Econ 452 Section 2 - STATA

Connor Cole

September 23, 2015

Basic Data Management

Dropping Observations

In our problem sets, we will often drop observations if they are missing values for variables of interest (note that some researchers, instead of dropping missing observations, will sometimes 'impute' values to missing variables). Note that dropping an observation is equivalent to dropping a row in the data. A missing observation in STATA is recorded, in the variables we use, as a "." (our data has no string variables that are composed of text information only. For these variables, a missing observation may be recorded in many different ways depending on how the text defines a missing variable). Thus, dropping an observation because it is missing some variable would be coded as:

```
drop if variablename1==.
```

Variable Management

Drop and Generate Commands

When you open a dataset, there are a set of variables listed in the variables window. Often, in the course of analyzing the data, you will want to create additional variables that are functions of the existing variables, either to create additional index variables or manipulate existing variables into new information. First, we consider simple operations. If we wanted to add two variables together to create an additional variable, we would write:

```
generate newvariable = variablename1 + variablename2
```

We could then easily drop this variable by writing:

```
drop newvariable
```

Replace Command

If we want to replace the values of one variable with the values of another, we would write:

```
replace newvariable = variablename1
```

If we wanted to just replace these values for a particular subgroup of the data, we could just add a conditioning statement clarifying that we only want to replace values of one variable with another for that particular subgroup. All other values of the variable for observations outside of the subgroup would then be unaffected.

```
replace newvariable = variablename1 if variablename==2
```

Using Drop and Generate to Make Subgroups

Often, we will want to create a new variable that creates an index for individuals for conditioning that is easier than spelling out all the individual conditioning arguments over and over. For example, if we wanted to compute statistics for the subgroup of individuals who are Black and earn below the median of earnings and are under the age of 30, writing out all three of those conditioning statements each time might be tedious not to mention difficult to understand for an outside party to understand when reviewing the code. To make our work easier, we could create an additional variable that defines this particular subgroup. While there are many ways to accomplish this goal, I recommend this strategy.

First, define a new variable with all missing entries:

```
generate subgroup=.
```

Then, we replace our subgroup variable with numbers to define various subgroups:

```
replace subgroup=0 if variablename1<=200

replace subgroup=1 if variablename1>200 & variablename1<=400

replace subgroup=2 if variablename1>400
```

Note that STATA treats missing observations as meeting whatever conditioning statement you provide unless you explicitly identify in your conditioning statement that the conditioning variables cannot be missing. We will not have to deal with this problem, as you will just generally drop missing observations in the first place, but it will be important to remember if you do work with STATA in the future.

Labeling Variable Values

As was discussed previously, the categorical variables in the dataset have value labels attached to them. Attaching labels to categorical variables can be a handy way of describing what the categorical variables you've created mean without requiring that someone using the dataset look at a separate data dictionary. To attach labels to a variable, you need to first define a label, and then apply it to the variable in question. To define the label, we type textttlabel define, provide a name for the value labels we are creating, and then list the each of the values the variable takes on followed by a statement in quotation marks that provides the text label that we'd like to see.

Staying with the previous example, we could create the following variable label:

```
label define variablelable1 0 "Variablename1 is less than 200" 1 "Variablename1
is between 200 and 400" 2 "Variablename1 if greater than 400"
```

And then apply it to the categorical variable we have created:

```
label values subgroup variablelabel1
```

Note that these variable labels change the appearance of the subgroup variable we have created, but they do not change the underlying values of the variable, which will still be numeric as before. We have simply changed the way that the variable appears in our STATA dataset. Again, if we wanted to look at values of this variable without the label we have created, we type:

```
tabulate subgroup, nolabel
```

The 'Egen' Command

The 'gen' command makes new variables that can only be functions of values in each row separately. The 'egen' command opens up all sorts of calculations of new variables on the basis of multiple the values in either a row or a column. For example, if we had data on income and sex, we could use the egen command to calculate standard deviations of income for men and women separately and save it in an additional variable that gives the relevant standard deviation for the sex of each individual. Or, we might want to add up all the values of some variable in a column and save it as an additional variable, or calculate the mean of all the variables in a column. Typing in `help egen` will give all various functions you can perform on data in a column using the egen function. In general, you will use the egen function by typing `egen newvariable =` and then putting in the relevant function, with the variables it takes as an argument in parentheses. The following section describes functions that will likely be of particular interest to you on your problem set.

If you want to calculate the mean of the values in a column and save it as an additional variable, you would type:

```
egen newvar = mean(variablename1)
```

If you want to calculate the standard deviation of the values in a column and save it as an additional variable, you would type:

```
egen newvar = sd(variablename1)
```

If you want to calculate the max of the values in a column and save it as an additional variable, you would type:

```
egen newvar = max(variablename1)
```

If you want to calculate the value of some percentile of the distribution of values in a column, for example the median, and save it an additional variable you would type:

```
egen newvar = pctile(variablename1), p(50)
```