Mgmt 469

Helpful Stata Commands

This contains virtually all the Stata commands you will ever need. You may find it helpful to experiment with them just to move more rapidly down the learning curve.

File Manipulation Commands

set mem xxm Allocates xx RAM for use by Stata. Must be done before you load up your data.

set mem xxm, permanently Always assigns XX RAM when you start Stata. You never need to perform this command again.

set matsize yyy "Reshapes" the available RAM to permit more variables into your model. The default for Intercooled Stata is 40 variables. You will probably need more than this for the ISP project.

use filename Reads in the Stata formatted file called filename.dta. Alternatively, you can use the "open file" button in the usual way.

insheet using textfile.txt Use this if you have created a tab delimited text file in Excel. Converts the file called textfile.txt into a Stata data file.

infile var1 var2 var3 using textfile.dat Reads in a plain text file named textfile.dat. In this example, the plain text file has three variables, named var1 var2 and var3.

save filename Saves a stata data set called filename.dta

save filename, replace Replaces an existing stata data set with a new version. Use this to save a file that you have already been working on. Tip: Save an original version of your data.

clear Clears the workspace to permit you to enter new data

drop var1 var2 var3 Drops these three variables from the data

keep var4 var5 var6 Keeps these three variables (and drops the rest!)

sort varname Sorts the data set in ascending order by the variable varname. (E.g., the data set yogurtsmall.dta has been sorted by week. The first week of data appears in the first row, etc.)

Merging Data Sets

merge varname using newfile This merges the active file with another file called the using file (The active file is the one you are currently working with. In this example, the using file is called newfile). The two data sets are matched to each other using the variable called varname. Before you merge, you need to sort both data sets by the matching variable:

After merging, you will see a new variable called _merge. Stata created this variable to help you take stock of the merge. It will take on one of three values:

_merge = 1 if the observation was in the active file data but not in the using data
_merge = 2 if the observation was in the using data but not in the active data
_merge = 3 if the observation was in both data sets.

Prior to merging again, you will need to **drop _merge**

Data Description Commands

label var varname "description of variable that you supply" I strongly recommend that you label all variables that you create. This will keep you from forgetting what each variable means.

de Describes all the variables. Stata will list all the variables in your data set and give tyeir properties (such as whether they are alphanumeric). Displays the labels for each variable.

de varname Describes varname, rather than all the variables

de var* Describes all variables that start with the letters var. (The * is a "wildcard". You can use * in almost any command to avoid typing in a long list of variables. E.g., **su z*** will generate summary statistics for all variables whose names begin with the letter z.)

list varname1 varname2 … Displays the values of the selected variables for all observations in your data.

Basic Statistics

su varname1 varname2 varname3 generates basic summary statistics such as mean, variance, and sample size for a list of variables.

su varname, de generates detailed summary statistics, including median and percentiles of the distribution, for one variable at a time.

corr varname1 varname2 generates a simple correlation table You can list as many variables as you like.

pwcorr varname1 varname2, sig generates a correlation table with significance levels

pwcorr varname1 varname2, star(.05) Puts a * next to correlations that are significant at p=.05. (You can choose some other significance level, of course.)

tab varname tabulates varname (that is, it lists all the values in ascending order, as well as their frequency of occurring in the data.)

table varname1 varname2 generates a two-way table. If varname1 takes on M possible values and varname2 takes on N possible values, you will get an M x N table.

table varname1, c(mean varname2) reports the mean of varname2, broken out by categorical varname1. For example, suppose you are working with the yogurtall data. If you type **table store, c(mean price1)**, you will get the mean value of price1 by store. Other statistics such as median, max, min, and sd (standard deviation) may be substituted for mean.

Creating and changing variables

ge newvar = varname1+varname2 Generate a new variable. Almost any mathematical expression is possible

replace oldvar=oldvar-2 Change the value of an existing variable

The **egen** command computes a summary statistic for all observations that belong to a group. See the last section of this document for more information about **egen**.

Conditional commands

Use the **if** statement to execute a command for a subset of the data.

replace oldvar =oldvar*othervar if oldvar <5 Change the value of oldvar only if the initial value of oldvar is less than 5.

The relational operators for the **if** statement are = =, >=, <+, and ~=.

You may combine conditions in the **if** statement. Note that & is “and”, | is “or”, and remember that “.” represents missing values.

replace oldvar =oldvar*othervar if oldvar <5&othervar~=. (& is "and".)

replace oldvar =oldvar*othervar if oldvar <5|othervar>10 (| is "or")

The **if** statement can be used to modify virtually any command. For example:

su varname if varname >100

This will generate summary statistics for varname, but only for those values where varname>100

Note: Stata considers missing values to equal ∞ for the purposes of evaluating **if** statements.

Regression and related commands

regress depvar predvar1 predvar2 Run a regression. The dependent variable comes first.

regress depvar predvar1 predvar2 if predvar2 >=20 Run the regression on the observations for which predvar is greater than or equal to 20.

You can recover both the predicted values and residuals from your regression. After a regression, type

predict varname to generate a new variable, varname, which equals the predicted values from the regression. You can, of course, select any name you wish for this variable.

The **predict** command also enables you to generate residuals after a regression:

predict resname, residuals

The new variable resname will contain the values of the residuals. (You can, of course choose any name for this variable that you wish.)

If you wish to generate a set of dummy variables in your model, use the xi: prefix

xi: regress depvar predvar1 predvar2 i.categoricalvar

Stata will generate a set of categorical dummy variables. If categoricalvar takes on N values, Stata will generate N-1 dummies. Be sure to interpret all the coefficients relative to the omitted category!

Note that the logic for other regression commands we will learn in class, including **logit, poisson, nbreg,** and **oprobit**, is similar to that for **regress**.

You may want to see if a group of variables belong in a regression. After the regression, type

test varlist where varlist is the list of variables you are testing

A useful interpretation of this test is as follows: "Did the variables *collectively* add more predictive power to the model than would have been expected from the same number of random variables?"

You can also compare the coefficients of different predictors to see if you can reject the null hypothesis that the coefficients are equal. Try:

test var1 = var2

This tests whether the coefficient on var1 is different from the coefficient on var2.

You can test whether the coefficient on var2 is twice that of var1:

test 2*var1 = var2

or if one coefficient equals the negative of the other:

test var1 = -var2

In fact, you can test any algebraic relationship among coefficients!

Again, the logic is the same for **logit** and other commands.

Advanced commands for working with grouped data

1) To create a group level variable but retain the same level of observation

egen newvar = fxn(oldvar), by(groupname)

The egen command allows you to compute a function such as the mean or the minimum, at the group level. It is best explained by means of an example. Suppose you have wage data on two firms in each of three markets.

firm	market	wage
1	Chicago	10
2	Chicago	11
3	Chicago	12
4	Boston	10
5	Boston	12
6	Boston	14

Suppose you want to create a variable that equals the mean wage, by market. If you have variables named wage and market, then you would type:

egen avgwage=mean(wage), by(market)

Your new data set will look like this:

firm	market	wage	avgwage
1	Chicago	10	11
2	Chicago	11	11
3	Chicago	12	11
4	Boston	10	12
5	Boston	12	12
6	Boston	14	12

In addition to computing the mean, **egen** allows you to use the following functions: min, max, median, sum, sd (standard deviation within the group), sum, count (the number of observations in the group), and many others described in the manual.

When you use the **egen** command, the number of observations remains unchanged.

2) To perform a similar group level calculation, but collapse the data so that the unit of observation becomes the group, you need to perform the **collapse** command. The syntax is a bit goofy, so watch carefully!

collapse (mean) avgwage =wage, by(market)

Your new data set will be

market	avgwage
Chicago	11
Boston	12

As with **egen**, you can use many other functions besides mean. Unlike **egen**, you can compute several different functions of several different variables, all at the same time. For example,

collapse (mean) avgwage =wage (min) minwage=wage, by(market)

will generate the following data:

market	Avgwage	minwage
Chicago	11	10
Boston	12	10