

Mgmt 469

Using Stata: An in-class instructional

Stata is a *command-driven* statistical software program. In other words, you type in a command, and Stata executes it.

- There are many statistical packages similar to Stata. Once you learn one, it is very easy to learn another.

I have told you about Stata's many advantages, but I should mention some of its drawbacks

- The most obvious is the need to learn a new language
- Stata's GUI interface, though straightforward and intuitive, can be cumbersome. This is because each command has so many options in the GUI that most users find it easier to type the commands.
- Stata has very good graphics capabilities and for some reason the GUI seems to work best when doing graphics. But Stata's graphics are not in the same league as Excel.
- The Stata electronic index can be frustrating, because you usually need to know the command name to get help using it. I have posted a file to the course page with a list of frequently used Stata commands.

Let's use Stata!

1) Configure the computer's RAM to allocate memory to Stata

- **set mem xxm** , where xx is the amount of RAM you want to allocate to Stata (e.g. set mem 100m). By giving more RAM to Stata, you speed it up and allow it to handle more data. You can give Stata a certain amount of memory every time you start up.

Just type **set mem xxm, permanently**

2) Tour Stata windows

- The first time you start Stata, you will see a number of windows that fill up only part of your desktop. You will want to resize and relocate the various Stata windows. If you make changes to the windows, Stata will default to these changes the next time you start up.

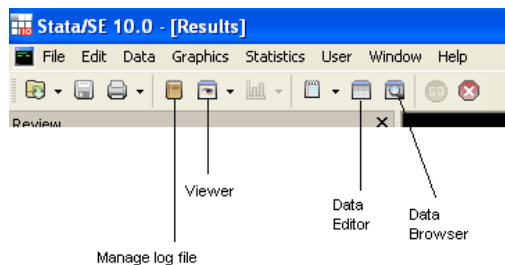
- You can also change fonts by right clicking in the window.

- If you want to save a particular window configuration for future use, you can click on **Prefs, Manage Preferences, Save Preferences, New Preferences Set**. You can then load this windowing configuration by going through the same menu.

- The Stata windows show you a history of commands, a list of variables, a command line and the results.

- You can re-enter a command by clicking on it. If you don't want to type variable names, you can click on them. (The variable name will appear in the command window.)

3) There are several useful buttons on the button bar that I will describe later



4) Stata permits you to invoke limited DOS commands. Though they are archaic, it is useful to learn a few.

- **cd c:\mgmt469** (to change the default folder)
- **dir** (to list out the contents of a folder)
- **pwd** (to see the current path and folder)

I find it useful to change my default folder to the one that holds my data set.

5) To open a Stata data file:

- Stata can only work with Stata data files, which have the suffix **.dta**

- Just as with other software, there are several options for opening a Stata data file. One nice option is to go to the **File** menu and select **Open Recent**.

- Another option is to type a Stata command, telling Stata to **use** a particular file.

- The syntax for the command is **use path\filename**

- For example, on my laptop I have a file **hmodata.dta** that resides in my d:\mgmt469 folder. To open that file, I have two command choices

- **use c:\mgmt469\hmodata**

or

- **cd c:\mgmt469**
use hmodata

The file hmodata.dta is now our active file.

[Note: When referring to path names with spaces (e.g., c:\my path\filename), you must use quotation mark (e.g. "c:\my path\filename")]

6) Note how these (and all other) commands appear in the *Review* window. You can press the *page up* or *page down* key to access past commands. Alternatively, you can click on past commands in the *Review* window.

7) You can obtain a brief technical description of all the variables, as well as any variable labels (descriptions of the variables) by typing **de**

8) You can eyeball the data in their rows and columns by clicking on the **browse** or **edit** buttons.

9) You can also examine your data by typing **list var1 var2**. If you only want to see observations in rows m through n, type **list var1 var2 in m/n**. (E.g., to see the first 10 rows, type **list var1 var2 in 1/10**.)

10) You can provide labels (brief descriptions) for your variables using label variable command. These labels appear when you use the **de** command.

- label var zold90 "% pop over age 65"

It is a good idea to label every variable in your data unless the variable name is fully self-explanatory.

11) You can get summary statistics, tabulations, and correlations

- **su hmopct1** (to obtain the mean, std. dev., range, and sample size)
- **su hmopct1, de** (same as above, plus median and other percentiles of the distribution as well as outliers)
- **codebook hmopct1** (another useful way to look at the data; helps in ways that **su, de** cannot)
- **tab hmopct1** (to get all values and their percentiles)
- **corr hmopct1 hmopct2** (for a simple correlation matrix)
- **pwcrr hmopct1 hmopct2, sig** (to obtain significance levels for the correlations)

12) There are several ways to keep track of your results

- To print the most recent results, just click on the print icon. This can be a real paper waster.
- I strongly urge you to create a “Stata log” – a separate file containing all your work. Here’s how to create a log file.
 - Click on the **log** button. This opens a familiar “save file” type window.
 - Save the file as type “Log” (do not use the .smcl default)
 - Alternatively, you can type **log using mylogfile, text**
- Stata will save a record of your work in a text file. You can edit and print this record using MS Word.
- You can temporarily stop the log file by typing **log off** and restart it by typing **log on**. In this way, you can avoid clogging up the log with unnecessary stuff. Type **log close** to end your log file.
- You can place comments in your log by placing an asterix (*) before the Stata command line – Stata will not execute lines beginning with an *
- To print the log file from within Stata, click on the **log** button again and select “View snapshot of log file”. Your log file will appear in a separate window. Now click on the little triangle next to the print icon and select the viewer.

12) It is easy to estimate a regression model

- Suppose you want to estimate the following model

$$\text{hmopct1} = B_0 + B_1 \cdot \text{qherf80} + B_2 \cdot \text{zold90}$$

Just type **regress hmopct1 herf80 zold90**

- Or, type **regress** and then click on the variables in the *Variables* window

13) You will often want to merge data from several data sets. You do this with the **merge** command.

- Both data sets must be Stata data sets.
- Both must have a common variable that is used to match the data sets to each other. Both data sets must be sorted by that variable.
- To make things interesting, let's add a data set that is currently in DOS format

14) First, **clear** the workspace. (Based on past performance, there is a 75 percent chance that I will forget to do this in class!)

15) The file worker.dat is also in the d:\mgmt469 folder. It is useful to take a look at it, for example by using MS Word. Here is what the first few rows look like:

```
0 .3830379 .2828467 .0605666
40 . . .
80 .2555066 .3215859 .0396476
120 . . .
160 .3108614 .2921348 .1161049
```

16) The **infile** command enables you to read in the DOS data set

- **infile citycode worker1 worker2 worker3 using
d:\mgmt469\worker.dat**

- (These are variable names of my choosing: **citycode** is a code used by the U.S. census to identify each metropolitan area in the U.S. **Worker1** is the percentage of the workforce in each metro area that works for firms with 5-100 workers. **Worker2** and **worker3** pertain to other size groups.)

17) Note the .'s in this data set. This is standard designation for missing values. For some of the rows, I am missing data on worker1, worker2, and worker3. Stata will ignore these rows when performing calculations using these variables.

18) You now have a Stata data set in your workspace. Label the variables and save it as a Stata data set, **worker.dta**

19) To combine the data sets **hmodata.dta** and **worker.dta** you need to use the **merge** command

- You need a *common variable* that identifies the unit of observation and links the two data sets. In this case, the linking variable is **citycode**.

- In general, the syntax to merge an active file with another file is:

merge varname using path\filename

where **varname** is the linking variable and **path\filename** is the file you are merging with the active file

- Start with one of the two files as your active file. In this case, we start with **worker.dta**.

- Unfortunately, you cannot simply type **merge citycode using hmodata**

The data need to be *sorted* to make sure that they match up correctly.

- The proper steps are:

Begin with your active file (in this case, **worker.dta**)

```
sort citycode  
save worker, replace  
use hmodata  
sort citycode  
merge citycode using worker  
drop _merge
```

20) The new, larger data set can be saved and analyzed.

The GUI merge interface is actually quite easy, but less helpful if you are programming.

21) It is very easy to create new variables.

- Suppose that you want a variable that equals the percentage of workers in large and very large firms (categories 2 and 3 in the data).
- Just pick a variable name (say, “bigfirm”) and use the generate, or **ge** command:

- ge bigfirm = worker2 + worker3
- Use **replace** to change values of an existing variable (e.g. **replace worker2=worker2*100**)

22) Use the **if** option to execute any command on a subset of observations. The **if** “operators” are:

- if a == b** (two equals signs): if a equals b
- if a > b**: if a is greater than b
- if a < b**: if a is less than b
- if a >= b**: if a is greater than or equal to b
- if a <= b**: if a is less than or equal to b
- if a ~= b**: if a does not equal b

For example: **ge bigfirm = worker2+worker3 if pmsa < 1000**

Here is how to create a dummy variable that equals 1 if the pmsa has more than 50% of its workers at large firms:

```
ge bigfirmdummy = 0  
replace bigfirmdummy = 1 if bigfirm > .50
```

23) You can have two or more conditions following the **if** operator. Use **&** for “and”. Use **|** for “or”.

- E.g. **if a > b & a > 200** means “if a is greater than b and a is greater than 200”

24) Note that when it comes to evaluating **if** statements, Stata treats missing values as equaling positive infinity. Thus, **ge newvar = 5 if x > 100** is satisfied if x is missing. To deal with this, add **replace newvar = . if x == .**