix above. This is not the most convenient way of doing this, and function **tabstatmat** helps here:

```
. tabstatmat mpgstat

mpgstat[3,2]
              price        mpg
Domestic  6072.4231  19.826923
 Foreign  6384.6818  24.772727
   Total  6165.2568  21.297297
```

Now once we have the table, we probably want to format it and insert into our future paper about cars of 1978. There are several ways to do that, such as:

- Use **matsave** to save the matrix into a separate dataset which you can then process separately, export to Excel, etc.
- Use **outtable** to directly generate formatted output

Here is an example of the **matsave**:

```
. matsave mpgstat, saving replace
data in memory will be removed temporarily, and restored later
Press any key to continue, or Break to abort
matrix mpgstat saved
(1978 Automobile Data)
data in memory restored
```

We have just created a file **mpgstat.dta** in our working directory. Later we can return to it, process the data in the way we see fit (e.g. merging with other summary tables) and export to a spreadsheet with **outsheet.**

Unlike **matsave, outtable** immediately saves a formatted table:

. matname mpgstat Price Mileage, columns(.) explicit
. outtable using mpgtable, mat(mpgstat) replace nobox caption(Summary Statistics) center clabel(Table1) format(%9.2f)

generates a file "mpgtable.tex" in the current directory:

```
% matrix: mpgstat file: mpgtable.tex   9 Oct 2007 14:09:58
\begin{table}[htbp]
\caption{\label{Table1} Summary Statistics}\centering\medskip
\begin{tabular}{lcc} \hline \hline
 & Price  & Mileage  \\  \hline
Domestic &   6072.42 &     19.83 \\
Foreign &   6384.68 &     24.77 \\
Total &   6165.26 &     21.30 \\
\hline \hline \end{tabular}
\end{table}
```

Which, when compiled in LaTeX, generates the following table:

Table 1.1. Summary Statistics

	Price	Mileage
Domestic	6072.42	19.83
Foreign	6384.68	24.77
Total	6165.26	21.30

Another useful function that allows to generate tables of various statistics is **statsmat**. It operates similarly to **tabstat** and directly creates matrices of summary statistics.

Accessing and formatting estimation results

Functions like **reg**, instead of returning results with **return,** use **ereturn** instead:

```
. reg price mpg rep78  displacement

      Source |       SS       df       MS              Number of obs =      69
-------------+------------------------------           F(  3,    65) =   13.18
       Model |   218184988     3  72728329.4           Prob > F      =  0.0000
    Residual |   358611971    65  5517107.24           R-squared     =  0.3783
-------------+------------------------------           Adj R-squared =  0.3496
       Total |   576796959    68  8482308.22           Root MSE      =  2348.9

------------------------------------------------------------------------------
       price |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         mpg |    -85.948   73.50957    -1.17   0.247    -232.7568    60.86074
       rep78 |   881.9648   319.7761     2.76   0.008     243.3278    1520.602
displacement |   16.96924   4.651345     3.65   0.001      7.67986    26.25861
       _cons |   1612.162   2528.171     0.64   0.526    -3436.944    6661.269
------------------------------------------------------------------------------

. ereturn list

scalars:
                  e(N) =  69
               e(df_m) =  3
               e(df_r) =  65
                  e(F) =  13.18233018866571
                 e(r2) =  .3782700042526083
               e(rmse) =  2348.852324614606
                e(mss) =  218184988.084482
                e(rss) =  358611970.7850834
               e(r2_a) =  .3495747736796517
                 e(ll) =  -631.4025111524516
               e(ll_0) =  -647.7986144493904

macros:
            e(cmdline) : "regress price mpg rep78  displacement"
              e(title) : "Linear regression"
                e(vce) : "ols"
             e(depvar) : "price"
                e(cmd) : "regress"
         e(properties) : "b V"
            e(predict) : "regres_p"
              e(model) : "ols"
          e(estat_cmd) : "regress_estat"

matrices:
                  e(b) :  1 x 4
                  e(V) :  4 x 4

functions:
             e(sample)
```

Similarly to **r(…)**, these can be accessed with **e(…)**:

```
. display e(F)
13.18233
```

There are functions that make use of these results and produce "pretty" output. One is **outreg** that we saw in the last session, another popular (and more powerful) one is **estout**.

For convenience, it is possible to store estimation results in a named structure, and call functions like **estout** later. This way you can group computations together, separately from output statements, and combine output from multiple models into one table. Let us estimate two regressions:

```
. reg price mpg

      Source |       SS       df       MS              Number of obs =      74
-------------+------------------------------           F(  1,    72) =   20.26
       Model |   139449474     1   139449474           Prob > F      =  0.0000
    Residual |   495615923    72  6883554.48           R-squared     =  0.2196
-------------+------------------------------           Adj R-squared =  0.2087
       Total |   635065396    73  8699525.97           Root MSE      =  2623.7

------------------------------------------------------------------------------
       price |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         mpg |  -238.8943   53.07669    -4.50   0.000    -344.7008   -133.0879
       _cons |   11253.06   1170.813     9.61   0.000     8919.088    13587.03
------------------------------------------------------------------------------

. estimates store model0

.  quietly reg price mpg rep78  displacement

. estimates store model1
```

Note how keyword **quietly** suppressed the output from 2nd regression, while producing the required statistics.

Now we can use **esttab** or the more powerful **estout** to create a table of regression output:

```
. esttab model0 model1

--------------------------------------------
                      (1)             (2)
                    price           price
--------------------------------------------
mpg                -238.9***       -85.95
                  (-4.50)         (-1.17)

rep78                               882.0**
                                   (2.76)

displacement                        16.97***
                                   (3.65)

_cons             11253.1***       1612.2
                   (9.61)          (0.64)
--------------------------------------------
N                     74              69
--------------------------------------------
t statistics in parentheses
* p<0.05, ** p<0.01, *** p<0.001
```

For example,

```
estout model0 model1 using model1.txt,
    cells(b(star fmt(%9.3f)) se(par)) style(fixed) legend
    stats(r2 N, fmt(%9.3f %9.0f) labels("R$^2$" "Obs."))
    starlevels(* 0.10 ** 0.05 *** 0.01) stard collabels(,none)
    varlabels(price "Price" mpg "Mileage" rep78 "Repair record"
    displacement "Displacement" _cons "Constant") replace
```

would create a file **model1.txt** with the following table:

```
                    model0           model1
Mileage          -238.894 ***       -85.948
                 (53.077)          (73.510)
Repair rec~d                        881.965 ***
                                  (319.776)
Displacement                         16.969 ***
                                    (4.651)
Constant         11253.061 ***     1612.162
                (1170.813)        (2528.171)
R$^2$                0.220             0.378
Obs.                    74                69
* p<0.10, ** p<0.05, *** p<0.01
```

We can also use **estout** to produce html- or TeX-formatted output. So,

```
esttab model0 model1 using models2.tex, style(tex) nonum legend
  stats(r2 N, fmt(%9.3f %9.0f) labels("R$^2$" "Obs."))
  title("Price as dependent variable")
  mlabels("\textbf{I}" "\textbf{II}")
  varlabels(price "Price" mpg "Mileage" rep78 "Repair record"
  displacement "Displacement" _cons "Constant") replace;
```

Will produce LaTeX code that will compile into the following table:

Table 1.2. Price as dependent variable

	I	II
Mileage	-238.9***	-85.95
	(-4.50)	(-1.17)
Repair record		882.0**
		(2.76)
Displacement		16.97***
		(3.65)
Constant	11253.1***	1612.2
	(9.61)	(0.64)
R^2	0.220	0.378
Obs.	74	69

t statistics in parentheses

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$