

Introduction to Macro and Loop in Stata

SOC 561

Programming for the Social Sciences

Hyungjun Suh

Mar. 7. 2016



Overview

- ▶ Macros (local and global macros)
- ▶ Loops (foreach and forvalues)
- ▶ Tempvar
- ▶ Tempfile

Macro: What is it and what does it do?

- ▶ Macros are abbreviations for a string of characters or a number (Long 2009:83). They also represent expressions, so as to make programming efficient.

3

Local and Global Macro

- ▶ **Local macros** are ephemeral macros. When you work with do-file editor, local macros are valid only in a single execution of commands in do-files or ado-files. Yet, when you work interactively, they persist until you delete them or the session is ended.
- ▶ **Global macros** are persisting macros. Once defined whether in do-file editor or in interactive command window, they persist until you delete them or the session is ended.

4

Local and Global Macro

- ▶ In general, using local macros are recommended. This is because when you work with many do-files or ado-files at the same time, global macros can cause conflicts across do-files or ado-files, which is error-prone.

5

Syntax of Macro Assignment

```
local macroname "string"  
local macroname = expression  
global macroname "string"  
global macroname = expression
```

6

Using Macros

back quote single quote

↓ ↓

local : `macroname`
 global: \$macroname

7

Example

It works without
quotation marks

```
. local y "female age income" ← Assignment
. display "`y'" ← Usage
female age income
```

8

Example

```
. global xyz "female age income"  
  
. display "$xyz"  
female age income
```

9

Example

```
. local y2 = 345+123  
  
. display `y2'  
468
```

10

Example

```
. sysuse nlsw88, clear
(NLSW, 1988 extract)

. local myvars "age race married grade south smsa"

. summarize `myvars'
```

This way, you can truncate a long list of variable, which helps you debug.

Variable	Obs	Mean	Std. Dev.	Min	Max
age	2,246	39.15316	3.060002	34	46
race	2,246	1.282725	.4754413	1	3
married	2,246	.6420303	.4795099	0	1
grade	2,244	13.09893	2.521246	0	18
south	2,246	.4194123	.4935728	0	1
smsa	2,246	.7039181	.4566292	0	1

11

Example

```
. local model_1 "age race"
. local model_2 "married grade"
. regress wage `model_1'
```

You can group different variables to different macros, which can reduce mistakes.

Source	SS	df	MS	Number of obs	=	2,246
Model	611.586289	2	305.793145	F(2, 2243)	=	9.30
Residual	73756.3811	2,243	32.8829162	Prob > F	=	0.0001
Total	74367.9674	2,245	33.1260434	R-squared	=	0.0082
				Adj R-squared	=	0.0073
				Root MSE	=	5.7344

wage	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
age	-.0771988	.0396188	-1.95	0.051	-.1548921 .0004945
race	-1.008454	.2549917	-3.95	0.000	-1.508498 -.5084093
_cons	12.08309	1.608499	7.51	0.000	8.928792 15.2374

12

Example

```
. regress wage `model_1' `model_2'
```

Source	SS	df	MS	Number of obs	=	2,244
Model	8251.11998	4	2062.78	F(4, 2239)	=	69.87
Residual	66103.2105	2,239	29.523542	Prob > F	=	0.0000
Total	74354.3305	2,243	33.1495009	R-squared	=	0.1110
				Adj R-squared	=	0.1094
				Root MSE	=	5.4336

wage	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
age	-.0551124	.037593	-1.47	0.143	-.1288331 .0186084
race	-.6067613	.2487806	-2.44	0.015	-1.094626 -.1188966
married	-.6599472	.2439325	-2.71	0.007	-1.138305 -.1815896
grade	.7269462	.045974	15.81	0.000	.6367901 .8171023
_cons	1.606477	1.701157	0.94	0.345	-1.729533 4.942488

13

Example

- ▶ Macros can be used when you want to specify long options as well (Long 2009:88-89).
- ▶ e.g.) local opt_tab “cell miss nolabel chi2”
 - tabulate south smsa, `opt_tab’
 - local opt_linF “lpattern(solid) lwidth(medthick) lcolor(black) msymbol(i)”
 - graph twoway (connected wage hours, `opt_linF’)

14

Loop: What is it and what does it do?

- ▶ Loops refer to commands which execute a group of commands multiple times (Long 2009:92). Think about the situation where you should replace some values of 1,000 variables the same way. You can do it manually, but it takes a lot of time and it is error-prone. Also, it can be the case that you need to do it again for some reasons, like false calculation or different theoretical consideration. Loops are useful in situations above.

15

Loop: What is it and what does it do?

- ▶ Loops are useful in following tasks (Long 2009:95-96):
 - Listing variable and value labels
 - Creating interaction variables
 - Fitting models with alternative measures of education
 - Recoding multiple variables the same way
 - Creating a macro that holds accumulated information
 - Retrieving information returned by Stata

16

foreach and forvalues

- ▶ Foreach is a more general loop. String, numeric, and variables are allowed as list, and lists do not have to have a pattern.
- ▶ Forvalues is a more specific loop. Only numeric is allowed as lists, and lists should have a clear pattern.

17

Syntax of `foreach (in)` command

```
foreach macroname in list {  
    commands referring to `macroname`  
}
```

18

Syntax of `foreach` (of) command

```
foreach macroname of list-type {  
    commands referring to `macroname`  
}
```

19

Syntax of `foreach` (of) command

List-types are

- Local
- Global
- Varlist: you should list variables
- Newlist: you should list the name of new variables
- Numlist: you should list numbers

20

Syntax of `forvalues` command

```
forvalues macroname = range {  
    commands referring to `macroname`  
}
```

21

Syntax of `forvalues` command

Range can have three forms (Long 2009:95):

- `#1(#d)#2`: from #1 to #2 in steps of #d
e.g.) `1(2)10` -> 1, 3, 5, 7, 9
- `#1/#2`: from #1 to #2 in steps of 1
e.g.) `1/10` -> 1, 2, 3, 4, ..., 10
- `#1 #t to #2`: from #1 to #2 in steps of (#t-#1)
e.g.) `1 3 to 10` -> 1, 3, 5, 7, 9

22

Levelsof command

- ▶ Levelsof command identify all values in a variable and put those values in a macro
(http://www.ssc.wisc.edu/sscc/pubs/stata_prog1.htm).
- ▶ Syntax: levelsof variable, local (macro)
- ▶ It is useful when the variable of interest has many values, like hundreds or thousands values.

23

Example

```
. sysuse nlsw88, clear
(NLSW, 1988 extract)
```

```
. tab1 race
```

```
-> tabulation of race
```

race	Freq.	Percent	Cum.
white	1,637	72.89	72.89
black	583	25.96	98.84
other	26	1.16	100.00
Total	2,246	100.00	

24

Example

```
. * foreach (in) example
. foreach i in 1 2 3 {
2.     summarize wage if race=='i'
3. }
```

Macroname.

It can be
anything.

You assign it.

list

command

macroname

25

Example

```
. * foreach (in) example
. foreach i in 1 2 3 {
2.     summarize wage if race=='i'
3. }
```

Variable	Obs	Mean	Std. Dev.	Min	Max
wage	1,637	8.082999	5.955069	1.004952	40.19808
Variable	Obs	Mean	Std. Dev.	Min	Max
wage	583	6.844558	5.076187	1.151368	40.74659
Variable	Obs	Mean	Std. Dev.	Min	Max
wage	26	8.550781	5.20943	1.80602	25.80515

26

Example

```
. * levelsof example
. levelsof race, local(races)
1 2 3
. foreach i in `races' {
2.     summarize wage if race==`i'
3. }
```

Alternatively, we can use levelsof command to specify values in race variable

Variable	Obs	Mean	Std. Dev.	Min	Max
wage	1,637	8.082999	5.955069	1.004952	40.19808
Variable	Obs	Mean	Std. Dev.	Min	Max
wage	583	6.844558	5.076187	1.151368	40.74659
Variable	Obs	Mean	Std. Dev.	Min	Max
wage	26	8.550781	5.20943	1.80602	25.80515

27

Example

```
. * foreach (in) example 2
. foreach i in south smsa union {
2.     tabulate `i', missing
3. }
```

List can contain variables, string, and numeric.

lives in south	Freq.	Percent	Cum.
0	1,304	58.06	58.06
1	942	41.94	100.00
Total	2,246	100.00	
lives in SMSA	Freq.	Percent	Cum.
nonSMSA	665	29.61	29.61
SMSA	1,581	70.39	100.00
Total	2,246	100.00	
union worker	Freq.	Percent	Cum.
nonunion	1,417	63.09	63.09
union	461	20.53	83.62
.	368	16.38	100.00
Total	2,246	100.00	

28

Example

```

. * foreach (in) example - trace
. foreach i in south smsa union {
2.     summarize `i'
3. }
- foreach i in south smsa union {
- summarize `i'
= summarize south

```

Variable	Obs	Mean	Std. Dev.	Min	Max
south	2,246	.4194123	.4935728	0	1

```

- }
- summarize `i'
= summarize smsa

```

Variable	Obs	Mean	Std. Dev.	Min	Max
smsa	2,246	.7039181	.4566292	0	1

```

- }
- summarize `i'
= summarize union

```

Variable	Obs	Mean	Std. Dev.	Min	Max
union	1,878	.2454739	.4304825	0	1

```

- }

```

Loops execute commands from the top to the bottom, and execute them again from the beginning.

29

Example

```

. * foreach (of) example
. foreach i of numlist 1 2 3 {
2.     summarize wage if race==`i'
3. }

```

Variable	Obs	Mean	Std. Dev.	Min	Max
wage	1,637	8.082999	5.955069	1.004952	40.19808

Variable	Obs	Mean	Std. Dev.	Min	Max
wage	583	6.844558	5.076187	1.151368	40.74659

Variable	Obs	Mean	Std. Dev.	Min	Max
wage	26	8.550781	5.20943	1.80602	25.80515

30

Example

```
. * forvalues example
. forvalues i = 1(1)3 {
2.     summarize wage if race==`i'
3. }
```

Variable	Obs	Mean	Std. Dev.	Min	Max
wage	1,637	8.082999	5.955069	1.004952	40.19808
Variable	Obs	Mean	Std. Dev.	Min	Max
wage	583	6.844558	5.076187	1.151368	40.74659
Variable	Obs	Mean	Std. Dev.	Min	Max
wage	26	8.550781	5.20943	1.80602	25.80515

31

Example

```
. * loop example 1
. * listing variable and value labels
. foreach i in married south smsa {
2.     local varlabel : variable label `i'
3.     display "`i'" _col(12) "`varlabel'"
4. }
```

married married
south lives in south
smsa lives in SMSA

This way, you can check as many variables' name and label as what you want at once.

32

Example

```
. * loop example 2
. * creating interaction variables
. foreach i in married south smsa {
2.     generate          gradX`i'=collgrad*`i'
3.     label variable  gradX`i' "collgrad*`i'"
4. }

. summarize gradXmarried gradXsouth gradXsmsa
```

Variable	Obs	Mean	Std. Dev.	Min	Max
gradXmarried	2,246	.1531612	.360223	0	1
gradXsouth	2,246	.0970614	.2961073	0	1
gradXsmsa	2,246	.1856634	.3889214	0	1

You can quickly make variables with a loop.

33

Example

```
.
. * loop example 3
. * fitting different models at once
. foreach i in 1 2 3 {
2.     regress wage grade if race==`i'
3. }
```

Results are omitted. A researcher can see difference effects of grade on wage by race quickly. For example, if you want to see different effects of a certain variable on the dependent variable in 100 countries, loops would be helpful.

34

Example

```
. * loop example 4
. * recoding multiple variables the same way
. generate meanwage=.
(2,246 missing values generated)

. forvalues k=34(1)46 {
2.     summarize wage if age==`k'
3.     replace meanwage=r(mean) if age==`k'
4. }
```

See the next slide
for the explanation

```
. tabulate meanwage, missing
```

meanwage	Freq.	Percent	Cum.
6.815027	53	2.36	2.36
7.169666	163	7.26	9.62
7.306658	78	3.47	13.09
7.333253	160	7.12	20.21
7.522215	222	9.88	30.10
7.675594	165	7.35	37.44
7.680887	208	9.26	46.71
7.884183	225	10.02	56.72
7.990341	234	10.42	67.14
8.048292	260	11.58	78.72
8.115071	219	9.75	88.47
8.136585	257	11.44	99.91
16.45329	2	0.09	100.00
Total	2,246	100.00	

35

Example

- ▶ This command tries to get mean wages by respondents' age and assign the derived value to each respondent.
- ▶ Alternatively, we can consider the following command:


```
generate meanwage=.
summarize wage if age==34
replace meanwage=r(mean) if age==34
```

 (same commands for all age values from 35 to 46)
- ▶ Forvalues loop is more efficient than the commands above.

36

Example

```
. * counters in loops
. local counter = 0

. foreach i in married south smsa {
2.     local counter = `counter'+1
3.     local varlabel : variable label `i'
4.     display "`counter'. `i'" _col(12) "`varlabel'"
5. }
1. married married
2. south   lives in south
3. smsa   lives in SMSA
```

If you want to assign numbers to each result, using counters would be helpful.

37

Example

```
. * using loops to save results to a matrix
. * make 3x2 matrix called 'res', having missing values
. matrix res = J(3, 2, .)

. * assign column and row names
. * 1st column = coefficient of grade
. * 2nd column = standard error of grade
. matrix colnames res = coeff se

. matrix rownames res = white black other
```

Loops allow a researcher to save results for future analysis. Suppose we want to see different effect of grade on wage by race. First, I make a matrix as above in which results will be saved.

38

Example

```
. local irow = 0
```

```
. foreach i in 1 2 3 {
```

```
2.     local ++irow ←
```

```
3.     quietly regress wage grade if race==`i'
```

```
4.     matrix res[`irow',1] = _b[grade]
```

```
5.     matrix res[`irow',2] = _se[grade]
```

```
6. }
```

```
. matrix list res
```

```
res[3,2]
```

	coeff	se
white	.68035405	.05892988
black	.83147089	.0716403
other	.76242972	.21754353

In this way, we can make a graph of coefficients standard errors or further analysis.

39

Tempvar: What is it and what does it do?

- ▶ Tempvar creates a temporary variable which is valid only in a single execution of commands in do-files or ado-files.
- ▶ Alternatively, we can make variables and then erase them manually. Yet, using tempvar is more efficient.
- ▶ Syntax step1: tempvar var1 (declaring variable name)
- ▶ Step2: commands referring to `var1'

40

Example

```
. tempvar myvar // declare a temporary variable name.
. egen `myvar' = mean(wage)
. generate wageR = wage/`myvar'
```

Execution ends here.

```
. summarize wageR `myvar'
```

Variable	Obs	Mean	Std. Dev.	Min	Max
wageR	2,246	1	.7410275	.1293882	5.246151

'myvar' has been deleted automatically.

41

Tempfile: What is it and what does it do?

- ▶ Tempfile creates a temporary dataset which is valid only in a single execution of commands in do-files or ado-files.
- ▶ Alternatively, we can save a dataset and use it, and erase it. Compared to this, tempfile is more efficient.
- ▶ Or, one can consider preserve/restore commands, but preserve command stores only one dataset.
- ▶ Syntax step1: tempfile data1 (declaring data name)
- ▶ Step2: commands referring to `data1'

42

Example

```
. sysuse nlsw88, clear
(NLSW, 1988 extract)

. foreach i in married south smsa {
2.     generate          gradX`i'=collgrad*`i'
3.     label variable  gradX`i' "collgrad*`i'"
4. }

. summarize gradXmarried gradXsouth gradXsmsa
```

Variable	Obs	Mean	Std. Dev.	Min	Max
gradXmarried	2,246	.1531612	.360223	0	1
gradXsouth	2,246	.0970614	.2961073	0	1
gradXsmsa	2,246	.1856634	.3889214	0	1

43

Example

```
. tempfile mydata // declare a temporary file name.

. save `mydata'
file C:\Users\HYUNGJUN\AppData\Local\Temp\ST_01000001.tmp saved

. clear

. use `mydata'
(NLSW, 1988 extract)

. describe gradXmarried gradXsouth gradXsmsa
```

variable name	storage type	display format	value label	variable label
gradXmarried	float	%9.0g		collgrad*married
gradXsouth	float	%9.0g		collgrad*south
gradXsmsa	float	%9.0g		collgrad*smsa

44

Sources

- ▶ StataCorp. Stata Programming Reference Manual Release 14. College Station, TX: Stata Press.
- ▶ Long, J. Scott. 2009. The Workflow of Data Analysis Using Stata. College Station, TX: Stata Press.
- ▶ Macros. http://www.ssc.wisc.edu/sscc/pubs/stata_prog1.htm
- ▶ “B] macros”.
<http://pierrefrancois.wifeo.com/documents/Intro-Stata---LSE-III.pdf>
- ▶ Tempfiles. <http://www.stata.com/statalist/archive/2004-01/msg00542.html>

45

- ▶ Questions and Comments to
suhhyungjun@email.arizona.edu

46

תודה
 Dankie Gracias
 Спасибо شكراً
 Köszönjük Merci Takk
 Grazie Dziękujemy Terima kasih
 Ďakujeme Vielen Dank Dėkojame
 Kiitos Täname teid 谢谢
Thank You Tak
 感謝您 Obrigado Teşekkür Ederiz
 Σας Ευχαριστούμ 감사합니다
 Bedankt Дėкујеме vám
 ありがとうございます
 Tack