Author: Prof. Dr. Joachim Winter

Prof. Dr. Iris Kesternich
KU Leuven
Winter Term 2015

Micro-Econometric Models

Introduction to Stata

Stata is a very powerful package for data analysis and econometrics, in particular for cross-sectional and panel data. Virtually all standard estimators in econometrics and also many advanced and relatively recent estimators can be invoked as 'black-box' routines – for example, when using maximum-likelihood estimators, the user rarely needs to bother with the numerics of maximizing the likelihood. While the basic syntax of Stata commands is simple and consistent, options give the advanced user additional control.

This document is really only a very brief introduction to Stata – the essence of Stata's own *User Guide*. Its aim is to provide new users with a quick reference for standard commands that are commonly used in data analysis. Users interested in specific commands for estimation of econometric models should refer directly to Stata's *Reference Manual* which contains details on the syntax of all commands, along with helpful examples and some mathematical background on statistical and econometric estimators.[1]

1 Starting Stata and managing its windows

Stata is started like any other Windows or Unix program.[2] Once Stata has come up, you see a standard Windows surface with four windows:

- *Stata command*
  In the command window, you can type commands one after another and execute them by hitting the return key.
- *Stata results*
  The results window displays the results of each command you've executed. Without further action by the user (explained below), these results are *not* saved.
- *Review*
  The review window contains a list of the last commands you've executed. By clicking on a command in the review window, it can be re-executed conveniently.
- *Variables*
  The variables window contains a list of all variables currently in memory.

[1] For some more recent estimators, the reference manual provides more detail than many econometrics textbooks while being more accessible than the original journal articles.

[2] Advanced users who work with large datasets on a regular basis should consider adding the **/k=x** optional argument to the command that invokes Stata, in order to assign x kilobytes of memory at start-up. The default memory assignment is not sufficient for large datasets.

There are a few other windows that pop up depending on context. These are 'viewer' windows that display graphics, help files, and other things. More on these windows below.

In general, the appearance of windows within the Stata surface can be controlled as in any other Windows program:

- The screen font size within a window can be changed by clicking on the little symbol in the top left-hand corner of the window's frame.
- The size of each window can be altered by grabbing and dragging a corner using the mouse. Windows can also be moved by grabbing and dragging the top bar.
- Windows can be closed by clicking on the symbol in the top right-hand corner.
- Windows can be opened using the 'Windows' drop-down menue.

Once you have started Stata, it will read and save all files in its startup directory by default. To see the startup directory, type **cd** in the command window. If this is not the directory where you want to keep your files, you can change to another directory using the **cd** command and standard DOS or UNIX syntax. There's another basic DOS command that also works in the command window: Type **dir** and the contents of the current directory will be listed.[3]

2 Basic command syntax and online-help

Everything in Stata can be done using commands which are typed one after another in the command window.[4]

Some commands – mostly concerned with file handling and with saving, viewing or printing of results – can also be invoked using Windows-style drop-down menues, but all commands for substantive data analysis are entered using the command-line interface. This introduction focuses on command-line commands. Figuring out how the drop-down menues work should be easy for Windows users.

The basic syntax of Stata commands is as follows:

command *list of variables* conditions , options

At a mininum, the command and a list of variables need to be specified. A list of variables contains the names of one or more variables, separated by blanks.

Conditions and options are optional. In addition, many commands allow the use of weights and a so-called 'by-option' (both of which are not covered in this introduction).

Some options can be used with many commands, others are specific to individual commands. For each command, the reference manual contains an entry that discusses all

[3] To invoke other shell commands, either DOS or UNIX, from within Stata, start the command line with a '!' and then enter the shell command – but only if you know what you're doing. Stata cannot prevent users from making mistakes such as deleting files when they do it using shell commands.

[4] A sequence of commands can also be combined in little programs – more on this below.

options available for that command. The essence of these entries is also available within Stata. There are several ways to access this information:

- When you know the name of a comamnd, type **help command** in the command window. The description of the command is displayed in the results window.
- Alternatively, open the 'Help' drop-down menue, select 'Stata command' and enter the command name. The description of the command is displayed in a new 'view' window.
- If you don't know the name of a command, you can search for a keyword either by typing **search** *keyword* in the command window, or by using the the 'Help' drop-down menue and selecting 'Search'. For example, searching for 'regression' displays a list of all help entries concerned with regression (which are quite a few).

Finally, many commands can be abbreviated. For example, to execute the command **summarize** (to be explained below), it is sufficient to type only its first two letters, **su**, since no other commands that begin with 'su' exist. Stata accepts abbreviated commands as long as there's no ambiguity. In the online and printed manuals, those letters that are the minimal unique abbreviation of a command are underlined.

3 Entering data and handling existing datasets

Stata can read and save data in two different ways, either using an internal, binary format, or as ASCII text files.

Stata's own datasets have the filename extension '.dta'. You can open a dataset by typing **use** *filename* (you need not specify the extension). Stata will look in the current directory. If the file is not in the current directory, you either have to **cd**, or you can specify the entire path in the **use** command.

When you make changes to your data, you may want to save the file at some point. This can be done by typing **save** *filename*. Stata will not allow you to overwrite an existing file with the same name. If you are sure that you want to overwrite the file, then type

save *filename* , replace .

This is an example of standard Stata syntax, with a command, a filename (rather than a list of variables, but conceptually, that's the same), and an option.

Before you save a dataset, it is helpful, but not necessary, to type **compress**. This command does exactly what its name says – it can result in substantially smaller file size (without loss of information).[5]

[5] Technically, variables are generated as double precision numbers by default. This is a waste of space in many cases (say, if it's just a dummy variable). Stata's **compress** command assigns the appropriate minimal precision storage type to all variables.

Once you have opened a Stata dataset, you might want to check what it actually contains. You can do this by typing **describe**. This produces a list of all variables in the dataset together with storage formats and some other information.

To look at the data, you can use an internal data browser/editor. The browser is safer since it does not allow you to change the data by accident. If you want to change some data item, you can use the editor. (Actually, you probably don't want to do this since such changes are not documented and you might forget after a while what and why you have altered your data, which is not good practice.) The browser and editor are invoked from the command line by typing **browse** and **edit**, respectively (or by using the Windows-style buttons in the top panel). Like all other commands in Stata, both commands refer to the dataset currently in memory. If there is no dataset in memory, you get an empty editor. You can enter new data by using the editor, and then save the file.

The browser/editor also shows how data are usually organized: Variables down columns, observations (i. e., data on individuals, firms, countries) across rows. If some data item is missing, it is represented by a '.' (more on missing data below).

If you want to clear a dataset from memory, you can type **clear**. If you have changed the dataset, Stata will prompt you to save your data before clearing memory.

For reading ASCII datasets, there are several different commands, depending on how the data are organized in the ASCII file. In the most simple case, when the data has the variables in fixed-column format, with observations in rows, you can use[6]

infile *list of variables* **using** *filename.asc*

Similarly, you can save a dataset as an ASCII file with fixed-column format by typing

outfile *list of variables* **using** *filename.asc*

However, if you do not plan to work with your data in a different programm, using Stata's internal binary format is faster and generates smaller files.

4 Some basic commands

Now you have some dataset in memory. Here is a list of basic commands that are useful for analyzing data. The following description is very brief – these commands are best learned by using them (rather than by reading about them). Many of these commands have specific options, and it is always helpful to consider the reference manual (either online or in print) to learn more.

summarize *varlist*

Lists some descriptive statistics for all variables in *varlist*.

summarize *varlist* , detail

Similar, but with more detailed information.

tabulate *varname*

For a discrete variable, this produces the frequency counts of all values.

[6] There exist alternative commands for reading ASCII files that are organized differently (the reference manual has details).

tabulate *varname1 varname2*
For two discrete variables, this produces a cross-tabulation.

correlate *varlist*
Produces the correlation matrix for a group of variables.

pwcorr *varlist*
Produces pairwise correlation coefficients. The main difference to **correlate** is how missing observations are handled; see the reference manual. A helpful option is **sig** which produces significane levels along with the correlations.

spearman *varname1 varname2* or **ktau** *varname1 varname2*
If at least one of two variables is discrete (and of course ordinal), you cannot use standard correlations – rather, using one of these two commands, you should produce Spearman or Kendall rank correlation coefficients.

5 Graphics

Stata has very powerful graphics capabilities – the manual has one volume just on graphics. Since the syntax can be quite complex, it often makes sense to first work with the "graphics" menu.

Here is a list of Stata 11 graphics commands:

histogram *varname*
This produces a histogram. Among the many options, try **bin**(n) which changes the number of bars to n.

scatter *varlist* xvar
This produces a scatterplot of two variables. The last variable *xvar* is on the horizontal axis.

line *varlist* xvar
Just like **scatter**, but connects the points with a line.[7]

twoway *(command1) (command2) ...*
This is a way to overlay several graphs. *command1, command2, ...* are graphic commands such as **scatter** or **line**. As an example, try **twoway** (**scatter** *yvar xvar*) (**lfit** *yvar xvar*).

6 Regression commands

Stata provides a fair number of regression commands: standard linear (OLS) and nonliner regression, maximum likelihood estimators for discrete dependent variables such as logit and probit, and many other estimators such as generalized method of moments (GMM) estimators.

[7] For reasons related to how Stata handles data internally, you need to sort the dataset before the connect option works correctly. You can do this by typing **sort** *varname*, where the sorting variable specified by *varname* is the variable that goes on the horizontal axis (say, time).

The basic syntax for all these commands is as follows:

regress *depvar xvarlist*

Regression commands begin with a command that specifies the estimator to be used (e. g., **regress** for linear regression or **logit** for a binary logit model), followed by a list of variables that begins with the dependent variable and then lists the explanatory variables. For most estimators, Stata adds a constant by default, so you don't have to specify it explicitly (but you can request that no constant be used by adding the appropriate option).

7 Generating and changing variables

New variables can be generated by using functions of existing variables.

generate *newvar = some funtion of existing variables*

For example, you could generate real GDP as the ratio of nominal GDP and a price index by typing **generate** *realgdp = gdp / price*.

The syntax for functions is pretty standard. You can add, substract, multiply and divide variables by using $+$, $-$, $*$, and $/$, and there are many other special mathematical and statistical functions – typing **help functions** produces a list of these.

Instead of generating a new variable, you can also replace the values of an existing variable by typing **replace** instead of **generate** (with the same syntax).

You can rename variables using **rename** *oldname newname*, and you can drop obsolete variables from your dataset by using **drop** *varlist*.

8 If conditions

By default, Stata commands are executed for all observations in the dataset that is currently in memory (that is, row by row).[8] As mentioned earlier, most commands allow adding an if condition (recall that in the command line, the if condition goes after the variable list, but before the comma, while options go after the comma). An if condition selects only those observations for which the condition is evaluated as true; all other observations are not used or unaffected by the command.

For example, suppose you have data on many people, including income and age. Then

summarize *income* if age > 40

produces summary statistics only for the subset of observations for which the age variable is larger than 40 (for people older than 40), while

replace *income = 10000* if income < 10000

assigns an income of 10000 to everybody who has an income of less than 10000. (Unfortunately, this won't work in real life.)

[8] Missing values are handled by Stata, but not always in an intuitive way; see below.

Logical expressions in if conditions use the following operators: '==', '>', '<', '≥', '≤', '! =' and '∼=' (the last two both meaning not equal).

Two expressions can be combined using '&', '|', '!' (which stand for and, or, not, respectively). The use of parentheses is standard.

9 Log files

You can use log files to preserve Stata's output. A log file is opened by typing

log using *filename* , replace

The replace option is only necessary if a log file with the same name exists and you want to overwrite it. An open log file is closed by typing **log close**. Closing log files once you're done is important, since you don't want to have open files hanging around (and you can't view or print open files).

You can select the filename for the log file yourself. By default, Stata generates the log file using its own internal SMCL format, adding the extension '.smcl'. Such a log file can be viewed by typing **view** *filename*, and it can be printed to the standard printer by typing **print** *filename*.

Since the SMCL log files contain formatting characters, they are not of much use in other programms (while they look quite nice on screen and if printed). If you want to produce a plain ASCII log file, type **set logtype text** before you open a log file. By default, such a log file will get the extension '.log'.

As any other Stata files, log files are saved in the current directory by default. If you want to save your logfiles elsewhere, you can specify a path.

10 Stata programs

In real-world projects, it is important to document your work so that you can either replicate it, or if something has gone wrong, start over without having to re-type all commands entered so far. Batch files are a good way to work with Stata. Batch files are plain ASCII text files, they have the extension '.do' and they can be generated and edited with your preferred editor or with Stata's internal editor (accessible using the button in the top panel).

A batch file is a little program with one Stata command in each line. By default, a carriage return is used by Stata to recognize the end of a command. This implies that commands cannot extend over more than one line.[9]

Once you have written a program, you can execute it by typing **do** *filename*. In a real-world project, it is a good idea to open a log file at the beginning of a program so that

[9] Actually, they can. If you type **#delimit ;** Stata knows that a command is delimited by ';' and that it can extend over more than one line. By typing **#delimit cr** you switch back to the default with a carriage return as command delimiter.

you preserve your output. And don't forget to add a command that closes the log file at the end.

11 Missing data

If you plan to use Stata seriously with real-world data, you need learn how Stata deals with missing data when it computes statistics and regressions – it's not always intuitive. Without going into detail, the missing data issue can be tricky and needs some consideration.

A peculiar problem results from the fact that Stata represents a missing value internally with the largest number allowed by a particular storage type. Usually, the user doesn't notice this and need not bother, but using the '>' operator in if-conditions can produce unwanted results when data contain missing values. Stata's *User Guide* has more details on these issues.