Econometric Softwares: Stata

Jérémy Do Nascimento Miguel

Class 6 Spring 2024

Outline

- Macros, Loops and Conditions
- Matrices and scalars
- Writing programs/commands
- Data management practices in economic research

Outline

Macros, loops, and conditions

Matrices and scalars

Local and global macros

- A macro is a string of characters (macro name) which is an alias for another string of characters (content of a macro)
- Macros can be local or global:
  - → A global macro is accessible across Stata do-files or command executions within a Stata session
  - → A local macro is accessible only within a do-file, or within a single execution of a do-file or commands.
- Global macro:
  - → Syntax: global globalname [=] exp | "text" | `"text"'
  - → Accessed by: $globalname or ${globalname}
- Local macro:
  - → Syntax: local localname [=] exp | "text" | `"text"'
  - → Accessed by: `localname'

Local and global macros: examples

local/global macro to store variable list:

- local controlvars "age gender educ"
- global hhcontrolvars "hhincome urban"
- regress income ‘controlvars’ $hhcontrolvars
  all control variables are added by local ‘controlvars’ and global $hhcontrolvars

local macro to store numeric values:

- local meanincome = 1590
- local SDincome = 781
- gen incomestd = (income - ‘meanincome’)/ ‘SDincome’
  here using global instead of local will do exact the same job

Loops

- If you need to repeat the same set of commands for different values, it is more efficient to use loops to iterate commands across values
- Three commands in Stata to construct a loop:
- forvalues: loop over numeric values
- foreach: loop over any sequences, text or numeric
- while: iterate until condition is no longer met
- Possible to nest loops within loops (e.g. a sub-loop will run for each value specified in a master loop)

Loops: forvalues

Syntax: forvalues localname = range {
commands that refer to 'localname'
}

For instance: forvalues i = 1/5 {
summarize age income if educ=='i'
}

- Here a **local** i serves as the loop index, and range is given by **numlist** 1/5 that takes values from 1 to 5, with the step of 1
- numlist can also be in descending order: 2/-2 → 2 1 0 -1 -2
- Possible to specify 'steps' in **numlist**: 1(5)100 → 1 to 100 with the step of 5

Loops: foreach

foreach is more general command for loops, can take any arbitrary list of elements (text or numeric)

Syntax: foreach localname in anylist {

commands that refer to 'localname'

}

For instance: foreach x in age educ income urban {

display "`x'"

summarize 'x'

}

Loops: while

whileis relatively less popular and needs condition to check every time it iterates and a parameter that changes its value in each iteration

Syntax: while exp {

command

}

For instance: local i = 1

while ‘i’<=10 {

sum y if x==‘i’

local i = ‘i’+1

}

Exercise 6.1

1. Open data hh comm.dta.
2. Define global named sumvars which contains following set of variables: poor expoor ppcd, nmigrant, empratiom, hhh female land
3. Run commands describe and summarize for this global sumvars
4. Design a loop for values from 0 to 6 (using forvalues). Within the loop, generate a dummy variable which identifies households by number of children from 0 to 6. When loop is finished to run, you should have seven dummy variables: nchild0, nchild1,...nchild6

Outline

Macros, loops, and conditions

Matrices and scalars

Scalars

- scalar can store a numeric value or a string
- Syntax for generating: scalar scalarname = exp

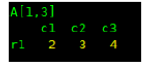
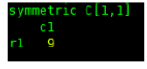
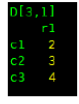
  scalar a = 4

  scalar b = "I am a scalar, my name is b"
- To print the content of a scalar: display scalarname
- To delete scalar from memory: scalar drop scalarname
- To list all scalars in memory: scalar dir

Matrices

- matrix in Stata can store a set of only numeric values in a row (vector), in a column (vector) or in a row X column array (matrix), bordered with names of rows and columns
- Syntax for generating: matrix matrixname = (row1col1, row1col2 row2col1, row2col2) , separates columns and \ separate rows
- To print the content of a matrix: matrix list matrixname
- To delete matrix from memory: matrix drop matrixname
- To list all matrices in memory: matrix dir

Matrices: examples

Vector / Matrix	Stata command	Stata Output (`matrix list`)
(2,3,4)	`mat A = (2,3,4)`	
Unit vector: $\begin{pmatrix}1\\1\\1\end{pmatrix}$	`mat B = J(3,1,1)`	
Matrix multiplication A x B	`mat C = A * B`	
Matrix transposing	`mat D = A'`	

Creating new stata commands

Before implementing a new command use findit to check for user-defined programs

- A program is defined by:

  program *progname*

  statacommands

  end

- Before a program can be redefined, it has to be dropped from memory first. Therefore, it is useful to include in your do-file first: capture program drop *progname*

Creating new stata commands

For example, write a program that calculates the interquartile range.

```
* Program that calculates the interquartile range.
capture program drop iqr
program iqr
    syntax varlist // no options etc. yet allowed ...
    summarize `varlist', detail
    scalar iqr=r(p75)-r(p25)
    display "Inter quartile range is " iqr
end

* Test your program:
sysuse auto.dta, clear
iqr mpg
```

Creating new stata commands

Testing it

```
. sysuse auto.dta, clear
(1978 Automobile Data)

. iqr mpg

                          Mileage (mpg)
-------------------------------------------------------------
      Percentiles      Smallest
 1%           12              12
 5%           14              12
10%           14              14       Obs                  74
25%           18              14       Sum of Wgt.          74

50%           20                       Mean            21.2973
                      Largest          Std. Dev.       5.785503
75%           25              34
90%           29              35       Variance        33.47205
95%           34              35       Skewness        .9487176
99%           41              41       Kurtosis        3.975005
Inter quartile range is 7
```

Creating new stata commands: debugging

Consider potential mistakes and try them out

- Display intermediate output, i.e. simplify your program and run code step by step without program command etc. (instead create macros and tempvars), include display, list, matrix list etc. where appropriate.
- Comment out critical commands.
- Pay particular attention to matrix dimensions and macros.
- Use set trace on and set trace off.

Creating new stata commands: debugging

- E.g. calculating an IQR doesn't make sense for string variable
- We need to accept only one variable to calculate an IQR
- **Can we restrict the program to single numeric variables only?**
- try help syntax to learn about setting syntax rules
- min and max to set the minimum and maximum number of variables that may be specified
- numeric, string, ... to restrict the specified varlist to consist of entirely numeric, entirely string
- (!) Include the return-command so that output of the command is accessible.

Creating new stata commands: debugging

Testing it

```
* Fixed version:
capture program drop iqr
program iqr
    syntax varlist(max=1 numeric) //only one numeric variable allowed
    quietly summarize `varlist', detail //quietly hides undesired output
    scalar iqr=r(p75)-r(p25)
    display "Inter quartile range is " iqr
    return scalar iqr=iqr // stores iqr as scalar in r-class => accessible after command
end

* Test the program:
iqr mpg price //this will give an error now

iqr mpg
return list //this will show which results are accessible after iqr command
```

Exercise 6.2

Create a command semean that computes the standard error of the mean.

1. Using auto.dta (sysuse auto), use returned results of summarize and display the standard error of the mean of price.
2. Define program semean that allows only one variable as argument and display the results in the form "Mean of x=" $\bar{x}$ " S.E. =" $\hat{\sigma(x)}$ ", where x is the variable name, $\bar{x}$ and $\hat{\sigma(x)}$ are the mean and its standard error. Make sure that the table created by summarize is not shown.
3. Modify such that program returns the mean and its std. error as scalars.

Organization of work

Why do we need efficient “organization of work”?

- Working with many files
- Easy to lose general overview
- Important for replication, especially years later

Organizaiton of work: a general example

Data

- **Original data**: raw data, **never edited (!)** and **well-protected (!)**
- **Working data**: data processed for analysis, often changed

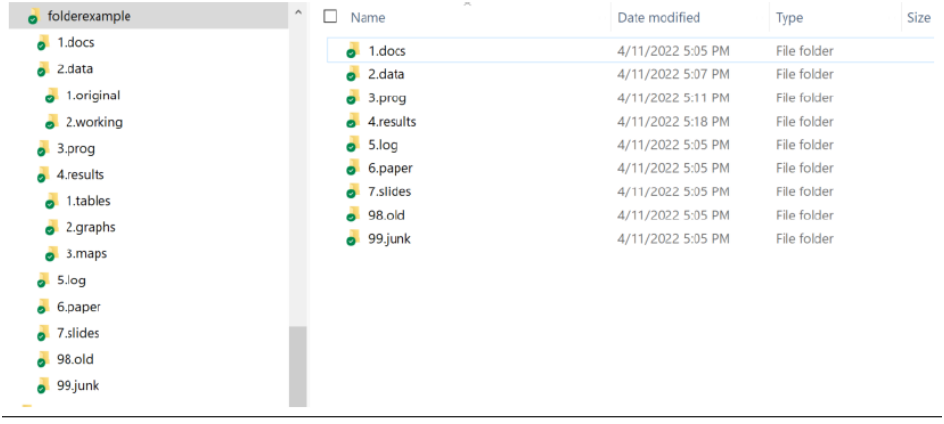
Processing

- **Do-files**: clean, ordered (i.e. numbering do-files), well-documented with comments, a master do-file if needed

Output

- **Tables**: clean, with explanatory notes, clear variable labels, a master table combining all results if needed
- **Graphs**: clean, explanatory notes, clear labels and scaling of axis, contrasted colors ("warm" vs. "cold" colors, e.g. red vs. blue)
- **Log-file**: named with current date to keep track of the whole process

Organizaiton of work: folder structure



Defining directories

- Commands to check current directories: cd or pwd
- You can change the current directory by typing: cd "somefolderpath "
- (!) Note that it is better to round folder path with double quotes (")
- More efficient is to use global macros to pre-define folder paths, e.g.: global myfolder "C:\Users\jdnmiguel\Dropbox\folder"
  cd $myfolder
- An alternative (and probably more efficient way) is to directly refer to folder path using globals rather than each time defining directory using cd
- e.g. to open data:

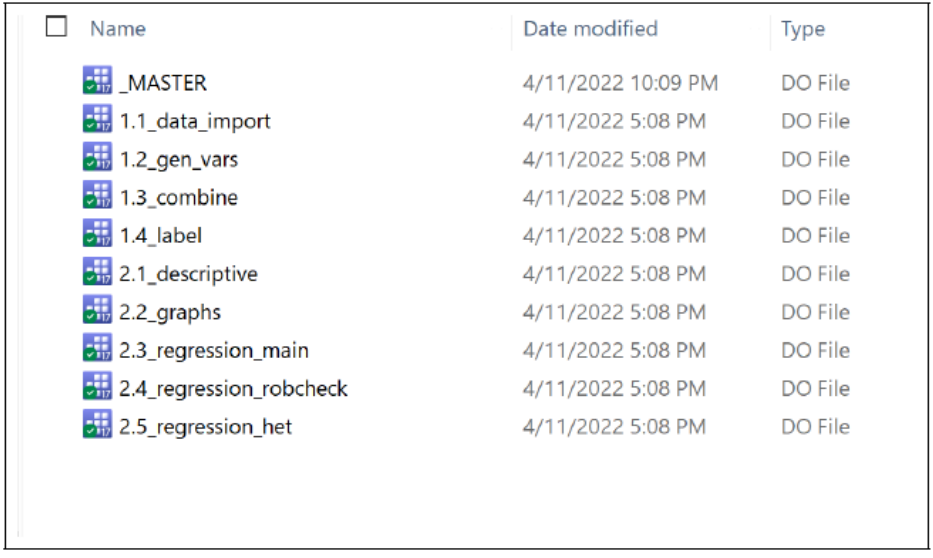
```
global myfolder "C:\Users\jdnmiguel\Dropbox\folder"
use "myfolder\somedata.dta", clear
```

Structure of do-files

General suggestions to organize do-files (but this is not strict, everyone is flexible):

1. Split do-files by topic, many do-files (all numbered by order), master do-file that runs all other do-files to produce final outputs
   Pros: structured, short do-files
   Cons: easy to get lost, difficult to find necessary line of code if many do-files, slightly difficult to control what to run and what not to run
2. Single do-file for data preparation, and single do-file for analysis, with many subsections
   Pros: structured, all-in-one place, easier to control what to run and what not to run, probably better format for journal submissions
   Cons: very long do-files

Structure of do-files: method 1



Name	Date modified	Type
_MASTER	4/11/2022 10:09 PM	DO File
1.1_data_import	4/11/2022 5:08 PM	DO File
1.2_gen_vars	4/11/2022 5:08 PM	DO File
1.3_combine	4/11/2022 5:08 PM	DO File
1.4_label	4/11/2022 5:08 PM	DO File
2.1_descriptive	4/11/2022 5:08 PM	DO File
2.2_graphs	4/11/2022 5:08 PM	DO File
2.3_regression_main	4/11/2022 5:08 PM	DO File
2.4_regression_robcheck	4/11/2022 5:08 PM	DO File
2.5_regression_het	4/11/2022 5:08 PM	DO File

Structure of do-files: method 1

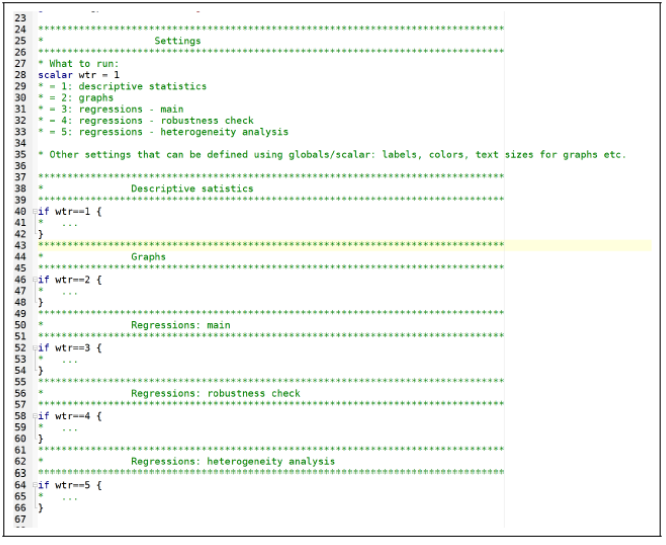
```
24 ********************************************************************************
25 *                   Data Preparation
26 ********************************************************************************
27 log using "$logpath\dataprep.log", replace
28
29 * Import data
30 do "$dopath/1.1_data_import.do"
31 * Generate variables
32 do "$dopath/1.2_gen_vars.do"
33 * Combine
34 do "$dopath/1.3_combine.do"
35 * Label final set of variables
36 do "$dopath/1.4_label.do"
37
38 log close
39
40 ********************************************************************************
41 *                   Analysis
42 ********************************************************************************
43 log using "$logpath\analysis.log", replace
44
45 * Descriptive statistics
46 do "$dopath/2.1_descriptive.do"
47 * Graphs
48 do "$dopath/2.2_graphs.do"
49 * Regressions: Main
50 do "$dopath/2.3_regression_main.do"
51 * Regressions: Robustness checks
52 do "$dopath/2.4_regression_robcheck.do"
53 * Regressions: Heterogeneity analysis
54 do "$dopath/2.5_regression_het.do"
55
56 log close
57
```

Structure of do-files: method 2

Structure of do-files: method 2

```
24  ********************************************************************************
25  *                    Data preperation
26  ********************************************************************************
27  * Import data
28  * ......
29
30  * Generate and rename variables
31  * ......
32
33  * Label variables
34  * ......
35
36  * Combine and save final data
37  * ......
38
```

Structure of do-files: method 2



```
23
24  ********************************************************************************
25  *                      Settings
26  ********************************************************************************
27  * What to run:
28  scalar wtr = 1
29  * = 1: descriptive statistics
30  * = 2: graphs
31  * = 3: regressions - main
32  * = 4: regressions - robustness check
33  * = 5: regressions - heterogeneity analysis
34
35  * Other settings that can be defined using globals/scalar: labels, colors, text sizes for graphs etc.
36
37  ********************************************************************************
38  *                  Descriptive satistics
39  ********************************************************************************
40  if wtr==1 {
41  *   ...
42  }
43  ********************************************************************************
44  *                  Graphs
45  ********************************************************************************
46  if wtr==2 {
47  *   ...
48  }
49  ********************************************************************************
50  *                  Regressions: main
51  ********************************************************************************
52  if wtr==3 {
53  *   ...
54  }
55  ********************************************************************************
56  *                  Regressions: robustness check
57  ********************************************************************************
58  if wtr==4 {
59  *   ...
60  }
61  ********************************************************************************
62  *                  Regressions: heterogeneity analysis
63  ********************************************************************************
64  if wtr==5 {
65  *   ...
66  }
67
```