Introduction to Stata (part 3)

Fall 1999

Table of Contents pg.

Reminder about missing values:

Stata stores missing values in computer memory as very large positive numbers. For the most part, this does not affect your programming, except when you would like to subset using commands such as "greater than." Some stata commands, such as summarize, automatically ignore missing values. This means that if you type **summarize age**, and 3 of 100 individuals are missing age information, the number of observations reported back to you by stata will be 97.

Unfortunately, the subsetting commands don't work this way, so that if you type **count if age>0**, stata will tell you that there are 100 individuals with ages greater than 0, even though we know 3 are missing. Instead, type **count if age>0 & age!=.**, and stata will report that there are 97 individuals with ages greater than 0 (A silly, but hopefully illustrative example).

do files

Often you will want to repeat a set of commands. Rather then typing the commands in every time, you can create a program to do what you want. In Stata, such programs are called "do- files". To create a do-file you simply type in a series of commands and save them in a text file (you can use Notepad or any other text editor for this purpose. If you use Word, be sure to save the file as a text-only file, not as a Word document. There is also an editor included with Stata - click on Do-file Editor under the "Window" menu.). Once you have saved a series of commands, you just type "`do`" and the filename in Stata. For example,

```
do "a:fevanalysis.txt"
```

Submitting your commands to Stata in the form of a do-file is a good habit to get into early. It forces you to think about your analysis in a more comprehensive way, rather than command-by-command. More importantly, though, it will save you time when you need to repeat similar analyses. For example, you may do an entire analysis and then discover that there was one mistake in your dataset which has changed your results; using a do-file allows you to fix the mistake and then rerun the entire analysis in one step.

A sample do file:

```
log using c:/b536/logfiles/lbw.descrip.log
use c:/datasets/lbw.dta

/* This is one way to write comments that are simply notes to yourself and will not actually be submitted to
stata as commands.   These notes can exceed one line, but you must remember to type the asterisk and slash
at the end*/

*Another way to write comments on one line only
*Just type an asterisk at the beginning of the line

/*the next command tells stata to scroll through the entire program so that you don't need to hit any keys to
continue processing the do file or seeing results on the screen*/
```

```
set more off
*logistic regression using age as continuous variable
logistic low age
logit
gen agecat=group(4)

*logistic regression using age as a categorical variable
xi:logistic low i.age
log close
```

Another useful programming practice is to include comments in your do-files that remind you what you are doing. If you come back to this analysis or do file several weeks or months later, it will be much easier to figure out exactly what you did, and thus to understand the results.

Post-estimation commands

Although Stata does not generally remember anything about your analysis that you don't save into a new variable, it does have short-term temporary memory which can be accessed using special commands. After estimating linear regression parameters using the command **regress yvar xvar**, for example, you might like to use results from that regression analysis to examine residual values, predicted values, etc. Depending on the specific command, there are specific post-estimation commands that apply only to the most recently estimated model.

Continuing with a linear regression example, after fitting your model using **regress yvar xvar**, you would type **predict varname, options**. The varname is a new variable that will be generated using results from the linear regression model you just ran. The default option is the fitted value – so if you don't specify any options, than the new variable will contain the predicted value for each person in the dataset based on the regression parameters. Typing **predict varname, rstudent** will create a new variable with the jackknifed residual values for each individual, which can then be used to create plots to evaluate regression diagnostics.

Other options stata allows you to specify with linear regression include: **leverage, residuals, rstandard** (for standardized but not studentized residuals), and **dfbeta(varname)**. A more complete list of options can be found in the reference manual or on online help.

With logistic regression, we will often want to perform likelihood ratio tests, and these can be calculated using post-estimation commands. After using the command **logistic yvar xvar**, type one of the following commands (or a series of commands):

lrtest, saving(#)
lrtest, using(0) model(1)

The **lrtest** command specifies different actions depending on the options that follow the comma. If the option says **saving**, then stata saves information from that model that will later be used to perform a likelihood ratio test. You must specify the number (which you can think of as a sort of variable name – you are naming the model by that number, though you won't see it when you browse the data). If the option says **using(#) model(#)**, then stata will actually perform the likelihood ratio test, where you are **using** the full model and comparing to the reduced **model** (so your full model number goes first in this command, and your reduced model number goes second). If you've confused the numbers of your two models, you will geta negative values for the likelihood ratio test statistic, which makes no sense in the context of a chi-square distribution.

As with linear regression, residuals can be accessed with post-estimation commands which you will find listed in the reference manual.

Quick reference: Functions

Mathematical functions:
abs(varname) *takes the absolute value*
exp(varname) *exponentiates*
ln(varname) *takes natural log*
log(varname) *takes natural log*
log10(varname) *takes log base 10*
sqrt(varname) *takes square root*
varname^x *raises varname to the xth power*

Statistical functions:
uniform() *chooses a number randomly between 0 and 1.*

Date functions:
mdy(m,d, y) *assigns an elapsed time using three separate variables for month, day, and year*

Special functions:
int(varname) *rounds values of varname to the next lowest integer value*
max(varname) *assigns the maximum value of that variable to a new variable*
min(varname) *assigns the minimum value of varname to a new variable that you create*
missing(varname) *evaluates to 1 if the variable is missing for an individual or 0 if it is not missing*
sum(varname) *keeps a running sum of that variable's values*

Reminder about xi:
Prefacing a command with **xi:** prepares Stata to create dummy variables and/or interaction terms. I have only ever used this preface with regression commands, though it works with other commands as well. When you preface your commands with this interaction expansion option, stata will expect some of the variables in the command to also be prefaced as in ***i.varname***. Whichever variable is prefaced in this way should already be a categorical variable (meaning not continuous). Stata will create dummy variables for k-1 categories, using the lowest category as the reference group.

This command can also create multiplicative interaction terms automatically by typing an asterisk between the variable names.

xi: regress systolic i.agecat *performs linear regression of systolic on age categories.*
xi: regress systolic i.agecat*i.income *performs linear regression of systolic on age categories, on income categories, and on the multiplicative interaction between the two categorical variables*
xi: regress systolic i.income*age *performs linear regression of systolic on income categories, on age modelled continuously, and on the interaction between income categories and continuous age variable*
xi: logistic highbp i.agecat *performs logistic regression of highbp on age categories.*
.