Training 2

In this training we will focus on what is often called data cleaning. That is how to clean the data in order to analyze it. Data might be dirty for a variety of reasons: variables might have weird names, variables might be in the wrong format, and there might even be some missing data.

Remember the wise words from the even wiser Marie Kondo:

*The objective of cleaning is not just to clean, but to feel happiness living within that environment*

As you work with a clean data set, you will feel that happiness. That is the goal, for you to experience the happiness of a clean data set.

Throughout Training 2, we will use the data set called “Training 2 data.dta”. This is a data set that comes from the American Community Survey (ACS). The ACS is an annual national representative sample of the US and has similar questions to the Census. It will have data on earnings, education, work experience, demographic characteristics and more. It is publicly available through IPUMS (<https://ipums.org/>)

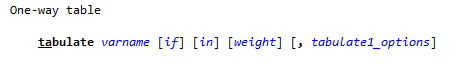
Here are the objectives for Training 2:

Objective 1: Conditional Statements

Objective 2: Manipulating data

Objective 1: Conditional Statements

The following are conditional statements that can be used with many other commands. If you go back to Training 1, Objective 3, when we discussed the tabulate command, the following was shown from the codebook:



Notice that after the varname (name of the variable) comes the potential to use conditional statements. For this command, and for most data summarizing commands, conditional statements are used before the , options.

Here are some common conditional statements:

*if* One of the most common conditional statements you will use is *if.* For example if you want to estimate the mean and standard deviation of a wages, but only want to look at earnings if the respondent was male. You could use the following:

*sum* earnings *if* male==1

Note: the double equal sign is the way STATA signals a condition is being imposed. It is saying look at male when it is already equal to one. The other option is if you want to change male to equal 1, we will look at that in objective 2.

Maybe the variable you want to condition on is a string variable, meaning that it is words and non-numeric. You could do that the following way;

*Sum* earnings *if* gender==“female”

Note: we still use the double equals sign and we put the value in quotation marks.

*and* You can use “and” when two conditions have to be me. For example, you could look at average hours worked for female business. An example of this code is as follows:

*sum* hours *if* male==0 & major==“business”

*or* You can use “or” when one of two or more conditions are met. For example, I want to look at average hours worked for respondents who are married or male

*sum* hours *if* married==1 | male==1

Note: the “or” signal is found on the keyboard right above the enter key, use the shift key with backslash.

There are other conditional statements you can use and you can always combine your favorite ones. But these three are the most common. Here are some other examples of conditional statements

*tab* age *if* education>12 tabulate the variable age for those with more than 12 years of education

*count if* education<=12 provide a count or how many observations are there with less than or equal to 12 years of education

sum earnings if major~=“business” summarize earnings of majors that are not business

sum earnings if major/=“business” /= is another way to say not equal to

Objective 2: Manipulating the Data

Manipulating the data means that we are going to generate new variables, redefine old variables, and manipulate the data in order to clean the data.

**Generate new variables**

equal to a value

The first thing we can do is generate new variables. Let’s suppose I want to create a variable that is equal to 1 for everyone in the data set.

*gen* lit=1

This will create a new variable called lit and give each respondent a value of 1.

equal to a function of existing variables

You can also generate new variables using your math skills. For example, there already exists a variable called age, what if we wanted a variable that was age squared (age2)?

gen age2=age^2

You determine the name of the variable, in this case I called it age2 which is the age squared.

What if you wanted to generate a new variable that measured hours worked per year? Well you could use hours, which is hours worked per week, and multiply it by weeks, which is weeks worked per year.

gen hours\_year=hours\*weeks

Note: you can’t call this new variable “hours” since it already exists. All new variable names have to be one word, so you can use underscore to combine words. It is always a good idea to give your variables names that describe what the variable is.

**Replace variables**

Creating dummy or binary variables

You can use replace to replace the values of a variable. The following shows a common application of gen and replace commands, creating binary variables.

Create a binary variable that equals 1 if female and 0 if not

*gen* female=0

*replace* female=1 if male==0

*gen* female=0

replace female=1 if gender==”female”

Both of those lines of codes will work.

Replacing missing data

Another example of using replace is what if you thought some data was incorrect and you want to drop those observations. For example, you don’t think anyone can work more than 80 hours per week. So, you want to drop the values when someone says they work more than 80 hours per week.

replace hours=. if hours>80

Note: the “.” Represents missing or no value of a byte variable (one with numbers).

Note: be careful when you do this, because once you do it, you can’t undo it and restore the original values later on in the code.

Generating a single variable from categorical dummy variables

Here is a problem for you to work on that will test your ability to apply gen and replace.

There are four variables for race. Create one variable for race that has four possible responses.

gen race=” “ \*note, open quotation is a missing value for string variables \*

replace race=“white” if white==1

replace race=“black” if black==1

replace race=“hispanic" if hispanic==1

replace race=“other race” if other\_race==1

or you could do this

gen race=“other race”

replace race=“white” if white==1

replace race=“black” if black==1

replace race=“hispanic" if hispanic==1

Conclusion

Was that training a lot shorter than the first training or am I getting the hang of STATA? It was shorter. While the basics are elementary, the applications are endless. We can combine conditional statements to the summary commands that we learned in Training 1, in fact you will do that quite a bit.

My experience is as you summarize the data, you will notice irregularities and anomalies that are caused by dirty data. The summary and tabulate are very helpful in noticing if there are data issues that need to be addressed.

One project I worked on examined student grades for each semester. Most students have 12-18 credit hours per semester. Running the sum command on credit hours showed a min value of 3 and a max value of 99. That is one busy student. I then tabulated the credit hour variable and saw that there were a non-trivial number of students who had less than 12 credit hours. These might be classified as part-time students. I was then forced to ask myself, do I want to include part-time students in my analysis?

There were a few students who had 19-23 hours. Thought not typical, I could think of examples where a student might take more than 18 credit hours in a semester. But then there were a bunch of students who had 99 credit hours. I looked at the dictionary or the explanation that the data provided for each variable in the data set. It identified that those with 99 credit hours were observations that were missing the number of credit hours taken. This allowed me to then clean the data to account for this missing data.

I tell this story so that you will be mindful as you summarize the data and look for ways in which your data might need to be cleaned, because it is through cleaning that you will experience happiness. Thank you Marie Kondo.