Methodology Seminars

Introduction to Stata - Class 3 and 4

Maria Perrotta

maria.perrotta@iies.su.se

Fall 2009

Outline of the course

Macros

A macro is used as shorthand – you type a short macro name but are actually referring to some longer name or string of characters.
For example, you may use the same list of independent variables in several regressions and want to avoid retyping the list several times. Just assign this list to a macro.
Macros are of two types – local and global. Local macros will only work within the program or do-file in which they are created. Global macros will work in all programs and do files.
Local macros are recalled with the special quotes `‘’`. Global macros are recalled with the $ symbol.

Example

```
local varlist gdppc edexp aidpc oecd
reg pri ‘varlist’ if year==1990
xtreg sec icrg ‘varlist’ if year>=1995

global varlist gdppc edexp aidpc oecd
reg pri $varlist
```

Looping
foreach

`foreach`-processing allows you to easily repeat Stata commands. The `foreach` loop allows for a list of commands

Example

```
foreach var in bike moto car boat motorboat {
   egen mean_`var'=mean(`var')
   display "Mean of `var' =" mean_`var'
}
```

`foreach` is followed by a macro name that you assign (e.g. `var`) and `in` is followed by the list of arguments that you want to loop on (e.g. `bike moto car boat motorboat`). This command can be easily used with variable names, numbers, or any string of text.
At the end of the list is an open bracket { that signifies the start of the repeated set of commands. Everything between the two brackets { } will be executed for each item in the list.

Looping

forvalues

Another command with very similar function and syntax is `forvalues`. `forvalues`-processing is more suitable for long lists that can be written as a range or interval; hence, it works only if we want to loop over numerical values.

Example

```
foreach t=1984/2002 {
   egen maxcar_`t'=max(car) if year==`t'
}
```

Looping

while

An alternative is the `while` command, for variable loop lengths – as long as the while condition is true, the loop will keep on looping.

Example

```
local i=0
while `i'<=3 {
   tempvar freq
   quietly egen `freq'=count(country) if icrg==`i'
   so `freq'
   display `i' " " `freq'
   local i=`i'+.25
}
```

The first command defines a local macro that is going to be the loop increment - it can be seen as a counter. The second command is the while condition that must be satisfied if the loop is to be executed. This effectively sets the upper limit of the loop counter. The final command before the close bracket } increments the counter, readying it to go through the loop again. The increase in the counter can be any value.

Branching

Branching allows you to do one thing if a certain condition is true, and something else when that condition is false. For example, suppose you want to display the minimum value of the corruption index for OECD countries and the maximum value for the others:

Example

```
local i=1
while `i'<=5 {
if `i'<=4 {
local function max
}
else {
local function min
}
tempvar corrupt
quietly egen `corrupt'=`function'(icrg) if inc==`i'
so `corrupt'
display `i' " " `corrupt'
local i=`i'+1
}
```

Note

It is very important to get the brackets {} right in your programs.

First: every `if` statement, every `else` statement, and every `while` statement must have their conditions fully enclosed in their own set of brackets.

Second: nothing should be typed after a close bracket, as Stata automatically moves on to the next line when it encounters a close bracket.

Finally: it is necessary to place the brackets and their contents on different lines, irrespective of whether the brackets contain one or more lines of commands.

Defining a program

A program contains a set of commands activated by a single command. Programs are useful when we want to repeat a fixed combination of separate Stata commands many times.

Example

We can define a program that displays the value of a particular variable for a particular country and year, every time we want.

```
program define show
tempvar obs
quietly gen `obs'=`1' if (countrycode=="`2'" & year==`3')
so `obs'
display"`1' of country `2' in `3' is:" `obs'
end
```

Once the program `show` is defined, we can use it exactly as any other Stata command.

```
show icrg USA 1984
```

the output will be:

```
icrg of country USA in 1984 is:  5.416666507721
```

User-written programs

A number of researchers have created their own programs. These extra commands range from exotic econometric techniques to mini time-saving routines. For example, the commands `outreg2` and `xtabond2`.

If you have Stata on your computer, you can locate the relevant command and then install it into your copy of Stata, using `search` or `findit`. From findit, you can click to go to a source or to install additions.

Estimation

Stata can perform many different models of estimation. Most have a similar syntax:

```
command varlist [weight] [if exp] [in range] [, options]
```

- The 1st variable in the varlist is the dependent variable, and the remaining ones are the independent variables.
- You can use Stata's syntax to specify the estimation sample, no need to create specific dataset.
- You can, at any time, review the last estimates by typing the estimation command without arguments.

Common commands

`regress` is the most basic form of regression, fits a linear model of `depvar` on `varlist` using ordinary least squares.

Here is an abbreviated list of other regression commands that may be of interest:
`xtreg` fixed- and random-effects linear models
`anova` analysis of variance and covariance
`heckman` Heckman selection model
`ivreg` instrumental variables (2SLS) regression
`qreg` quantile regression
`svyregress` linear regression with survey data
`tobit` tobit regression
`xtabond` Arellano-Bond linear, dynamic panel-data estimator
`xtregar` fixed- and random-effects linear models with an AR(1) disturbance

Common options

- The `level()` option is used to specify the width of the confidence interval. The default is `level(95)`.
- The `robust` option specifies that the Huber-White ”sandwich” estimator of variance is to be used in place of the traditional calculation. The standard errors take into account heteroscedasticity.
- The `cluster(varname)` option specifies that the observations are i.i.d. across groups (clusters) but allows for any pattern of heteroskedasticity and autocorrelation *within* groups. `varname` specifies the variable that identifies the groups. For example, it is reasonable to expect that in a panel of countries, errors are correlated across time for the same country, but independent across countries. Then, it is common practice to cluster the standard errors at the country level.
- `noconstant` excludes the intercept from the model (in practice, it constrains the intercept at 0).

Regression output

- The top-left corner gives the ANOVA decomposition of the sum of squares in the dependent variable (Total) into the explained (Model) and unexplained (Residual).
- The top-right corner gives the statistical significance results for the model as a whole.
- The bottom section gives the results (coefficients, s.e., t-test, confidence intervals) for the individual independent variables.
- Notice that Stata automatically adds the constant term or intercept to the list of independent variables: use the `noconstant` option if you want to exclude it.

Find more at:
http://www.ats.ucla.edu/stat/stata/output

Post-estimation

Once you have carried out your estimation, there are a number of post-estimation commands that are useful:

- You can recall the estimates, VCM, standard errors, etc...: `eststo`, `esttab`, and more;
- You can carry out hypothesis testing: `test` (Wald tests), `testnl` (non-linear Wald tests), `lrtest` (likelihood-ratio tests), `hausman` (Hausman's specification test);
- A number of predicted values can be obtained after all estimation commands. The most important are the predicted values for the dependent variable (fitted values) and the residuals, that can be obtained with the command `predict varname` and the option `, residuals` respectively.

Post-estimation diagnostics

Stata comes with a large amount of regression diagnostic tools. We will focus on two useful tools for detecting influential observations and looking at partial correlations.

- `lvr2plot` (read leverage-versus-residual squared plot). Leverage tells you how large the influence of a single observation on the estimated coefficients is. Observations with high values could potentially be driving the results obtained (especially if they also have a large squared residual) so it is a good idea to check whether excluding them changes anything.
- `avplot` (added-variable plot). Graphs the partial correlation between a specified regressor and the dependent variable.

Post-estimation tests

The results of each estimation automatically include for each independent variable a t-test (for linear regressions) on the null hypothesis that the “true” coefficient is equal to zero. Additionally, it is possible to test any linear hypotheses about the coefficients using the `test` command, such as:

Example

`test y1985` tests that coefficient on y1985 equals 0
`test y1985=0.5` tests that coefficient on y1985 equals 0.5
`test y1985 y1990` tests that coefficients on y1985 & y1990 are jointly zero
`test y1985+y1990=-0.5` tests that coefficients on y1985 & y1990 sum to -0.5
`test y1985=y1990` tests that coefficients on y1985 & y1990 are the same

Presenting the output

STATA presents the output of estimation commands in a format that differs from the one commonly seen in journal articles. Transforming STATA output to publishable format may be very time consuming.

`outreg`, or the updated version `outreg2`, is a command that automates the process of converting the regression results to most of the standard presentation formats by creating a text file for inclusion in a word processing table. The options `word`, `excel`, and `tex` automatically produce the converted files.

Both `outreg` and `outreg2` allow for numerous text-related, coefficient, t-statistics, standard errors, and statistics options.
`outreg2` is though not a standard tool, so you need to install it first. Try typing from within Stata:

```
net install outreg2, fr(http://fmwww.bc.edu/RePEc/bocode/o)
```

Example

The do file:

```
reg pri gdppc
outreg2 using test, tex ct(Primary) lab
reg sec gdppc
outreg2 using test, tex ct(Secondary) lab
reg ter gdppc
outreg2 using test, tex ct(Tertiary) lab
```

produces a TeX fragment for the following table:

Table: Correlation between expenditure per student and GDP, by sector.

	(1) Primary	(2) Secondary	(3) Tertiary
GDP per capita	0.000191***	-0.000953***	-0.00936***
	(1.89e-05)	(0.000130)	(0.000666)
Constant	13.03***	39.30***	224.0***
	(0.227)	(1.509)	(7.834)
Observations	1690	2277	2164
R^2	0.057	0.023	0.084

Standard errors in parentheses
*** $p<0.01$, ** $p<0.05$, * $p<0.1$

Producing a document

LaTeXis the de facto standard for the communication and publication of scientific documents; it includes features designed for the production of technical and scientific documentation.

Stata works relatively well in combination with LaTeX. Plenty of commands exist either inbuilt in Stata (eg. the `estimates` family) or generated by users, that allow to produce TeX fragments, especially regression tables, to be used in typesetting documents.
Check: http://www.ats.ucla.edu/stat/stata/latex/default.htm

Also Stata graphs, saved in `.eps` or `.png` format can normally be included in LaTeX documents.

The normal way to work with Stata and is through do-files that generate results (summary statistics, regression tables and graphs), and then LaTeXcommands to recall them in a document: `\input` for TeX fragments and `\includegraphics[]` for figures.

Statweave

Statweave is a new software, only a beta version for the moment, that allows to combine more directly Stata and LaTeX. A single file, that can be read and edited in any TeX processor but also any word processor (eg. Notepad++), contains text interwoven with code chunks, that call and run Stata from within the typesetting system. The output is a PDF file (or other format) that displays text and, according to the options chosen, Stata code and/or results (tables, graphs or just single statistics).

Example

Look at the file Stata-test.swv. You can open and edit it in TeXnicCenter or Notepad; remember to save it with the extension `.swv`.
Stata-test.pdf is the output file.

Statweave - installation

Statweave is a freeware; it can be downloaded, together with a very short manual and some examples, at
http://www.cs.uiowa.edu/ rlenth/StatWeave/

In the installation dialogue, it is important to specify correctly the location of the `.exe` files for the applications you want to use, in particular Stata (can also be set up for Matlab and R) and the TeX processor (for MiKTeK, it should be the file \miktex\bin\latex.exe). Look for them first in your computer, and take note of their location.

If you experience problems making it work, try the following fixes:

- the configuration file, `statweave.cfg`, is based on MiKTeX 2.6. You can modify it for the version of MiKTeK currently on your computer by opening the file in notepad and replacing the version number.
- a copy of the file `statweave.sty` should always be in the same folder as the `.swv` file you're trying to build.
- create a shortcut of the file `statweavegui` on the desktop, or in an accessible place; drag your .swv file on it in order to compile it (the manual is written for use within DOS environment).

Example

Let's look more in detail at the source file `Stata-test.swv`.

- In the preamble, a special line sets the language specific option, i.e. the options that will be applied to "Stata language". There are options for the formatting of the code and for the formatting of the results. The main ones:
  - `codefmt` alters the formatting so that the code chunk is displayed in blue, and surrounded by a box.
  - `echo` displays the code chunck; preceding the option by a ! is equivalent to set it to false (`echo=FALSE`), suppressing the display of code chunks (try to change)
  - `result=tex` specifies that the output of the code chunk is expected in TeX. The default is verbatim.
  - the option eval can be set to `TRUE` or `FALSE`. In the second case, the code is not run by Stata, but only displayed.
  - `hide` suppresses the display of the output
- The code chunks are delimited by special tags, and can be labeled (to be recalled at later points). Some of the otpions above, and more, can also be specified for single code chunks. For example, options specific for figures (`fig, height, width`).

The software is quite unstable! If you manage to run this file, try removing the lines 55-57.