KAIST Summer Session 2018

Module 2. Causal Inference with STATA

STATA Programming

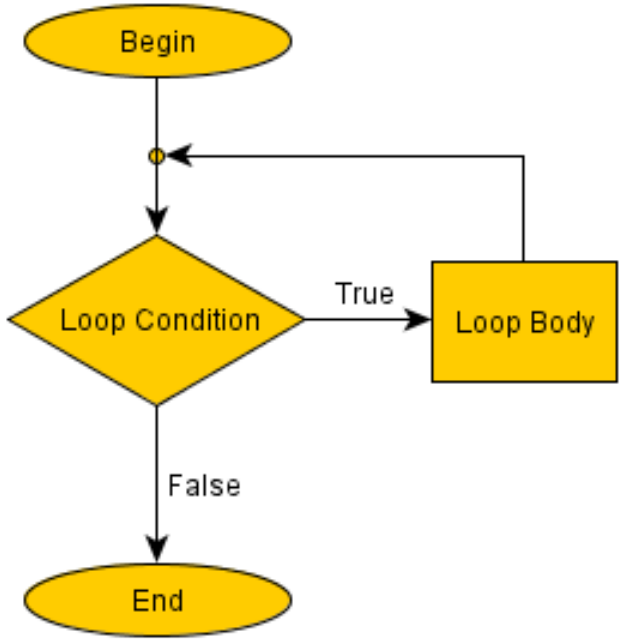
KAIST College of Business

Jiyong Park

2 August, 2018

Why Do You Need STATA Programming?

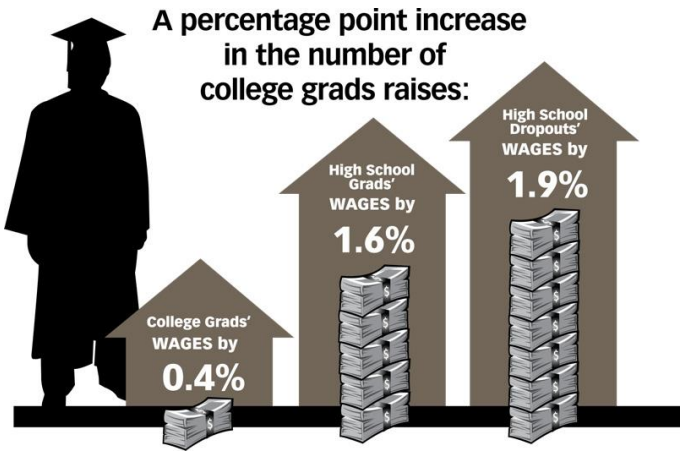
- Programming (e.g., loop, if-then conditioning) enables to automate your work and makes it consistent.
  - ➢ STATA supports these simple programming features!
- Automating your STATA work saves you from repeating very similar codes over and over again.
- It also reduces chances of mistakes whenever you try to tweak the codes in each step.



Let's Conduct a Hypothetical Empirical Investigation

- Research question (note that this is a hypothetical question)

  Does college education increase wages? If so, how does labor union influence the relationship between education and wages?




```
sysuse nlsw88, clear
sum
gen lnWage = ln(wage)
```

(1) Defining Program Variable: *Macro*

- Macros in STATA are the equivalent of variables in other programming languages to represent variables, values, texts, commands, statements etc.
  - ➢ Macros can be local (only visible within the range { }) or global.

Defining variable lists

```
local varlist age race married grad south smsa
summarize `varlist'

local demographic age race married
local job_condition tenure i.industry i.occupation

reg lnWage `demographic', robust
reg lnWage `demographic' `job_condition', robust
reg lnWage collgrad `demographic' `job_condition', robust
```

Storing commands

```
local union_condition "if union==1 & c_city==0"
reg lnWage collgrad `demographic' `job_condition' `union_condition', robust
```

Storing values

```
global parttime_hour = 30
reg lnWage collgrad `demographic' `job_condition'  if hour > $parttime_hour, robust
```

(2) Utilizing Matrix Data Structure: *Matrix* and *Mata*

- Matrix data structure enables you to manage variables, thereby your estimations, far more efficiently and effectively.
  - ➢ STATA provides a matrix programming language with similar syntax and capabilities as other matrix languages, so-called Mata.

STATA

```
matrix X = (3,7\1,5\2,3\5,4\4,6)
matrix list X
display X[3,2]

matrix U = J(rowsof(X), 1, 1)
matrix U2 = J(2, colsof(X), .)

matrix sum = U'*X
matrix list sum

matrix meanvec = sum/rowsof(X)
matrix list meanvec
```

MATA

```
mata

X = (3,7\1,5\2,3\5,4\4,6)
X
S = ("first", "second" \ "third", "fourth")
S
meanvec = mean(X)
meanvec

X*X'
X#X
A = (10, 20 \ .2, .3)
luinv(A)
lusolve(A,I(2))
help m5 intro

end
```

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix}$$

(2) Utilizing Matrix Data Structure: *Matrix* and *Mata*

- Matrix data structure enables you to manage variables, thereby your estimations, far more efficiently and effectively.
  - ➢ Indeed, Mata is mainly designed to implement the estimation models.
  - ➢ (Example) Ordinary Least Squares (OLS)

$$y = X\beta + \epsilon$$

$$\hat{\beta} = (X'X)^{-1}X'y$$

MATA

```
sysuse auto, clear

mata

y = st_data(., "price")
X    = st_data(., "mpg trunk")
n    = rows(X)
X    = X, J(n,1,1)
XtX  = quadcross(X, X)
XtXi = invsym(XtX)
b    = XtXi*quadcross(X, y)

end

mata: b
reg price mpg trunk
```

(3) Conditional Branching: *If-Else* Statement

- The *If-Else* statement allows the program to be branched by conditions. However, one final thing to note is that it is important to distinguish between the conditional *if* command (reg y x **if** ~) and the programming *if* statement (if ~ { } else { }).
  - ➢ Indeed, *if-else* statement is mainly designed to define a program or function.

Bad (wrong) example

```
sysuse nlsw88, clear
sum
gen lnWage = ln(wage)

if c_city==1 {
   reg lnWage collgrad union `demographic'
`job_condition', robust
}
else {
   reg lnWage collgrad c.collgrad#c.union
`demographic' `job_condition', robust
}
```

Good example

```
forvalues i = 0/1 {
   if `i' ==1 {
   display _newline "regression for c_city = `i'"
   reg lnWage collgrad union `demographic'
`job_condition' if c_city== `i', robust
}
   else {
      display _newline "regression for c_city = `i'"
      reg lnWage collgrad union c.collgrad#c.union
`demographic' `job_condition' if c_city== `i', robust
   }
}
```

(4) Repeating (Loop): *Foreach* and *Forvalues*

- Loops refer to execute a group of commands multiple times.
  - For example, think of the situation where you need to replace some values of 1,000 variables in the same way, or to estimate same specifications for 1,000 times (e.g., random shuffling test, rolling regressions).

Loop over variables

```
local example age race married collgrad wage

foreach var in `example'{
   display _newline "ttest `var', by(union)"
   ttest `var', by(union)
}

foreach var of local example{
   display _newline "ttest `var', by(union)"
   ttest `var', by(union)
}

foreach var of varlist `example'{
   display _newline "ttest `var', by(union)"
   ttest `var', by(union)
}
```

```
foreach i in 1 2 3 4 5 …? {
   sum wage if occupation==`i'
}


levelsof occupation, local(occ_level)

foreach i in `occ_level' {
   sum wage if occupation==`i'
}
```

(4) Repeating (Loop): *Foreach* and *Forvalues*

- Loops refer to execute a group of commands multiple times.
  - For example, think of the situation where you need to replace some values of 1,000 variables in the same way, or to estimate same specifications for 1,000 times (e.g., random shuffling test, rolling regressions).

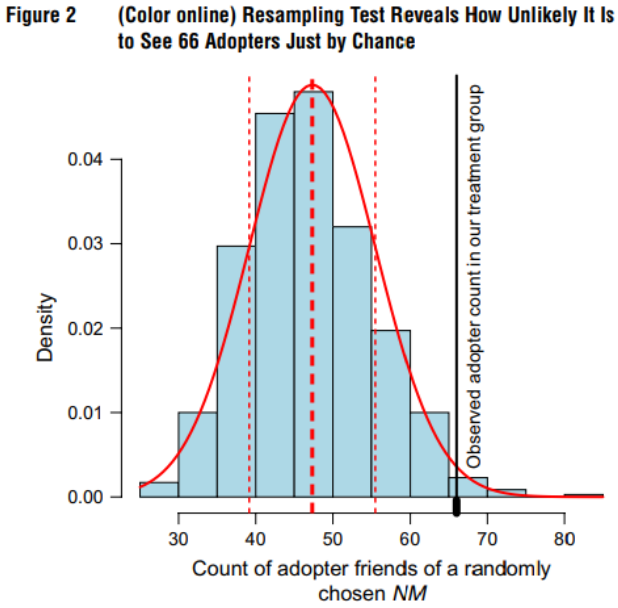
Loop over numbers

```
local example age race married collgrad wage

forvalues occ_level = 1/13 {
  display _newline "ttest wage, occupation =
`occ_level'"
  ttest wage if occupation == `occ_level', by(union)
}

forvalues occ_level = 1/13 {
  local value_label : label(occupation) `occ_level'
  display _newline "ttest wage, occupation =
`value_label'"
  ttest wage if occupation == `occ_level', by(union)
}
```

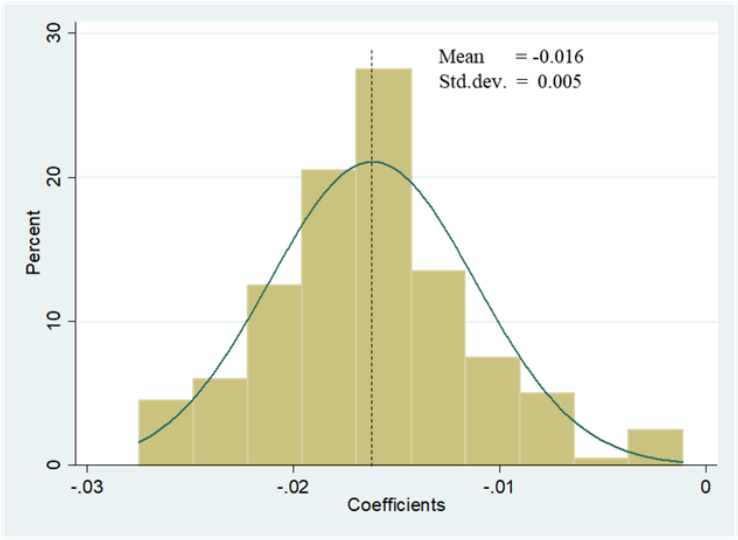
```
tab occupation, matcell(freqs)
matrix list freqs
display freqs[1,1]
display freqs[2,1]

forvalues occ_level = 1/13 {
  local value_label : label(occupation) `occ_level'
  if freqs[`occ_level', 1] >=10 {
    display _newline "ttest wage, occupation =
`value_label'"
    ttest wage if occupation == `occ_level',
by(union)
  }
}
```

(4) Repeating (Loop): *Foreach* and *Forvalues*

- Example: Resampling / bootstrapping
  - ➢ Resampling strategy has been popular for the goal of robustness checks or placebo tests (e.g., Bapna and Umyarov 2015; Park et al. 2018).



(Bapna and Umyarov 2015)



(Park et al. 2018)

Bapna, R. and Umyarov, A., 2015. Do Your Online Friends Make You Pay? A Randomized Field Experiment on Peer Influence in Online Social Networks. *Management Science*, 61(8), pp.1902-1920.

Park, J., Kim, J., Pang, M. and Lee, B., 2018. The Deterrent Effect of Ride-Sharing on Sexual Assault and Investigation of Situational Contingencies. *KAIST Working Paper.*

(4) Repeating (Loop): *Foreach* and *Forvalue*

- Example: Resampling / bootstrapping

Bootsttrapping

```
sysuse nlsw88, clear
gen lnWage = ln(wage)

reg lnWage collgrad union `demographic' ,
vce(bootstrap, rep(100) size(100))
```

Customized resampling

```
matrix jypark = J(100,1,.)
matrix list jypark

forvalues i=1/100{
  sysuse nlsw88, clear
  gen lnWage = ln(wage)
  sample 100, count
  reg lnWage collgrad union `demographic', robust
  matrix b=e(b)
  matrix jypark[`i',1]=b[1,1]
}

matrix list jypark

mata
A = st_matrix("jypark")
coefficient_mean = mean(A[,1])
coefficient_sd = variance (A[,1])^0.5
end

mata: coefficient_mean
mata: coefficient_sd
```

(5) Defining Program Function: *Program*

- You can define your own function or program which is a collection of statements that are grouped together to perform an operation. (e.g., 'reg' command is a build-in program)

```
sysuse nlsw88, clear
gen lnWage = ln(wage)

program define jypark_ttest
tab occupation, matcell(freqs)
matrix list freqs

forvalues occ_level = 1/13 {
  local value_label : label(occupation) `occ_level'
  if freqs[`occ_level', 1] >=10 {
    display _newline "ttest wage, occupation =
`value_label"“
    ttest wage if occupation == `occ_level',
by(union)
  }
}
end

jypark_ttest
program drop jypark_ttest
```

```
program define jypark_tabstat
  levelsof `1', local(level)
  foreach i in `level' {
    local value_label : label(`1') `i'
    sum wage if `1' ==`i'
display _newline "summary of wage, `1' =
`value_label"'
  }
end


jypark_tabstat occupation
tabstat wage, by(occupation)

jypark_tabstat industry
tabstat wage, by(industry)
```

(6) Automating Your Work: *Do* and *Ado* Files

- *Do* file includes your own STATA codes, enabling repeated actions. In contrast, *Ado* file defines a generic program or package which anyone can use.
  - ➢ Unlike *Do* file, *Ado* file should not contain specific functions or variables.

Do File

doedit

Ado File

sysdir
sysuse auto, clear
encode(make), gen(_make)
jypark_adotest _make

You need to locate your own ado files to PERSONAL directory.

```
. sysdir
   STATA:  E:\STATA14\
    BASE:  E:\STATA14\ado\base\
    SITE:  E:\STATA14\ado\site\
    PLUS:  c:\ado\plus\
PERSONAL:  c:\ado\personal\
OLDPLACE:  c:\ado\
```

```
sysuse nlsw88, clear

// Own program
program define jypark_tabstat
   levelsof `1', local(level)
   foreach i in `level' {
      local value_label : label(`1') `i'
      sum wage if `1' ==`i'
display _newline "summary of wage, `1' = `value_label'"
   }
end

/* Build-in program */

jypark_tabstat occupation
tabstat wage, by(occupation)

jypark_tabstat industry
tabstat wage, by(industry)
```

STATA_Lab4.do

```
program define jypark_tabstat
   levelsof `1', local(level)
   foreach i in `level' {
      local value_label : label(`1') `i'
      display _newline "ado test `1' = `value_label'"
   }
end

exit
```

jypark_adotest.ado

Final STATA Replication Project

Let's Replicate the Work of Evans et al. (2016)

- Research question

  Are homelessness programs really effective in preventing homelessness?

ECONOMIC POLICY

The impact of homelessness prevention programs on homelessness

William N. Evans,[1,2,3] James X. Sullivan,[1,3*] Melanie Wallskog[4]

Despite the prevalence of temporary financial assistance programs for those facing imminent homelessness, there is little evidence of their impact. Using data from Chicago from 2010 to 2012 (n = 4448), we demonstrate that the volatile nature of funding availability leads to good-as-random variation in the allocation of resources to individuals seeking assistance. To estimate impacts, we compare families that call when funds are available with those who call when they are not. We find that those calling when funding is available are 76% less likely to enter a homeless shelter. The per-person cost of averting homelessness through financial assistance is estimated as $10,300 and would be much less with better targeting of benefits to lower-income callers. The estimated benefits, not including many health benefits, exceed $20,000.

Evans, W.N., Sullivan, J.X. and Wallskog, M., 2016. The Impact of Homelessness Prevention Programs on Homelessness. *Science*, 353(6300), pp.694-699.

Let's Replicate the Work of Evans et al. (2016)

- Table 2. ITT effects of fund availability on shelter spells

Dependent variable	Shelter admittance		Days spent in shelter
	3 months after calling	6 months after calling	6 months after calling
Main sample			
Funds are available	−0.014**	−0.016**	−2.620**
(Standard error)	(0.005)	(0.005)	(0.878)
n	4448	4448	4448
Mean of dependent variable for control group	0.016	0.021	3.132
Homogeneous subsample			
Funds are available	−0.019	−0.021	−2.987
(Standard error)	(0.013)	(0.013)	(1.911)
n	1431	1431	1431
Mean of dependent variable for control group	0.027	0.033	4.714
Rent callers			
Funds are available	−0.014**	−0.015**	−2.278**
(Standard error)	(0.006)	(0.006)	(1.025)
n	3574	3574	3574
Mean of dependent variable for control group	0.018	0.021	3.169
Security deposit callers			
Funds are available	−0.014*	−0.026**	−4.571**
(Standard error)	(0.008)	(0.010)	(1.538)
n	874	874	874
Mean of dependent variable for control group	0.012	0.020	3.062
Rent callers, below median income			
Funds are available	−0.023**	−0.022**	−3.389**
(Standard error)	(0.008)	(0.009)	(1.486)
n	1781	1781	1781
Mean of dependent variable for control group	0.023	0.025	3.565
Rent callers, above median income			
Funds are available	−0.004	−0.007	−0.942
(Standard error)	(0.008)	(0.009)	(1.352)
n	1790	1790	1790
Mean of dependent variable for control group	0.013	0.018	2.783

*$P < 0.10$; **$P < 0.05$; two-tailed *t* test of the difference between treatment and control groups.

Homeless_Table2.do

Evans, W.N., Sullivan, J.X. and Wallskog, M., 2016. The Impact of Homelessness Prevention Programs on Homelessness. *Science*, 353(6300), pp.694-699.

End of Document