***Stata Tips #1***

***PBHS 31001 Epidemiologic Methods Winter 2024***

***Adapted from previous TAs’ notes***

The information below will be written for Stata 16.

**Do files**

Do files allow you to re-execute your commands and resume work where you left off when stopping work? The file extension for these files is “.do”. These files are required whenever you turn in Stata assignments in this class (i.e., you will turn in 2 files, one PDF and one Stata do file).

***General notes on do files:***

The do-file allows you to stay organized—you don’t have to go back and annotate and organize the file afterward when you may have forgotten exactly why you performed a specific command or if a command’s output ended up being what you needed. Within a do file, you can start a line of text with an asterisk, which will NOT be executed– this is how you make notes about your commands to yourself or others. Feel free to use this feature liberally; think of them as notes for your future self, who may come back to this project months from now and may not be as fresh on the material as you are. Learning to organize your code is a tremendous and essential skill to develop, but it comes with time and practice. In your comments, you can explain how/why you used a unique command you prefer, etc., as much or as little detail that would be helpful to remind you or someone unfamiliar with the project to get the hang of what you are currently doing.

After inputting an asterisk and writing your comments, you can write the lines of commands you would like after it. Once you are done, click the “Execute” button (far right on the top of the do file editor with the blue play button). I would make sure to click the execute button after every few lines of code you write to ensure your commands don’t have any errors and that it produced the output you want.

You can find numerous sources on this common topic online with a simple Google search.

The following examples are from a modified version of the Framingham data (“Canvas-> assignments-> Problem set 1 -> Framingham data.dta”):

**Opening files and reviewing the data**

1. If you have previously been using a different dataset that still shows up in Stata, it is a good idea to clear the memory by typing the following in the command box and hitting enter (be sure to save any changes in the previous file if need be): **clear**
2. Now open the data file being used, and then begin do-file.
3. It is always a good idea to review the data first to see what they look like. There are a number of useful commands for this. These include *list*, which is simply a display of the data in the command window, and *describe*, which will give a list of the variables and their types (string, float etc). You can simply type *describe* in the command window to list all variables (the list here is so long you will need to click the blue “more” at the bottom of the window (or hit the space bar) to see them all. If you find yourself having to continually click on “more” and want to turn this feature off, you can type, “set more off” in the command line either before or any time during your session.). More commonly, you can follow the command with the names of only the variables in which you are interested.

**list**

**describe**

**describe sex age stroke**

1. To see a statistical summary of non-string variables (you cannot run statistical analyses on strings), you can enter the abbreviation *sum* by itself to see a summary of all variables. To avoid further massive output, this is perhaps ill-advised with this large dataset (if you accidentally do this, you can type “q” or hit the red stop-sign at the top of Stata to stop the output); the following command instead only summarizes a couple of variables:

**sum age**

**sum age stroke**

You will see the number of observations and the means, standard deviations, etc., for these variables.

1. You can also create frequency tables of variables with the tabulate or *tab* command:

**tab bpmeds tab sex stroke**

You can list multiple variables after tab to cross-tabulate variables. This is useful and will be discussed in future sessions.

1. Be aware of missing data.

**tab bpmeds, missing count if totchol==.**

**Generating simple variables**

Creating new variables is integral to data analysis, including the first homework assignment. Where data are in a continuous numerical format as in the “lungca” data set, one way to do this is to perform simple mathematical functions in Stata to create a new variable. This new variable will represent the product, quotient, log etc. of the values of one or more existing variables. With continuous variables, this is easy because their raw value for a given observation is the actual value of interest, thus the variable names can be treated as numbers for the purposes of writing code.

Let’s say you want to represent cigpday (cigarettes per day) as packs per day instead.

Let’s assume there are 20 cigarettes/packet (although this assumption is debatable)

1. We’ll use the *gen* or *generate* command for this in which we name this new, modified variable *cigpack*. To divide by 20, enter:

**gen cigpack=cigpday/20**

2) As always, you can examine the data produced with *tab* and *sum*:

**sum cigpack**

1. If you decide you prefer to use 25 cigarettes/pack instead of 20, you can use the command replace:

**replace cigpack=cigpday/25**

1. You can also perform simple mathematical computations directly using variable names (e.g., gen quotient= variable1/variable2 will create a new variable that represents the quotient of the values of these variables for the available observations – there is no need to manually enter any numbers for this type of calculation).

**gen chratio=totchol/hdlc**

**sum chratio**

1. Another useful command is *egen*, which is similar to gen but a bit more complex-used for generating variables that are summarized statistics.

**egen cursmoke\_count=sum(cursmoke)**

1. If at any point you find a variable is no longer needed, doesn’t work properly, etc., you can *drop* it by using the following command where “variable name” is the name of the unwanted variable. (If you close Stata without saving when it asks, it will automatically drop all newly generated variables, preserving the original data. It will only create your new variables again when specifically commanded, such as when using a do file that does so):

**drop chratio**

If you permanently drop a variable, you will have a problem if any other new variables rely upon it for their calculated values! You will then get error codes indicating bad syntax when you try to run commands with these other, new variables.

1. Now close Stata. IMPORTANT: in general, if Stata gives you the further option of saving data when it closes, typically, you would NOT do so. This would save over the original data set you were using, which may permanently alter the original variables depending on what you did during the session.
2. Now, your pretend break is over, and you would resume work. Go to your saved do file and click on it – Stata should re-open and execute your commands including opening the data set. You will end up where you left off. NOTE: if there is an error in any command, Stata will stop at that command and will not execute any subsequent commands in your do file. You’ll see the dreaded red text indicating the problem. If you would rather look at the do file first before running it, open up Stata first and then open up your do file from the “File” dropdown menu.
3. You may now close Stata; you should not have to re-save anything at this point.

**Reshaping** (using the lungca.dta)

Going from wide to long:

reshape long period den, i(age) j(decade)

**long** tells reshape that we want to go from wide to long

**period** tells Stata that the “stem” of the variable to be converted from wide to long is period

i(age) option tells reshape that **age** is the unique identifier for records in their wide format j(decade) tells reshape that the suffix of **period** (i.e.,1 2 3 4 5) should be placed in a variable called decade

Going from long to wide:

**wide** tells reshape that we want to go from long to wide

**period** tells Stata that the variable to be converted from long to wide is **period**

**i(age)** tells reshape that **age** uniquely identifies observations in the wide form

**j(decade)** tells reshape that the suffix of **period** (1 2 3 4 5) should be taken from the variable **decade**

reshape wide period den, i(age) j(decade)

**Helpful resources:**

1. UCLA Stata Learning Module <https://stats.idre.ucla.edu/stata/modules/>
2. In the Stata command: type “help *any keyword*”