Methodology Seminars

Introduction to Stata - Class 1 and 2

Maria Perrotta

maria.perrotta@iies.su.se

Fall 2009

Outline of the course

Why Stata

- Stata is the most widely used package for applied econometric work. The current version, Stata11, has been introduced very recently.
- Stata can handle and manipulate very large datasets, good for panel and time-series, flexible graphic capabilities.
- Stata is rather strong in linear and generalized linear models estimation, not so oriented towards nonlinear estimation.
- Check www.stata.com/capabilities/ for what Stata can/can't do.

Finding help

Finding help

- Stata has a very good user-friendly built in manual, accessible:

Finding help

- Stata has a very good user-friendly built in manual, accessible:
  - from within Stata, typing `help` and `search`
  - on the web, at http://www.stata.com/help.cgi?contents

Finding help

- Stata has a very good user-friendly built in manual, accessible:
  - from within Stata, typing `help` and `search`
  - on the web, at http://www.stata.com/help.cgi?contents
- Some useful websites:

Finding help

- Stata has a very good user-friendly built in manual, accessible:
  - from within Stata, typing `help` and `search`
  - on the web, at http://www.stata.com/help.cgi?contents
- Some useful websites:
  - official website: www.stata.com
  - tutorials: www.cpc.unc.edu/services/computer/presentations/statatutorial and www.princeton.edu/ erp/stata/
  - users-written programs (advanced use): ideas.repec.org/s/boc/bocode.html
  - more useful material: www.ats.ucla.edu/stat/stata/sk/

How Stata looks like

- Windows platform (if Stata is installed on local computer)
  - Result window - where results are displayed
  - Command window - where commands are typed
  - Review window, top-left - past commands appear in order; clicking, the command appears in the command window again
  - Variables window, bottom left - all the variables read from the dataset, along with their labels; clicking, the variable will appear in the command window
- On the server, after login type `stata` at the prompt. Only result window.

Interactive and batch mode

Instead of typing commands one-by-one interactively, you can type them all in one go within a do-file and simply run the do-file once.

Advantages of batch mode:

- you remember what you do
- you can run your commands many times with slight changes
- it saves computing time and memory
- especially useful on the server!

The Windows platform has a built-in editor for do-files.

Alternatively, type in notepad (or notepad++) and save with .do extension. To run the do-file, type

```
do "filename.do"
```

or

```
run "filename.do"
```

Do files and log files

Some suggestions

In practice, use interactive analysis to explore the data and try out single commands; as soon as something useful is produced, copy the line in the do-file.
`#review`, PageUp and PageDown can be useful.

Separate do files that create data from do-files that analyze data. Create a separate master do file that does everything.

Open and close log files in your programs.

Write a lot of notes in your do files. Everything you do should be clear, the aim is reproducibility, by yourself and by others.

The no-hassle do file

```
version 9
capture log close
cd "C:\...\dirname"
log using filename, replace
/* What project this is */
/* What this do-file does */
/* Who wrote it */
/* When it was written */
clear
set memory XXm
use datafolder/dataname
*** your commands go here ***
log close
exit
```

cd log commands

Location

- `cd` returns the current location
- It is useful to specify the directory where you want to work, because that's where Stata will look for and save files, without specifying the whole path every time.
- `cd` followed by the directory name in quotation marks, e.g.

```
cd "C:\...\dirname"
```

or, on the server,

```
cd "/home/.../dirname")
```

- `dir` or `dir *.dta` lists the content of the directory or folder, and looks for stata files within the directory
- `mkdir "newdir"` creates a new directory

back

Log files

A log file is the diary of your work session, a text file that records all your commands and all Stata output exactly as it appeared on the results window, from the moment you type

```
log using filename
```

to the moment you type

```
log close
```

If you want to do something off the records:

```
log off
```

and then to go back to recording mode:

```
log on
```

If you want to add something to a closed log:

```
log using filename, append
```

or overwrite it:

```
log using filename, replace
```

Where do we find commands?

- These lecture notes
- `help` and `search` followed by command name or anyword
- Online sources, e.g. www.ats.ucla.edu/stat/examples/default.htm

Reading the data

Data in Stata format (.dta)

The command for this case is `use`:

```
use "datafolder/filename"
```

the `.dta` extension is implicitly assumed.

If the dataset is very big, type the list of variables you want to read in the command:

```
use varname1 varname2 varname3 using datafolder/filename
```

It saves computing time and memory, and helps to have a clearer overview of what is in the data.

If the dataset is from a different version:
`saveold` when working on a newer version
`use10` when working on an older version. !Must be installed (`ssc install`)!

Reading the data

Other formats

- If each data entry is separated by a comma (called the `.csv` format), use `insheet`. This is the format created by spreadsheet or database programs (e.g. Excel). The option `delimit()` allows to specify the separator, if it is a character different from comma.
- If the separator is a tab or a space, use `infile`.
- If the data file is in fixed format (no separator between data entries, entries are identified by fixed width columns), create a dictionary (`.dct` file)

Example

We will use some data files for examples and exercises:

- `icrg.dta` is a panel dataset containing an index of corruption at country level. It is in Stata9 format.
- `khhh31FL.DAT` is an household level survey from Cambodia DHS98, in fixed format with a dictionary.
- `cambhh1.csv` is a subset of the same survey data, in comma separated values.

Reading the data

Some ground rules

When saving a csv- or txt-file for reading into Stata:

- The first line in the spreadsheet should have the variable names, and the second line onwards should have the data. If there is anything else in the top row of the file, then delete this row before saving.
- Any extra lines below the data or to the side of the data (e.g. footnotes) will also be read in by Stata, so make sure that only the data itself is in the spreadsheet before saving.
- The variable names cannot begin with a number. If the file is laid out with years (e.g. 1980, 1985, 1990, 1995) on the top line, place an underscore in front of each number (use the spreadsheet package's "find and replace" tools): 1980 becomes _1980 and so on.
- Make sure there are no commas in the data as it will confuse Stata about where rows and columns start and finish
- In Stata, missing numeric observations are denoted by a single dot (.), missing string observations are denoted by blank double quotes (""). Other notations for missing values can confuse Stata, e.g. it will read double dots (..) or hyphens (-) as text.
    - Option 1: Use find and replace in the original format to replace such symbols with single dots (.) or simply to delete them altogether
    - Option 2: After insheeting, inspect carefully, `replace` the missing values and change the variable to numeric.

If you need to create a dictionary, find an excellent guide at
www.columbia.edu/cu/lweb/indiv/dssc/eds/stata_write.html

Examining the data

Some useful commands:

- About the format of the data: `describe`, `codebook`. Consider carefully the format, especially after importing new data!
- About the content of the data:
    - `summarize`; the option `detail` returns more information
    - `inspect`
    - `table` and `tabulate`; option `missing`
    - `list (in/if)`
    - `assert`
- About the relations between variables: `tabulate twoway`, `correlate` and `pwcorr` (pairwise correlations)

Organizing the dataset

- Have clear names and clearer labels for variables
  - `rename, label data, label variable, label define, label values`
- Select the content of the dataset
  - `keep, drop`
  - logical operators: `==, !=, >, <, >=, <=, &, |`
- Re-arrange variables and observations
  - `order, move, sort`

Examples

```
rename countrycode ccode
label var countrycode "Country ISO-code"

keep if (year>=1990 & year<=1995)
drop if (year<1990 | year>1995)
drop if countrycode!="ITA"

order countrycode countryname
sort -income
```

Combining datasets

Vertically

Vertically: add more observations for the same variables

Example

`oecd.dta` contains observations on the icrg index for 21 OECD countries; `nonoecd.dta` contains observations on the same variable for 54 non-OECD countries.

```
use oecd
append using nonoecd
```

Note: the data in memory (in this case `oecd.dta`) is called the master data, the data that is being added (`nonoecd.dta`) is called the using data.

Combining datasets

Horizontally

Horizontally: add more variables

Example

`data2000.dta` contains observations on the icrg from year 2000, `data1990.dta` contains figures from 1990:

```
use data2000
merge using data1990
```

Notes:
- for this use of `merge`, called one-to-one, you must be sure that master file and using file(s) contain exactly the same observations and in the same order
- a safer option is to merge on a common indicator variable (or set of variables). In this case, both datasets must be previously sorted according to the indicator(s)

Combining datasets

Horizontally

Horizontally: add more variables

Example

`data2000.dta` contains observations on the icrg from year 2000, `data1990.dta` contains figures from 1990:

```
use data2000
merge using data1990
```

Notes:
- for this use of `merge`, called one-to-one, you must be sure that master file and using file(s) contain exactly the same observations and in the same order
- a safer option is to merge on a common indicator variable (or set of variables). In this case, both datasets must be previously sorted according to the indicator(s)

Example

```
use data2000
so countryname
merge countryname using data1990
```

Combining datasets

Horizontally: merge variable

Stata automatically creates a variable called _merge which indicates the results of the merge operation. It is crucial to tabulate this variable to check that the operation worked as you intended. The variable can take on the values:
1 : observations from the master dataset that did not match observations from the using dataset
2 : observations from the using dataset that did not match observations from the master dataset
3 : observations from both datasets that matched
With the option `, update` _merge takes two additional values:
4 : missing values in the master data are replaced with values from the using data.
5 : some values in master data disagree with the values in using data.
If you want to replace the values in the master data with the values in the using data, specify the option `replace` together with `update`

Panel data manipulation

Panel data (=cross-section time-series) are of the form x_{it}.

When we work with panels, we must tell Stata that the dataset is a panel, in order to use commands that are specific for panel (for example, `xtdes`, `xtsum`, `xtreg`, `xtivreg`,...). This can be done with the combination:

```
iis country
tis year
```

or:

```
xtset country year
```

Both the id variables must be numeric.

Panel data manipulation: reshape

Datasets may be laid out in wide or long formats.

Wide form:

country	var1980	var1990
country1		
country2		

Long form:

country	year	var
country1	1980	
country1	1990	
country2	1980	
country2	1990	

`reshape wide var, i(country) j(year)` converts from long to wide.
`reshape long var, i(country) j(year)` converts from wide to long.

The variable(s) following `long` or `wide` contains the data we want to reshape. The i() specifies the variable(s) whose unique values denote a logical observation in wide format. In long format, i() and j() together completely identify each observation.

Creating new variables

gen, egen and replace

The two most common commands for creating new variables are `gen` and `egen`. We can create a host of new variables from the existing data with the gen command, for example:

```
gen lgdp=ln(gdp)
gen gdpsq=gdp^2
gen ten=10
```

The `egen` (extended generate) command creates new variables based on summary statistics, such as sum, mean, min and max:

```
egen totalgdp=sum(gdp), by(year)
egen avggdp=mean(gdp), by(year)
egen maxgdp=max(gdp)
```

The replace command modifies existing variables in exactly the same way as gen creates new variables:

```
gen lgdp=ln(gdp)
replace lgdp=ln(1) if lgdp==.
```

Note: when using `if` statements, remember Stata considers missing values as arbitrarily big.

Creating new variables

String and numeric

String variables can be converted into numeric with `encode`. The labels are automatically assigned, based on the original string values. You can go in the other direction and create a string variable from a numerical one, using `decode`, as long as the numeric variable has labels attached to each value.

However, if a numeric variable is mistakenly read into Stata as a string (that only contains numbers), use instead the function `real`:

```
gen newvar=real(var)
```

Creating new variables

Dummy variables

Dummy variables, or indicator variables, are binary variables that take value 1 if some condition is met, 0 otherwise.
You can use generate and replace to create a dummy variable as follows:

```
gen corrupt=.
replace corrupt=0 if (icrg<3)
replace corrupt=1 if (icrg>=3 & icrg!=.)
```

or:

```
gen corrupt=(icrg>=3 & icrg!=.)
```

Remember again missing values are assumed to be arbitrarily big. In general, it's good practice to always inspect carefully the results after generating new variables.

Creating new variables

Dummy variables

`tabulate` and `xi` are useful to create a set of dummy variables, associated with the values of an existing variable:

Example

```
tab countryname, gen(cdum)
xi i.income
```

Note: such sets of dummies can be recalled in later commands using a wild card, cdum* or _I* respectively, instead of typing out the entire list.

`xi` also creates interaction terms and can be used directly in the syntax of another command.

Example

```
xi i.income*icrg
regress consumption _Iincome* _IincXicrg*
```

or, equivalently:

```
xi:  reg consumption i.income*icrg
```

Creating new variables

Lags and leads

Stata has inbuilt time series operators, which can be used after the data has been declared as having a time series or panel structure (with `tsset` or `xtset`). However, if this is not possible for some reason, `gen` can be used in the following way:

Example

```
so countrycode year
by countrycode:  gen lag=icrg[_n-1] if year==year[_n-1]+1
```

Processing the statement country-by-country is necessary to prevent data from one country being used as a lag for another.
The `if` condition avoids problems when there isn't a full panel of years (if some years are missing).
A lead can be created in similar fashion.

Creating new variables

collapse

`collapse` converts the data into a dataset of summary statistics, such as sums, means, medians, and so on.
Note: Very different from `egen`! `collapse` creates a new dataset, and it will erase the existing one, in particular it will drop the variables that you don't want or can't transform.

Proposed exercises

1. **Make the Cambodia household data in Stata format.**
   - Download the file `cambhh1.csv` to your own folder. Have a look at it in Excel, make changes if needed, and save it in csv format. Open Stata and go to your folder.
   - Choose the appropriate command, `insheet`, `infix`, or `infile`. Read carefully the Stata help for the syntax of the command.
   - Inspect carefully the data in Stata comparing with the original file: are all the variables in the format they are supposed to be (strings are strings and numeric are numeric)?
   - Read the description of the variables in the text file cambhh1.txt. Rename the variables with intuitive and easy-to-remember names (check the Stata help for rules on variable names), and give them clear labels.
   - Label the data so you remember the content. Save the new data file in Stata9 format.
2. **New variables**
   - Use the file you created at point 1. Create numeric variables for the binary variables (questions with yes/no answer).
   - How would you define relatively poor and relatively rich households? Create two dummies for rich and poor households.
   - Create a new dataset at the village level rather than household level, containing the share of rich and poor household in each village (Hint: use `collapse`).
3. **Combining files**
   Combine the corruption index with information on world region and income group the countries belong to (i.e., `icrg.dta` and `inc_reg.dta`). What is the appropriate command?