Two Page Stata

An Introduction to Stata

Andrew Grogan-Kaylor

2023-09-23

1 Introduction

An introduction to Stata in 2 pages.[1] Commands that you actually type into Stata are represented in `monospace font`. x and y refer to variables in your data. The treatment here is intended to be extremely brief, in order to create a kind of "cheat sheet" that can be presented in 2 pages. More documentation on any command is available in the printed or PDF Stata manuals, or by typing `help command`.

So many projects have the same, or similar, workflow.

$$\text{have a question} \rightarrow \text{get data} \rightarrow \text{process and clean data} \rightarrow$$
$$\text{analyze data} \rightarrow \text{visualize data} \rightarrow \text{make conclusions}$$

A little bit of Stata can go a long way.

The general idea of most Stata commands is `command variable(s), options`. Often it is not necessary to use any options since the authors of Stata have done such a good job of thinking about the defaults.

The Stata interface makes it extremely easy to do rapid interactive data analysis. Hit **PAGE-UP** to recall the most recent command, which you can then quickly edit and resubmit.

Use the **DO FILE EDITOR** to save Stata commands that you want to use again in a `.do` file, and to create an *audit trail* of your work so that your workflow is *documented* and *replicable*. `log using filename, replace` will save a log file of your results. `log close` closes the log file.

2 Get Acquainted With Your Data

Once you are in the right directory, `use "mydata.dta"` will open your data. `save "mydata.dta", replace` will save your data.

It is good practice to start by *looking at* your data. `browse x y z` will open a data window with variables `x`, `y` and `z`.

`codebook x y` will produce a nicely formatted codebook of selected variables, which is especially useful if you have added variable labels and value labels. `codebook` is especially useful for seeing how numerical values are associated with value labels. `codebook` by itself will list every variable in your data and generate a lot of [probably too much] output.

`lookfor` allows you to find variables that contain a specified keyword. This is especially useful in large data sets with many variables. Often abbreviated keywords are the most helpful. e.g. to find a poverty variable, type `lookfor pov`.

With very large data sets, it may be helpful to use `keep x y z` to only keep the variables with which you are working.

`describe` tells you about the contents of a specific variable. E.g. `describe x y`. `describe, short` will tell you very basic things about your data, including the number of observations in the data set, and the size of your data file.

[1] Comments, questions and corrections most welcome and may be sent to: Andrew Grogan-Kaylor @ agrogan@umich.edu. This document available on the web @ https://agrogan1.github.io/Stata/

3 Process Your Data

Data with missing values, often represented as negative numbers (e.g. -99, -9, -8) need to be recoded so that the missing values are represented as a missing value character (".") that Stata knows to exclude from calculations.

`recode x (oldvalue = newvalue), generate(z)` will recode a variable into a *new* variable, **often a good idea**. `recode x y z (-99/-1 = .)` will recode negative numbers from -99 to -1 to missing for x, y and z variables in your data. `recode x (7/9 = .)` changes 7 through 9 to be missing for x. Indeed, `recode` will change specific values in your data to anything you want, not just missing values.[2]

It is often convenient to `rename` your variables so that the variables have more intuitively understandable names e.g. `rename x depression`.

You can create new variables out of old variables using `generate newvar = expression` e.g. `generate newvar = oldvar1 + oldvar2`.[3]

It is sometimes useful to `sort` your data. `sort x` will sort your data by the values of x.

4 Descriptive Statistics

`summarize` gives you basic descriptive statistics for a variable, such as the mean (average). Especially useful for continuous variables. E.g. `summarize x y` or `summarize x y, detail`. `tabulate` gives you a frequency distribution for your variable. Especially useful for categorical variables. e.g. `tabulate x`.

5 Bivariate Statistics

Tabulating two categorical variables together gives you a cross-tabulation of those variables, e.g `tabulate x y, row col chi2`. `pwcorr x y, sig` gives you the pairwise correlation of two continuous variables. `oneway x z, tabulate` gives you a oneway ANOVA of continuous variable x over categorical variable z.

6 Multivariate Statistics

`regress y x` regresses y on x.[4] `regress y x z` regresses y on x and z.[5] `regress y x i.z` regresses y on x and z, treating x as continuous and z as a set of categorical indicator variables.[6] `regress y c.x##i.z` regresses y on continuous x and categorical z, providing both main effects for x and z and the interaction of x and z.

7 Graphing[7] [8]

`histogram x` will give you a nice display of one variable.[9]

`twoway scatter y x` gives you a scatterplot of your data. `twoway lfit y x` will give you a linear fit graph. The two syntaxes may be combined e.g. `twoway (scatter y x) (lfit y x)`.

`graph bar, over(x)` is useful for creating a bar graph of the counts of a categorical variable x. `graph bar y, over(x)` will create a bar graph of the means of y over categories of x.

---

[2] `encode x, generate(x_NUMERIC)` is often useful to create a *numeric* version of *string* variables.
[3] `alpha oldvar1 oldvar2` will calculate Cronbach's alpha from this scale.
[4] After running many multivariate models `estat summarize` will give you simple descriptive statistics for the specific sample used in that particular analysis.
[5] Other regression commands follow a very similar format: `command y x z` but are beyond the purview of this 2 page guide.
[6] `i.x` is Stata's notation for treating independent variables as *categorical* or *indicator* variables.
[7] For all graphs, options after a "," will be helpful in titling your graph e.g. `twoway lfit y x, title("...") xtitle("...") ytitle("...")`
[8] Graph schemes can change the overall look of a graph. `, scheme(s1color)` is often a good choice.
[9] `histogram x, percent` will scale the y-axis more intuitively in terms of percentages. `histogram x, discrete` gives a nicer display for categorical variables. The `percent` and `discrete` options can be combined.