Stata For Dummies

Christopher Zorn
University of South Carolina
Oxford Spring School
June 18-20, 2007

Table of Contents

1 Introduction

Stata is a statistical software package we'll be using for much of this course.[1] Stata has a number of advantages over other currently available software. For example, Stata keeps all data in memory; this makes it very, very fast. Stata is also largely command–driven (as opposed to menu–driven); while this makes its learning curve a bit steeper, once you learn Stata you will find it is substantially faster, easier, and more flexible in use than its competitors. It also has excellent manuals, on–line help, and user support, both via phone and e–mail. Additionally, Stata offers a number of more advanced techniques not available in other packages. Finally, compared to its competitors Stata is actually somewhat affordable; this means that, should you find yourself in a place where Stata is not widely used, it is feasible for you to simply purchase your own copy.

This guide is intended only as a handy reference for use as you learn the basics of Stata. I strongly recommend that you also read *Getting Started with Stata for Windows, Release 8*, and that you familiarize yourself with the Stata *Reference Manuals* as well.

[1] Of course, you're more than welcome to use any software you'd care to, including (say) R; but Stata will do everything we cover in this course.

2 Things You Need To Know

Stata is not a hard program to learn, or to use. Before you begin working with Stata, here are some basic things to keep in mind.

- Stata's manuals are extremely comprehensive: in addition to assistance about capabilities, procedures, commands, etc. they also often offer illustrative examples of commands, and even tips for using and interpreting different statistical techniques. The *Reference Manuals* (four volumes) are arranged alphabetically by topic. In addition, there are several other useful texts for Stata users, including the *User's Guide*, *Getting Started with Stata 8.0 for Windows*, and, of particular interest for this course, Stata's *Survival Analysis & Epidemiological Tables* and *Cross-Sectional Time-Series Reference Manuals*.

- Stata's on-line help facility is easily the best available. In essence, Stata has put the entire set of manuals into its help files. This meals that by typing "`help <topic>`" on the command line (or simply pressing the `<F1>` key), you can obtain a wealth of information on the procedure you are interested in. An even more useful way of finding out what Stata is capable of is Stata's `-findit-` command, which is an extremely comprehensive Stata-specific search engine and which can be accessed simply by typing "`findit <topic>`" at the command line. You are advised to read the section on *Help* (Chapter 4) in *Getting Started with Stata* closely. The `help` facility in Stata is your best friend; in the event of confusion, it should always be your first resort.

- Stata also has a comprehensive WWW site (`http://www.stata.com`). Particularly once you begin using Stata for your own research, this site is a valuable place to find out about the capabilities of Stata. There is also a Stata listserv; directions for subscribing can be found at the website. It is not necessary (nor even recommended) that you to subscribe to the listserv for this course, but you may wish to do so later.

- Your instructor has used Stata very extensively since 1994 or so. I know quite a bit about the software, and can probably answer any questions you might have at this stage.

- Stata's technical support is available through e-mail. You can send questions to `tech@stata.com`, and the good people at Stata will generally respond within 24 hours. I recommend this only as a last resort, to be used after consulting the help utility, manuals, your colleagues and your instructor.

3 Starting Stata

You will be using Stata 9.0 for Windows. This software is available on PC's in the data lab, under "Statistical Packages." The program is called "WStata" (for "Windows Stata") or "Stata 9.0." The icon is an odd-looking little grey box with some green lines and the word

"STATA" in it. Double-click on this to start the program.

When you do so, you will be greeted by four windows in the program. In the upper left is the "Review" window; we'll return to this shortly. Below that is the "Variables" window. the largest window is for "Stata Results"; this is, appropriately, where the results of your Stata commands are displayed. Below that you will find the "Stata Command" (or just "Command") window. This last window is a single line (we may also refer to it as the "command line"), and will have a blinking cursor in it. Stata is waiting, patiently, for you to tell it to do something.

In addition to the four windows, there will also be a series of buttons across the top. In addition to the "File", "Edit", etc. pull-down menus familiar to Windows users, these buttons allow for quick implementation of commonly-used features and commands. More on this below.

4 Entering Commands

In order to tell Stata what to do, you need to enter a command. To do this, you simply type the command on the command line and press the "Enter" key. I'll denote commands in boldface, following a period (.). So, if I wanted you to type "SPSS bites" in the command window, we'd write:

```
. SPSS bites
```

You need not type the period at the beginning; this is merely how the command you enter will appear in the Results window. (If you actually type this particular command, Stata will agree with you, but give you an error message nonetheless). Entering a command on the command line and pressing "Enter" does at least two things: it executes the command, and it displays the results of that command in the display window. As an example, type:

```
. version
```

The Results window will now display:

```
. version
version 9.2
```

The basic syntax of Stata is pretty simple:

```
. command variable (variable variable ...), options
```

So, if you wanted to regress a variable ineptly named `Y` on two other ineptly named variables `X1` and `X2`, you would type:

```
. regress Y X1 X2
```

Note the absence of commas. Commas are used if you want to add "options" to your command, options being exactly that. So if you wanted to do a crosstable of `Y` and `X1`, and also wanted to know the χ^2 statistic for that crosstab, you would enter:

```
. tab2 Y X1, chi
```

Also note that Stata's variable names, and most commands, are case-sensitive: it is perfectly possible to have three variables named `PARTYID`, `partyid` and `PartyID` in the same dataset. An interesting and useful trait of Stata is that, while many of the commands can be executed either by typing them or using the pull-down menus, in many cases typing them is actually quicker and more efficient. For this reason, among others, I'll generally stick to illustrations of command-line entry of commands, rather than talking about the pull-down features; the latter are intuitive, and you can pick them up easily enough on your own.

5 Data Stuff

Stata is used to analyze quantitative data. Stata keeps the data you are using in resident memory. The up-side to this is that it makes Stata extremely fast, especially compared to memory-swappers like `SPSS` and `SAS`. The down-side is that, if you have the luxury of a lot of data, you must have the additional luxury of additional memory.

There are at least three ways of getting data into Stata. One is to enter it. To do this, either type:

```
. edit
```

in the command window, or click on the "Edit" button on the toolbar at the top. Either way, you'll be taken to a vaguely-spreadsheet-looking thing, which is Stata's data editor. This works pretty much like a (primitive) spreadsheet, with the annoying exception that one must hit `<Enter>` (as opposed to one of the arrow keys) to enter the data into the cell. (Also note: Stata's character to denote missing data is a period, "."). To leave the editor, just close the window, being sure to "Preserve" the changes you've made by clicking on the button of the same name. Note that the second way of getting data into Stata is simply to cut-and-paste data from some other source (e.g., a spreadsheet works nicely) into the data editor window.

The final way of getting data into Stata is to `infile` it. This is basically the way of bringing an already-created Stata dataset into the program. You can do this either by typing:

```
. infile (path) filename
```

in the command window, or pulling down the File menu and selecting Open. You're then given a standard Windows file dialog box.

In either event, once you've selected or inputted your data, the variable names and descriptions will appear in the Variables box. Some useful commands for getting to know your data are:

```
. d
```

Short for `-describe-`, this will list information about the data file currently in memory, including its name, description, and size, as well as variable names, storage types (e.g. strings, floats, etc.) and labels.

```
. su
```

Short for `-summarize-`, this command gives basic summary statistics on your variables: name, valid N, mean, standard deviation, minimum and maximum. It will show zero observations for any string (i.e., alphanumeric or alphabetic) variables in your data. Note that either of these commands will take a variable list. e.g.:

```
. su varname1 varname2 ...
```

which has the effect of only showing the relevant information for the variables listed.

```
. browse
```

This command is like `-edit-` except that you can't change the data. Its good if you just want to "look" at the data (hence the name).

```
. sort varname (varname varname...)
```

This command does what it says: it sorts the data from lowest to highest according to the values in variable. This can be important, since some data manipulation procedures will change the order of your data without even telling you. A good rule is to always have an ID variable, and to `-sort-` on it frequently. Multiple variables in the `-sort-` command sort on the first variable first, and then within values of the first `-sort-` variable, sort on the

second. So if you had data by country (`nationid`) and year, you might type:

```
. sort nationid year
```

to sort the data for most panel–data applications.

The `-generate-` (or `-gen-`) command is used to create variables. The general syntax is:

```
. gen varname = expression
```

The expression(s) in question can be numbers, other variables, or combinations thereof. Stata follows standard practice for operators:

`+` Addition	`-` Subtraction	`*` Multiplication
`/` Division	`^` Exponents/powers	`()` for parentheses

Stata also slavishly follows the correct order of operations; when in doubt, use parentheses to make sure of your calculations. As an example, suppose we had data on two variables X and Y, and we wanted to calculate a third variable $Z = \frac{\frac{4}{3}X^3 - 81Y^4 + 5}{31X - 7}$. We'd enter:

```
. gen Z = ((4/3)*X^3 - 81*Y^4 + 5^(1/2))/(31*X - 7)
```

and Stata would create a new variable `Z`, which would now appear in the *Variables* box at the left. If there is any missing data in any of the variables used in the expression, Stata generates a missing value for the new variable on that observation as well. There are other, more advanced things one can do with the `-gen-` command, but I'll leave it to you to explore these possibilities.

The `-recode-` command does just that; it recodes the values of variables into other values. The basic syntax is:

```
. recode varname value = value value = value ...
```

You of course may recode several categories into one, though it is good practice to generate a variable identical to the one being recoded before doing so in order to preserve the original categorizations:

```
. gen partyid2 = partyid
. recode partyid2 1=1 2=1 3=2 4=3 5=3
```

You can also recode missing data into other values and vice-versa, remembering that the symbol for missing data in Stata is a period:

```
. recode gnp -999 = .
```

This is almost always a good idea when importing data from (e.g.) SPSS, or any other program which assigns actual numeric values to missing data.

The `-replace-` command is a bit different from `-recode-`. `-replace-` changes the values of an existing variable, but allows you to do so according to an expression, rather than just changing values for values. The basic syntax is:

```
. replace varname = expression
```

where *expression* is essentially the same as that used in `-gen-`. `-replace-` is an amazingly useful command, especially when used in combination with conditional expressions such as `if` and `by` (see below).

6 Conditional Expressions

Stata provides a simple, consistent way to implement most of its commands conditionally; that is, for a subset or subsets of the data. Two of the most useful are the `-if-` and `-by-` subcommands. The `-if-` subcommand is used to select a subset of observations for use with the instant command. The general syntax is:

```
. command if expression & expression ...
```

As an example, suppose you have data on all U.N. nations, and you want to run a regression of `Y` on `X` for only those nations in the OECD. Assume further that you have a variable indicating OECD membership (1) or nonmembership (0). To do this, you use the `-if-` subcommand:

```
. regress Y X if OECD==1
```

This command will then only include those observations in the data for which the expression following the `-if-` subcommand is true. Note several things about the `-if-` subcommand:

- The `-if-` subcommand follows the command, but comes before any options (i.e., before a comma). If, for some incomprehensible reason, we wished Stata to report standardized (beta) coefficients in the above regression, we'd use:

  ```
  . regress Y X if OECD==1, beta
  ```

- The expression that follows if doesn't necessarily need to be an equality, but can be an inequality or even a formula. It is perfectly acceptable, for example, to use the

command:

```
. regress Y X1 if X2^2 < X3 - 3
```

- One generally needs to use two equality/inequality signs, of some sort, for the `-if-` subcommand to work. The equality/inequality terms Stata recognizes are:

`==` equals	`~=` or `!=` does not equal
`>` is greater than	`<` is less than
`>=` is greater than or equal to	`<=` is less than or equal to

  This means that typing expression `if X=1` will give you a syntax error; get used to using the double-equal sign (`==`) in your if statements.

- One can string together several conditions following an `-if-` subcommand by using the ampersand (`&`) symbol. So if we wanted our regression to only include OECD countries and only those countries with a GNP greater than $100 billion, we could enter:

```
. regress Y X if OECD==1 & GNPbill>100
```

  Likewise, we can use the "not" (`~` or `!`) and "or" (`|`) symbols to connect different expression following an `-if-` subcommand. See the help file for "operators" for more details on the use of operators.

While `-if-` is very powerful, it can also be limiting. Suppose we had survey data which included a variable for income, arranged into ordinal categories ("less than $10k", "$10k-$20k", "$20k-$30k", etc.), and we wanted to run separate regressions of `Y` on `X` for each category. One way to do this is to use separate `-if-` commands:

```
. regress Y X if income==1
. regress Y X if income==2
. regress Y X if income==3
```
etc.

A more efficient way to accomplish the same thing would be to use the `-by-` subcommand. The `-by-` subcommand essentially performs a separate command for each group of data defined by some particular variable. The basic syntax for `-by-` is:

```
. by varname : command ...
```

In our example, we would enter:

```
. by income : regress Y X
```

Stata would then estimate separate regressions of `Y` on `X` for each category of `income`, and report the results of each. An important thing to remember is that, in order for the `-by-` subcommand to work, the data must be sorted by the variable defining the categories. Stata will remind you if you forget to do this:

```
. by OECD : regress Y X
not sorted
r(5);

. sort OECD
. by OECD : regress Y X
(output...)
```

Also note that `-by-` can be combined with `-if-` for most commands, including data generation commands such as `-gen-` and `-replace-`. This set of commands make Stata a powerful program for data manipulation.

7 Basic Statistics

Stata will do just about any statistics you care to use. Here are a few of the basics:

```
. tab1 varlist
```

This is the one-way crosstab command; it presents a frequency table, complete with percentages for each category. Listing more than one variable yields multiple frequency tables. Two-way crosstabs are similar:

```
. tab2 varname1 varname2 (etc...), options
```

Listing more than two variables here will yield multiple two-way crosstabs, one for each possible pair of variables. Options include row, column and cell percentages, and measures of association (e.g. χ^2, γ, τ_b, Spearman's ρ, etc.). See the `-help-` for the `-table-` command for more information. Another useful basic statistic is:

```
. corr varlist, options
```

which generates a correlation matrix for the variables listed, using casewise deletion of missing data. Options include `covariance` for covariance (rather than Pearson's rs). For pairwise correlations using all available data, use:

```
. pwcorr varlist, options
```

Finally, basic linear (OLS) regression is performed using the `-regress-` command, which can be shortened to `-reg-`:

```
. reg depvar indvar1 indvar2 ..., options
```

8 Stata for Survival Models

Stata uses a series of commands beginning with `-st-` to implement the duration models we'll be learning in this class. I'll go into these in some detail as they arise; in the meantime, a partial list you may want to check out (either in the manuals or using the `-help-` command) includes:

`-stset-`	Declare data to be survival-time data.
`-stdes-`	Describe survival-time data.
`-stsum-`	Summarize survival-time data.
`-stvary-`	Report which variables vary over time
`-stfill-`	Fill in by carrying forward values of covariates.
`-stgen-`	Generate variables reflecting entire histories.
`-sts-`	Generate, graph, list, and test the survivor and cumulative hazard functions.
`-stir-`	Report incidence-rate comparison.
`-strate-`	Tabulate failure rate.
`-stcox-`	Estimate Cox proportional hazards model.
`-stphtest-`	Test of Cox proportional hazards assumption.
`-stphplot-`	Graphical assessment of the Cox prop. hazards assumption.
`-stcoxkm-`	Graphical assessment of the Cox prop. hazards assumption.
`-streg-`	Estimate parametric survival models (exponential, Weibull, gompertz, lognormal, loglogistic, gamma).
`-stcurv-`	Plot fitted survival functions.

9 Panel/TSCS Models

Stata's commands for panel and time-series cross-sectional (TSCS) data all begin with the letters `-xt-`. In addition, there are a few commands that we may encounter that also require that we use the `-ts-` (for "time-series") series of commands as well. Again, we'll discuss these as we go along, but a quick list of the most important ones is below.

`-iis-`	Denotes the variable identifying cross-sectional units.
`-tis-`	Denotes the variable identifying time points.
`-tsset-`	Same as `-iis-` and `-tis-`, in one command. Generally preferred to those.
`-xtdes-`	Describes TSCS data.
`-xtsum-`	Summarizes TSCS data.
`-xttab-`	Tabulates TSCS data.
`-xtreg-`	TSCS regression for continuous data – fixed– and random–effects models.
`-xtregar-`	Fixed and random effects models with AR(1) errors.
`-xtgls-`	GLS models for TSCS data.
`-xtabond2-`	Arellano–Bond dynamic TSCS model.
`-xtpcse-`	TSCS regression with "panel–corrected standard errors" (requires `-tsset-`).
`-xttobit-`	Random effects tobit (censored) regression.
`-xtprobit-`	Random–effects and GEE binary probit models.
`-xtlogit-`	Fixed–effects, random–effects and GEE binary logit models.
`-xtpois-`	Fixed–effects, random–effects and GEE Poisson (event–count) models.
`-xtgee-`	Generic command for GEE models.

10 Shortcuts, Features, etc.

These are a few nice things that make Stata more user-friendly.

- The `-d-` command informs you, after displaying variable names and labels, whether the data have changed since you last saved them. It is a good habit to do a `-d-` before shutting down Stata, just to make sure that you don't lose any important changes you've made. If you do not heed this advice, however...
- ...Stata will not let you exit without saving your data, if changes have been made to it.
- The `PageUp` and `PageDown` keys are your second-best friends (after, of course, the `-help-` command). Using `PageUp` in the command window will display the last command you ran; pressing it again will display the one before that, etc. Thus you can scroll up or down through past commands using these two keys. This means that Stata is actually easier and faster to use than menu-driven programs, since much data analysis is repeating the same or very similar commands. Relatedly, the *Review* window contains a list of past commands; you can click on these and they will appear, as if by magic, in the command window, ready to run.
- You can cut-and-paste from Stata's *Command* and *Results* windows directly to a word processor, if you use OS-X or Windows 95/98/NT/2000/XP/whatever.
- If (as I hope you do) you use LaTeX to typeset your results, a useful command is called `-outtex-`. Issued after any other command, this command takes the results from the

previous command and formats them for cutting-and-pasting into LaTeX. So:

```
. reg Y X

  Source |       SS       df       MS      Number of obs =     100
---------+------------------------------   F(  1,    98) = 2036.32
   Model |  78.9979079     1  78.9979079   Prob > F      =  0.0000
Residual |   3.8018611    98  .038794501   R-squared     =  0.9541
---------+------------------------------   Adj R-squared =  0.9536
   Total |   82.799769    99  .836361303   Root MSE      =  .19696

------------------------------------------------------------------------------
       Y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
---------+--------------------------------------------------------------------
       X |   .9501933   .0210566    45.13   0.000     .9084071    .9919795
   _cons |   1.147475   .0647553    17.72   0.000     1.018971     1.27598
------------------------------------------------------------------------------

. outtex

%------- Begin LaTeX code -------%

{
\begin{table}[htbp]\centering
 \caption{Estimation results : regress
\label{tabresult regress}}
\begin{tabular}{l c c }\hline\hline
\multicolumn{1}{c}
{\textbf{Variable}}
 & {\textbf{Coefficient}}  & \textbf{(Std. Err.)} \\ \hline
X  &  0.950  & (0.021)\\
Intercept  &  1.147  & (0.065)\\
\hline
\end{tabular}
\end{table}
}

%------- End LaTeX code -------%
```

Finally, rest assured that the best way to learn Stata is to use it. If you are new to Stata, I encourage you to get on the computer and "play around" with the software; run a few nonsense regressions on made-up data, look into some `-help-` files, and learn by doing.