

Two Page Stata

Andrew Grogan-Kaylor

2024-01-22

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.

have a question → get data → process and clean data →
analyze data → visualize data → 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*.

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.

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.

¹Comments, 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/>

`recode x (oldvalue = newvalue)`, `generate(xR)`² will recode a variable into a *new* variable. `recode x (-99/-1 = .)`, `generate(xR)` will recode negative numbers from -99 to -1 to missing for x. `recode x (7/9 = .)`, `generate(xR)` 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. Reverse coding often looks something like `recode x (1=3) (2=2) (3=1)`, `generate(xR)`.

Like many other statistical programs, Stata makes a large distinction between variables that are coded as *numeric*, and variables that are coded as *strings*. `describe x` will help you to ascertain the variable type. `encode x`, `generate(x_NUMERIC)` is often useful to create a *numeric* version of *string* variables. In the special case where the values of your variables are actually *numbers* but stored as *strings*, `destring x`, `generate(x_NUMERIC)` may be the more helpful and appropriate command. The differences between the two commands are subtle, so consult Stata help on each command.

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 Graphing⁷⁸

`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.

²While generating a new variable is optional, it is almost always a good idea.

³`alpha oldvar1 oldvar2` will calculate Cronbach's alpha from this scale.

⁴After running a multivariate model `estat summarize` will give you simple descriptive statistics for the specific sample used in that particular analysis.

⁵Other 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.

⁷For all graphs, options after a ";" will be helpful in titling your graph e.g. `twoway lfit y x, title("...") xtitle("...") ytitle("...")`

⁸Graph schemes can change the overall look of a graph. , `scheme(slc)` is often a good choice. In newer versions of Stata, , `scheme(stcolor)` works well.

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.