

Stata Basics

Will Doyle

PURPOSE

In this class we'll continue exploring some of Stata's basic functionality.

A template for do files

All *.do files for this course should follow the outline provided by the example do files. An example template is also posted on the course website. The point is to have a standard structure that will make sense for every new *.do file you create.

Downloading packages (*.ado files)

One of Stata's great virtues is that it is highly extensible. Users are frequently creating new commands for Stata and posting them on the web. To get these commands, we use the `net search` command. From there, a list of possible matches will be presented. After choosing the one you want, go ahead and download it into Stata. You now have access to the new command.

```
. net search renvars
(contacting http://www.stata.com)
```

```
4 packages found (Stata Journal and STB listed first)
-----
```

```
dm88_1 from http://www.stata-journal.com/software/sj5-4
  SJ5-4 dm88_1. Update: Renaming variables, multiply and... / Update:
  Renaming variables, multiply and systematically / by Nicholas J. Cox,
  Durham University, UK / Jeroen Weesie, Utrecht University, Netherlands /
  Support: n.j.cox@durham.ac.uk, j.weesie@fss.uu.nl / After installation,

dm88 from http://www.stata.com/stb/stb60
  STB-60 dm88. Renaming variables, multiply and systematically / STB insert
  by Nicholas J. Cox, University of Durham, UK / Jeroen Weesie, Utrecht
  University, Netherlands / Support: n.j.cox@durham.ac.uk
  j.weesie@fss.uu.nl / After installation, see help renvars

cleanchars from http://fmwww.bc.edu/RePEc/bocode/c
  'CLEANCHARS': module to replace specific characters or strings in variable
  names and/or variable labels and/or string variable values and/or value
  label names and levels with stated characters/strings (using 1-1 or m-1
  match) / cleanchars is a program that helps out with replacing /

renvarlab from http://fmwww.bc.edu/RePEc/bocode/r
  'RENVARLAB': module to rename variables, with option of using variable
  labels to create new variable names / This command is an extension of
  renvars (also available from / SSC), which renames a list of variables by
  applying the given / transformation to all of the variables. It has all of
```

Loading data

Stata data files end in `*.dta`. They are easily loaded with the `use` command. Some datasets, like the one we will use today, can be downloaded from the Stata website directly with the `webuse` command. While we won't really use these toy datasets in our research, they can come in handy for small exercises (such as we will do today). They are also the datasets used in the Stata help files.

```
. webuse school, clear

. save school, replace
file school.dta saved
```

outsheet dataset

Stata can export data in almost any commonly-used format. The most commonly used format for data files is in what's called ASCII delimited form, with a delimiter of either a comma or a tab. To export a dataset we use the `outsheet` command. You can specify which variables from the current dataset you'd like to export. `*.csv` is a good format for Microsoft Excel files.

```
. outsheet using "school_data.csv", comma replace
```

QUICK EXERCISE

Export the current dataset in tab delimited format, then go ahead and open up the new dataset in Excel.

insheet dataset

Similarly, Stata can read in data in most any format using the `insheet` command:

```
. insheet using "school_data.csv", comma clear
(11 vars, 95 obs)
```

describe dataset

Once you have your dataset in memory, you can `describe` it to get a quick overview of what you have:

```
. //Save as tab delimited
. outsheet using "school_data.tsv", replace

. //Open up tab delimited file
. insheet using "school_data.tsv", clear
(11 vars, 95 obs)

. describe
```

Contains data

```
obs:          95
vars:          11
size:         1,900
```

	storage	display	value	
variable name	type	format	label	variable label

```

-----
obs          byte      %8.0g
pub12        byte      %8.0g
pub34        byte      %8.0g
pub5         byte      %8.0g
private      byte      %8.0g
years        byte      %8.0g
school       byte      %8.0g
loginc       float     %9.0g
logptax      float     %9.0g
vote         byte      %8.0g
logeduc      float     %9.0g
-----

```

Sorted by:

Note: dataset has changed since last saved

Labeling data and variables

Properly labeling datasets and variables will make life *much* easier for you and anyone else who uses your dataset. To label an entire dataset, use the `label data` command:

```
. label data "Voting on school expenditures"
```

To label variables, use the `label variable` command:

```
. label variable loginc "Log of income"
```

```
. label variable vote "Voted for public school funding"
```

Make sure that your variable labels are highly descriptive and directional (e.g., instead of labeling a binary variable **gender** label it **female**). We can see our labels if once again `describe` our data:

```
. describe
```

Contains data

```

obs:          95                                Voting on school expenditures
vars:          11
size:         1,900

```

```

-----
variable name  storage  display  value
              type    format   label   variable label
-----
obs           byte     %8.0g
pub12         byte     %8.0g
pub34         byte     %8.0g
pub5          byte     %8.0g
private       byte     %8.0g
years         byte     %8.0g
school        byte     %8.0g
loginc        float    %9.0g                Log of income
logptax       float    %9.0g
vote          byte     %8.0g                Voted for public school funding
logeduc       float    %9.0g
-----

```

Sorted by:

Note: dataset has changed since last saved

Labeling values within variables

For many binary/categorical variables, you'll want to say what the underlying levels of the variable in the dataset mean. This is a two-part process. First you define the label values through `label define`; next you apply them to your particular variable with `label values <variable> <label>`:

```
. tab vote
```

Voted for			
public			
school			
funding	Freq.	Percent	Cum.
-----+			
0	36	37.89	37.89
1	59	62.11	100.00
-----+			
Total	95	100.00	

```
. label define voteopts 0 "no" 1 "yes"
```

```
. label values vote voteopts
```

```
. tab vote
```

Voted for			
public			
school			
funding	Freq.	Percent	Cum.
-----+			
no	36	37.89	37.89
yes	59	62.11	100.00
-----+			
Total	95	100.00	

Transforming variables

Transforming a variable refers to using an operation to create a new version of an existing variable. In our dataset, both income and household spending on education are expressed in terms of the natural log of the existing variable. If we want to have the values of these variables in nominal scale, we need to use the `gen` command to create a new variable:

```
. gen inc = exp(loginc)
```

```
. sum loginc inc
```

Variable	Obs	Mean	Std. Dev.	Min	Max
-----+					
loginc	95	9.971017	.4118853	8.294	10.82
inc	95	23093.31	8871.35	3999.8	50011.07

Recoding variables

Recoding a variable involves changing the values of a variable based on its existing variables. We often want to recode variables in order to make them more useful for whatever analysis we're working on. I'll show you two different ways of recoding, one based on the `gen` command and a more complicated one using the `egen` command with the `recode` command:

```
. sum inc
```

Variable	Obs	Mean	Std. Dev.	Min	Max
inc	95	23093.31	8871.35	3999.8	50011.07

```
. gen inc_bin = 0
```

```
. replace inc_bin = 1 if inc > r(mean)
(30 real changes made)
```

```
. egen inc_q = cut(inc), group(4)
```

```
. recode inc_q (0 = 1 "First Quartile") ///
> (1 = 2 "2nd Quartile") ///
> (2 = 3 "3rd Quartile") ///
> (3 = 4 "4th Quartile"), gen(new_inc_q)
(95 differences between inc_q and new_inc_q)
```

```
. // Binary variable for greater than median
. sum inc, detail
```

inc					

	Percentiles	Smallest			
1%	3999.8	3999.8			
5%	12500.2	7500.312			
10%	12500.2	7500.312	Obs		95
25%	17500.79	7500.312	Sum of Wgt.		95
50%	22493.91		Mean		23093.31
		Largest	Std. Dev.		8871.35
75%	27501.62	50011.07			
90%	34996.39	50011.07	Variance		7.87e+07
95%	34996.39	50011.07	Skewness		.9732624
99%	50011.07	50011.07	Kurtosis		4.775115

```
. gen bin_inc_med=0
```

```
. replace bin_inc_med = 1 if inc > r(p50)
(30 real changes made)
```

```
. tab bin_inc_med
```

bin_inc_med	Freq.	Percent	Cum.
0	65	68.42	68.42

1		30	31.58	100.00
-----+-----				
Total		95	100.00	

```
. gen bin_inc2=0

. replace bin_inc2= 1 if new_inc_q==3 | new_inc_q==4
(61 real changes made)
```

```
. tab bin_inc2
```

bin_inc2		Freq.	Percent	Cum.
-----+-----				
0		34	35.79	35.79
1		61	64.21	100.00
-----+-----				
Total		95	100.00	

```
.
```

QUICK EXERCISE

Create a variable that is equal to 1 if it's greater than the median of income. Properly label the variable and its values.

Computing a new variable

Computing a variable involves using the values of other variables to create a new variable. For instance, to calculate an effective property tax rate, we want to divide the property tax value by income:

```
. gen ptax = exp(logptax)

. gen taxrate = ptax / inc
```

EXERCISES

1. Create a new binary variable for whether or not the family has any children in public schools. Properly label your variable and variable values.
2. Create a new variable for percent of household income spent on education. Properly label your new variable.
3. Create a new variable for persons with low, moderate and high percentages of spending on education. Label the variable and value labels properly.
4. Tabulate household spending and voting for public school funding. What do you find?

Init: 06 June 2015; Updated: 05 September 2018