Stata Recitation - Week 5 - Modifying Data I

McCourt School of Public Policy, Georgetown University

SWBAT

- Open and save non-system data sets
- Work in a do-file
- Create new variables as functions of existing variables
- Use notes to document workflow

Resource for Variable Creation - on blackboard Nagler, Jonathan. 1995. Coding Style and Good Computing Practices. PS: Political Science and Politics 28(3): 488-492.

Opening non-system data sets.

- Log in to Blackboard
- Download data set: auto-week5.dta
- Move it to a folder on your desktop: week5
- Open Stata with the program icon
- Change working directory to the week5 folder using File >> Change Working Directory
- Notice `cd` command printed in results window

Basic variable creation command: Generate

- Make sure to start with an existing data set. Otherwise, there are no observations to assign a value to!

```
clear
use auto-week5.dta
generate x1 = 1
* list first 10 observations
list in 1/5

generate x2 = 2
list in 1/5

generate y = x1 + x2
list in 1/5

sum x1 x2 y
```

- What if we changed our mind about the value of `x1`?
- You can't generate it again:

```
gen x1=2
x1 already defined
r(110);
```

- We could use the replace command:

```
replace x1 = -1
list in 1/10
```

- But then y is incorrect, so we also have to replace y.

```
replace y = x1 + x2
list in 1/10
```

- But this is a very bad idea.
- Requires a lot of extra typing, error prone, no clear record.
- The better approach is to re-create the variable correctly.
- You can do this easily, if you are working in a do-file.

A note on drop (DANGER):

- You can use the command `drop` to remove variables from your dataset.
- However, it is bad practice to use `drop` when you need to recreate a variable.
- Instead, fix the mistake in your do-file and rerun all your commands.

```
gen testdrop
drop testdrop
```

In Class Activity 1

- Create a do-file that creates x1, x2, and y.
- Do-file should include -use- command
- Change the value of x1 and re-run the do-file.
- Use the do-file to input the remaining commands for this session.

Commenting your work

What are comments?

- Comments refer to text that is in the do-file, but ignored by Stata
- Comments are green in the dofile editor
- Different types of comments: `help comments`

What is the point of comments?

- Some data transformations are self-explanatory, i.e. `logmpg = log(mpg)`
- Most are not.
- Use comments to document what you are doing,
- helpful for other people reading your code
- helpful for yourself when you re-read your code, and have no idea what you were doing.
- Always use more comments than you think that you need
- Go back and comment your do-file.

Saving Progress

- Now we have made some changes to the data set that we want to preserve.
- Your main product is the do-file, not the data set.
- We save both, but the do-file is the important part.
- You may want to turn on the `Save before do/run` option.
- In Windows, this is under Edit >> Preferences in the do-file window
- Whenever you start working on that data again, start from the do-file, not the data set.
- To save the dataset, add a `save , replace` command to the end of the do-file.
- Give saved data set a new name
- Data set will be saved into the working directory unless specified otherwise
- Do file should have these commands.

```
use "auto-week5.dta", clear

generate x1 = 1
```

```
generate x2 = 2
generate y = x1 + x2

sum x1 x2 y
save "auto-week5-modified.dta", replace
```

Re-run do file with save command.

Close out of Stata. Reopen Stata, open do-file, run do-file.

- Do-file template, with hassle-free logging of output:

```
* Set working directory
* Note: get this command using File >> Change Working Directory
cd "C:\Users\myusername\..."

* If any log file is open, close it
capture: log close
* Start a new log file, replacing any previous versions
log using "mylogfile.txt" , text replace

* Don't display -more- in results window (optional)
set more off

* Clear any changes that have been made to the data in memory
clear
* open fresh version of source data
use "mysourcedata.dta"

* Stata commands *

* Save new data set with changes made by this do-file
* replace any previous version, keeping only the most up-to-date version
save "myupdateddata.dta" , replace

* Close log
log close
```

Operators and Functions

- Add variable creation commands to do-file before the save command.
- We can also create new variables from existing variables
- A list of operators can be found here:

```
help operators
generate weightforlength = weight/length

browse weight length weightforlength
sum weight length weightforlength

generate mpgsq = mpg^2
browse mpg mpgsq
sum mpg mpgsq
```

- You can do more complex operations with functions

```
help functions

generate logmpg = log(mpg)
browse mpg logmpg
sum mpg logmpg

generate int_headroom = round(headroom)
```

- Or combine functions and operators

```
generate z1 = log(mpg* turn) + sqrt(abs(x1 * gear_ratio))
```

- General rules on operators
- white space is helpful for legibility, but not required
- use parentheses when combining multiple operators
- General rules on functions
- Use full name, no abbreviations
- No space between function name and parentheses
- Always use parenthesis, even if there are no arguments
  - `gen u = runiform()`
- The terms in the parenthesis are called "arguments"
- Arguments can be variables, numbers, other functions, or combinations
- Separate multiple arguments with commas
- Spaces don't matter inside the parentheses. Use extra spaces for legibility.

```
gen max_1 = max( mpg , headroom , weight , turn )
list mpg headroom weight turn max_* in 1/10

gen max_2 = max( mpg , headroom*10 , weight / 100 , turn )
list mpg headroom weight turn max_* in 1/10

gen max_3 = max( mpg , headroom*10 , log(weight)/2 , turn )
list mpg headroom weight turn max_* in 1/10

gen max_4 = max( 50 , 23 , 81 , 72 )
list mpg headroom weight turn max_* in 1/10

gen max_5 = max( 50 , 23 , 81 , 72 , mpg , headroom*10 , weight/10 , turn )
list mpg headroom weight turn max_* in 1/10
```

Common Errors

- Generate the same variable twice

  ```
  gen x=1
  gen x=2
  x already defined
  r(110);
  ```

- Replace before generate

```
replace y = 3
variable y not found
r(111);
```

- Invalid variable name

```
gen 4score = price/mpg
help varname
4score invalid name
r(198);

gen score4 = price/mpg
```

- Not a valid operator

```
gen a = price \ mpg
gen a = price / mpg
```

- If the `=exp` part of the generate statement is incorrect, it is often interpreted as a variable name.
- The variable name may be either invalid or non existent.
- Not a valid operator, no white space

```
gen b = price\mpg
gen b = price'mpg
```

- Not a valid operator, with white space

```
gen b = price ' mpg
```

- Spelling mistake

```
gen c = npg
gen c = mpg
```

- not specifying arguments correctly

```
gen d = max(length weight trunk)
gen d = round(length , weight)
```

- space between function name and parentheses

```
gen d = round (turn)
```

- Or, if a variable exists with that function name:

```
gen round = 0
gen d = round (turn)
```

In Class Activity 2

Answer the following questions using the example data set: `census.dta`

Use a do-file to produce the output that you used to arrive at your answers.

Use comments before and after the command to document the question you are answering and the answer.

If you haven't already, create a log to save the results.

1. Create a new variable giving the total population age 17 and younger in each state. What is the average state population of those age 17 and younger?
2. Create a new variable giving the proportion of residents age 17 and younger in each state. Which state has the lowest proportion of population age 17 and younger? What was that proportion?
3. Create a new variable giving number of marriages as a proportion of total population. What state had the highest proportion of marriages? What was that proportion?
4. Create a new variable giving number of divorces as a proportion of total population. What state had the highest proportion of divorces? What was that proportion?
5. What is the average rural (not urban) population for all states?
6. Create a new variable giving the log of population for each state. Create histograms for population and log population. Title graphs appropriately. Export graphs and insert into a Word document.

In Class Activity 2 Answer

```
clear
sysuse census.dta


* 1. Create a new variable giving the total population age 17 and younger in each state.
* What is the average state under population of age 17 and younger?

generate poplt18 = poplt5 + pop5_17
summarize poplt18

* 1,272,229


* 2. Create a new variable giving the proportion of residents age 17 and younger in each sta
*    Which state has the lowest proportion of population age 17 and younger?
*    What was that proportion?

generate proportionlt18 = poplt18 / pop
sum proportionlt18
list state proportionlt18 if proportionlt18<.25

* Florida 0.242


* 3. Create a new variable giving number of marriages as a proportion of total population.
```

```
*    What state had the highest proportion of marriages?
*    What was that proportion?

generate proportionmarriage = marriage / pop
sum proportionmarriage
list state proportionmarriage if proportionmarriage > 0.14

* Nevada 0.143


* 4. Create a new variable giving number of divorces as a proportion of total population.
*    What state had the highest proportion of divorces?
*    What was that proportion?

generate proportiondivorce = divorce / pop
sum proportiondivorce
list state proportiondivorce if proportiondivorce > 0.017

* Nevada 0.017


* 5. What is the average rural (non-urban) population for all states?

generate poprural = pop - popurban
sum poprural

* 1,189,896


* 6. Create a new variable giving the log of population for each state.
* Create histograms for population and log population.
* Title graphs appropriately.
*    Export graphs and insert into a Word document.

gen logpop = log(pop)

histogram pop , title(Distribution of State Population)
graph export popdist.png, replace

histogram logpop , title(Distribution of State Log Population)
graph export logpopdist.png, replace
```