Introduction to Stata

Lecture X

Tomas R. Martinez

UC3M

September, 2019

Macros, loops and tips about coding

- “Even if you dont intend anybody else to read your code, theres still a very good chance that somebody will have to stare at your code and figure out what it does: That person is probably going to be you, twelve months from now.” - Raymond Chen

Macros

- Local macros is a “variable” that stays in the memory of you do-file as long it is running
- It can be pretty much anything
- First you define it: **local lclname [=expression / text / list]**
- Then you call it by putting between: \`lclname'
- **Example:** varlist as local
  - local myvariables educ incwage sex
  - summarize \`myvariables'
  - describe \`myvariables'

Macros

- **Example:** Sample selection as local macro
  - local sample1 if sex==1 & age >=18 & age <=65
  - summarize incwage `sample1'
  - reg incwage age age2 `sample1'
- If you have to calculate a bunch of statistics using a macro can save a lot of work, and it is much easier to change!
- **Example:** Set directory as a macro
  - local path c:\user\stata\course\
  - use "`sample1'mydata.dta", clear
  - save "`sample1'mydata.dta", replace

Macros

- Careful when you set text and numbers!
- local number 3+3 $\rightarrow$ the macro `number' refers to "3 + 3"
- local number = 3+3 $\rightarrow$ the macro `number' refers to "6"

Macros

- Global macros, differently than local, are persistent
- **Syntax:** global macroname [=expression / text / list] (pretty much the same as local)
- But you call a global macro with $: *$macroname*
- **CAREFUL:** If define a global macro in a do-file it will be carried on even in other do-files!
- Global macros can cause conflicts across different programs, thus local macros are preferable
- Personally, I only use to define directory across do-files

Loops

- A loop is a way to repeat the same command multiple time, saving space and time
- A more “general” loop: **foreach**
- Specific to numeric: **forvalues**
- Loop until a certain condition is met: **while**
- Why would you need a loop in Stata:
    - Creating interactions
    - Fitting multiple models (e.g. one regression for each occupation)
    - Recoding many variables in the same way
    - Opening, modifying and saving multiple data set

Loops

- A **foreach** loop can take strings, list of variables...
- Example

```
foreach varname in incwage inctotal incbus {
gen log_‘varname’=log(‘varname’)
}
```

- In this case *varname* is a local macro which is going to disappear after the loop is done

Loops

- A **forvalues** loop takes list of number
- Examples:

```
foreach i =1/4 {
sum incwage if educ==‘i’
}

foreach i =0(5)45 {
di "value: ‘i’"
}
```

Loops

- A **while** runs until a condition is met
- Useful when you don't know when exactly the iteration should stop
- Maybe it does not depend on a number, but a certain condition of your data

```
while [condition] {
[do something]
}
```

If clauses

- The **if** clause allows Stata to execute a command only if certain condition holds
- It can be extended with multiple conditions using **else if**
- It checks the condition on “cascade”: Check the initial condition, if not satisfied go to the next clause

```
if [condition] {
[do something]
}
else {
[do something else]
}
```

Organizing your project

- When you have a project very often you have multiple data sets and do-files
- How to maintain that whole thing organized?
- All the data and do-file should be in one folder
- Create a subfolder for the output: graphs, regression tables and etc
- Have a main do-file that calls all the other do-files
- Enumerate your do-files: 00_main.do, 01_merge.do, 02_sample.do, 03_facts.do, 04_regressions.do...

Organizing your project

- Your do-files should be as commented as possible
- Give explicit variable names; drop temporary variables
- Your loops should be indented
- If you have a long code it is useful to temporally change the delimiter **# delimit ;**
- Use the command **quietly** before others to suppress the output in the screen
- If your do-file produces output, it is useful to have a local macro stating the version: **local version v1** $\rightarrow$ then include in the name of your output: graph_`version'.png
- Now if you change something but still want to keep track of old results change *v1* to *v2*