A Brief Stata Cheatsheet

Yuji Shimohira-Calvo

January 22, 2018

INTRODUCTION

This document is a concise cheatsheet of some basic Stata commands used throughout the course Doing Survey Research at the University of Edinburgh. This document will be updated as you learn more commands in class. Keep in mind that a good programmer is also a "lazy" programmer: if you can save 0.5 seconds typing something shorter, do it! That is why this document also tells you the shortest abbreviation for commands in Stata. The underlined bits of a command are the shortest ways to let Stata know which command you want to use (and same things applies to command options).

1 USEFUL COMMANDS FOR EXPLORING A DATASET

su**mmarize** variable(s) (1.1)

The *summarize* command calculates and displays a variety of univariate summary statistics. You can use it without specifying any variable to display the same univariate summary statistics for all variables in the dataset. If you specify one or more variable, *summarize* will only display the summary statistics of those variables.

describe variable(s) (1.2)

The *describe* command produces a summary of the dataset in memory. Like the *summarize* command, *describe* can be used with or without specifying variable names.

codebook variable(s) (1.3)

The *codebook* command examines the variable names, labels, and data to produce a codebook describing the dataset. This command can be used with or without specifying variables names. A useful option for *codebook* is *compact*, which displays a compact report on the variables:

codebook variable(s), compact (1.4)

2 GETTING SOME HELP

help command.name (2.1)

The *help* command displays help information about the specified command or topic. In fact, all the descriptions in this document are copied&pasted from Stata's help manual. The help command not only tells you how a command's syntax works, but also it gives you examples on how to use that command you need help with. It also lists all the available command options (like the *nolabel* option for the *tabulate* command).

3 USEFUL COMMANDS FOR UNIVARIATE AND BIVARIATE ANALYSIS

tabulate variable(s) (3.1)

The *tabulate* command produces a one-way (or two-way) table of frequency counts. Two is the maximum number of variables you can use with this command. A useful command option is *nolabel*, which displays numeric codes rather than value labels. For example:

tabulate variable(s), nolabel (3.2)

4 RELATIONAL AND LOGICAL OPERATORS

You can use logical operators with many commands. This is useful when you want to look at subsets of your dataset.

- == (equal to).
- != (not equal to).
- > (greater than).

- < (less than).
- >= (greater than or equal to).
- <= (less than or equal to).
- & (and).
- | (or).

Some examples using the above-mentioned operators are:

tabulate variable if variable==10 (4.1)

This tabulates only those cases where the variable is 10.

tabulate variable if variable>= & variable<=20 (4.2)

This one tabulates those cases between 10 **and** 20 including both 10 and 20.

tabulate variable if variable==10 | variable==20 (4.3)

This tabulates only the cases where the variable is 10 **or** 20 (notice the difference between **and** and **or**).

tabulate variable if variable!=10 (4.4)

This tabulates all cases **except** for those where the variable is 10.

5 USEFUL COMMANDS FOR DATA MANAGEMENT

drop variable(s) (5.1)

The *drop* command eliminates variables or observations from the data in memory. You can also use operators if you want to eliminate only some specific cases. For example:

drop if variable==10 (5.2)

This would eliminate those cases where the variable is 10. You can get âĂIJmore fancyâĂİ with it if you want to:

drop if variable==10 & variable2==20 (5.3)

This one eliminates those cases that satisfy the logical statement where variable is 10, **and** variable2 is 20. Only cases that are 10 in variable **and** 20 in variable2 are dropped.

generate newvariable=oldvariable (5.4)

The *generate* command creates a new variable. The example above duplicates a variable called "oldvariable" into a new variable named "newvariable." If you want to create a new "empty" variable you do:

generate newvariable=. (5.5)

Mind the period (punctuation mark) above. If you were to tabulate that new variable, Stata will report that "there are no observations."

rename oldname newname (5.6)

The rename command changes the name of an existing variable; the contents of the variable are unchanged. This command has some useful options:

rename variable, upper (5.7)

This option uppercase the target variable.

rename variable, lower (5.8)

This option lowercase the target variable.

rename variable, proper (5.9)

This option capitalises the first letter of the target variable.
With the *rename* command you can use a "wildcard" called "_all" (mind the underscore). For example:

rename _all, lower (5.10)

With the above you are lowercasing all variables in the dataset with one single line (no need to type all the names, a good programmer is also a "lazy" programmer). Speaking of "laziness",

I personally prefer lowercasing all my variables because that saves me the "effort" of clicking shift+letter when typing the names of variables. However, I know of some people who prefer uppercases because it is an "easier" way to distinguish variable names in lines of code. Your code, your call.

recode variable (rule) (rule2) (5.11)

The *recode* command changes the values of numeric variables according to the rules specified. Values that do not meet any of the conditions of the rules are left unchanged, unless an otherwise rule is specified. Consider the following example:

recode variable (0=1) (1=2) (5.12)

With the above you are changing the **old** numeric value 0 to a **new** numeric value 1, and the old numeric value 1 to a new numeric value 2. Now, say you have 100 different numeric values in that variable, but you only want to do "0 to 1", "1 to 2", "the rest of values to 3." You can easily do this in one line of code:

recode variable (0=1) (1=2) (else=3) (5.13)

A very useful option for this command is *generate*. Remember that *generate* is a command, but in this case it is acting as a command option. Using the example above:

recode variable (0=1) (1=2) (else=3), generate(newvariable) (5.14)

This is useful because you are recoding a variable (i.e., manipulating the dataset) into something more meaningful to you, but at the same time you are keeping the old (original) variable by creating a new variable that contains the transformed values.