School of Social and Community Medicine

An Introduction to Stata

Version: Jan 2017

Contents

Introduction to Stata

This session is an introduction to Stata. The basic use of Stata will be demonstrated today, then you will work through summarizing and manipulating a new dataset tomorrow.

Stata is a general purpose statistical package. It has the following advantages:

- It is very fast
- It has excellent data handling facilities
- It has good graphics facilities
- New commands can be written by users, so that Stata is constantly updated

Stata is best used by typing commands in the command window or in Stata's text editor (called a *do file*), where you can type several commands and execute them individually or in blocks. Data manipulation or analysis is carried out through these commands; if the commands are saved in a *do file*, then it is easy to redo an analysis.

In this introduction, commands are written in **`bold courier font`**, for example the general format of commands is:

`command varname(s) if … in … using … , options`

where:

- **`command`** is the name of the command to be used (e.g. **`summarize`**)
- ***`varname(s)`*** is a list of one of more variables (e.g. age)
- **`if …`** restricts the command to a selection of observations that fulfil a criterion (e.g. participants that are over 40 years of age)
- **`in …`** restricts the command to a selection of observations based on their order (e.g. the first 10 observations)
- **`using …`** chooses a data file to carry out the command on (used with a limited number of commands)
- **`, options`** any options a command has must be typed following a comma

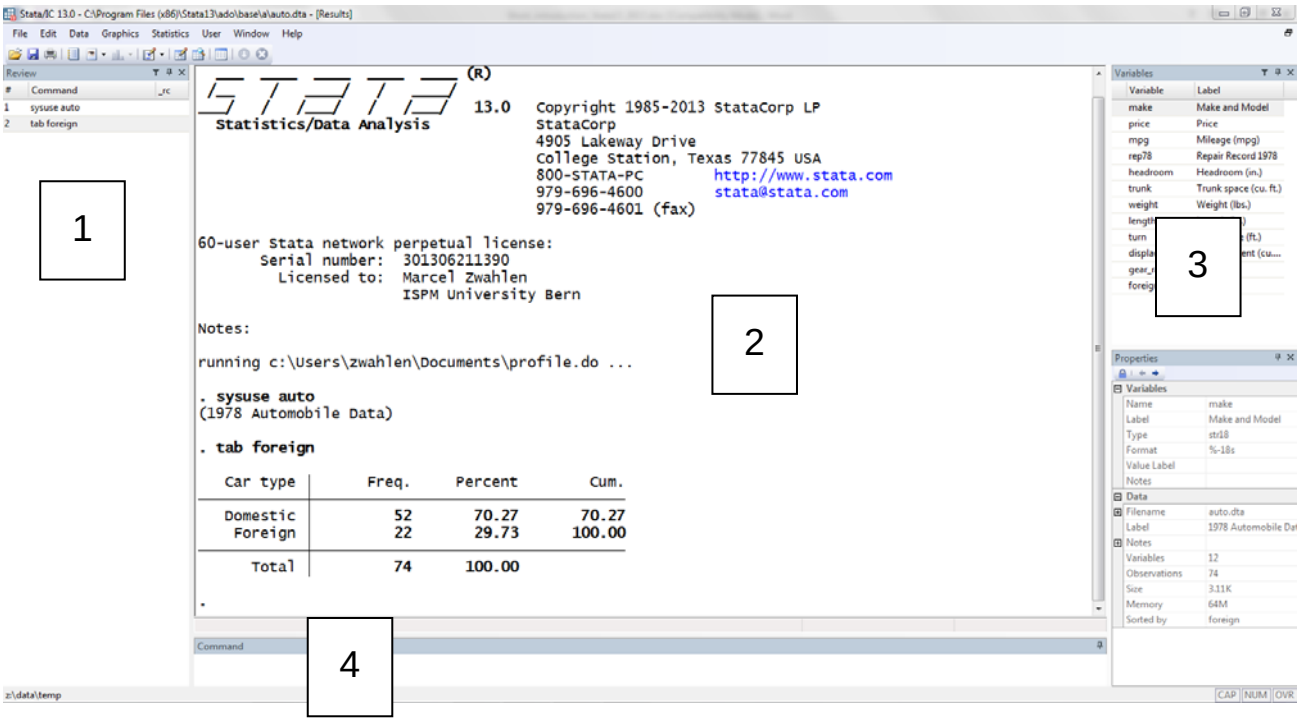
Many examples of useful commands will be presented in this introduction. During the exercise, please feel free to ask if you would like to know about any other commands.

If you like to see short videos describing how specific tasks can be done in Stata, visit http://www.youtube.com/user/statacorp.

What Stata looks like

Stata welcomes us with four opened windows and a blinking cursor in the Stata command window. Each window gives us different information:

1. Review: shows all previous commands used
2. Results: all output of any command appears here
3. Variables: here we have a list of all variables of a dataset in use as well as their labels
4. Stata Command: this is the place where you type your commands



Do files

Stata's text editor is opened by clicking this button: 

The text editor allows you to write out all the commands for working with Stata and save them as *do files*. This is useful for keeping track of the commands you are entering and repeating analyses.

To execute the commands in *do files*, highlight the line or block or lines that you want to run and press ctrl+d, or click this button: 

As we go through the introduction, feel free to either type commands into the command window (number 4 above), or type the commands into the *do file* and run them individually.

NOTE: each command needs to be on one separate line. If you want to extend the command across multiple lines, the line separator "///" needs to be used.

Section 1 – Getting Started

1.1 The Training Dataset

A dataset, **`Training.dta`**, was created for this introduction: the dataset contains simulated information for the following variables:

1. ID number
2. Age (years)
3. Gender
4. Height (cm)
5. Weight (kg)
6. Systolic blood pressure (mmHg)
7. Diastolic blood pressure (mmHg)
8. Smoking status

The dataset needs some editing before it can be analyzed. We will go through loading the dataset and the basic commands for manipulating data before taking a brief look at analysis. After this, another dataset has been prepared that you can work through by yourself.

You will find the required data files on the USB stick provided.

1.2 Loading and Saving Data

There are many ways to load a Stata data file; the simplest is to click the "open" icon  as for any other program – load the **`Training.dta`** dataset now. This loads the dataset into Stata memory, and displays the command Stata used to do this:

```
use "folder location\Training.dta", clear
```

Loading Stata files can be done either by using the "open" icon, or by typing **`use`** followed by the location of the dataset (the **`clear`** option is optional but necessary if data is already open in Stata).

The dataset should now be loaded into Stata. You will see all the variables on the upper right hand window, and the lower right hand window displays some information about the dataset, for example that it has 8 variables and 500 observations.

We are now going to save a copy of the dataset to work with. To save the dataset, you can either use the "save" icon  and choose a folder to save in, or use the command:

```
save "folder location\Training_initials.dta", replace
```

The option **`, replace`** allows Stata to overwrite any saved Stata files with the same name in the same folder. It is optional, but Stata will give an error message if any file exists with the same name. If you don't specify the folder location, Stata will save to the working directory folder unless you specify a different folder.

1.3 The Working Directory

The working directory of Stata is the folder in which Stata operates. Essentially, you select a folder on your computer and Stata looks in this folder to load and save datasets, which means you don't need to type in the folder location so many times when saving and loading datasets.

The necessary command for setting the working directory is:

`cd "folder location"`

The folder location can either be typed out, or (much easier) selecting the "copy address as text" option when right clicking on the address bar of the folder you want Stata to work in and pasting it into Stata. The command **`cd`** stands for "change directory".

1.4 Useful Information

- If you ever need more information about a command or are searching for the right command, you can type **`help commandname`** to bring up Stata's help feature. This is very useful, as not only can you find the right commands, you can see the command's syntax (how the command needs to be set out to work) and examples of how the command works.
- Stata syntax differentiates between bits of the command that are necessary and those that are optional. If any parts of the command are in square brackets, e.g. **`[, options`**], then they are optional.
- Commands can usually be shortened to save time, for example the **`tabulate`** command (used later) can be shortened to **`tab`**.
- The left window in Stata displays the commands that you have previously entered into the command window at the bottom of Stata. If you ever want to repeat commands, you can select them from this window or scroll through them using the page-up and page-down keys in the command window.
- Stata is case-sensitive – it distinguishes between upper and lower case letters. As such it is common practice to only use lower case in Stata to avoid issues with case-sensitivity. It also require American spellings e.g. summarize, color.
- Strings in Stata (i.e. letters, a word or list of words as opposed to numbers) almost always need to be inside quotation marks for Stata to understand properly. Hence, when we specify file names, we put them in quotation marks.
- Stata stores missing numerical values as "." – a period. The actual value of a missing value is a large number close to infinity. This is important to remember when analyzing data.
- Should you want to load an Excel spreadsheet into Stata, the command is: **`import excel filename`**

Section 2 – Describing Data

2.1 List

Now the data is in Stata, it is time to take a look at it using common commands.

One basic descriptive command is **list** which lists observations in the dataset:

```
list
```

Just typing **list** by itself will list all observations and variables in the dataset. Once the results window is full, Stata will wait for you to click –more- or click the spacebar in the command window. Alternatively, you can stop Stata from continuing to list data by clicking the red cross at the top  .

Using the **list** command, we see that all variables are numeric except gender, and that one variable, var1, isn't properly named.

You can restrict which variables are shown by typing them after **list**, for example if you only wanted to list ID numbers and age:

```
list id age
```

2.2 Summarize

Summarize gives a summary of one or more variables:

```
summarize
```

Typing **summarize** will display a summary of all variables, giving information on the number of observations, mean value, standard deviation, and minimum and maximum values. Adding a **detail** option gives you even more information, including the median and other percentiles, and adding variable names after **summarize** will restrict the summaries to those variables. For instance, if we wanted to know the median of age, we would write:

```
summarize age, detail
```

The summarize command can be shortened to **sum**, and is often used to examine the data before manipulation. If you want to summarize all variables at once just type **sum** without any variables. This allows us to check the means and range of values for all of our variables.

2.3 Tabulate

Then command **`tabulate`**, shortened to **`tab`**, is used to produce one- and two-way tables of categorical variables, depending on whether one or two variables are specified:

`tabulate` ***`variable_name(s)`***

If one variable is specified, then Stata displays the frequency and percentage of observations for each level of the category. For example:

```
tab smoker
```

shows us that 384 participants (76.8%) have a level of 0 (do not smoke), 97 (19.4%) have a value of 1 (do smoke) and 19 (3.8%) have a level of 9 (refused to answer/missing).

If two variables are specified, then Stata displays a two-by-two table of frequencies. For example:

```
tab smoker gender
```

shows us that in total, 199 females and 185 males do not smoke, 52 females and 45 males smoke, and 11 females and 8 males have missing data.

The options **`column`** and **`row`** can be specified to give the column percentage and row percentage respectively. For example, if we wanted to know the percentages of each gender that smoke and do not smoke:

```
tab smoker gender, col
```

As Stata is case sensitive, it has not put “Females” and “females” together into one group (and the same for males). We will put these together when manipulating the data to make it easier to analyze the data.

2.4 Restricting Commands

The in qualifier

Commands in Stata can be restricted to a subset of the data. The first method of restriction is to certain observations, such as the first or last 10 observations. To do this, type the command as usual, but add **`in` *`range`*** to the end of the command before any options. The ***`range`*** refers to a range of numbers, specified either as a single number or one number to another number (e.g. 1/10, where the / takes the place of " to "). For example:

```
list in 1/10
```

lists the first 10 observations for all variables.

A negative number means the number x^{th} from the end of the observations (so **`-10`** means the 10^{th} observation from the end of the dataset, in this case observation 491). The letters **`f`** and **`l`** can be used in place of stating the first and last observations, so **`f/10`** means the first 10 observations, and **`-10/l`** means the last 10 observations. Note that **`l`** is the letter "l", not 1.

```
list in -10/l
```

The if qualifier

It is also possible to restrict commands to observations that satisfy a given condition, such as to observations that are all female, all smoke, or smoke and are female. The qualifier here is **`command if exp`** , where **`exp`** is an expression and the observations are restricted to observations in which the expression is true. For instance:

```
summarize age if smoker == 1
```

summarizes age only for participants that smoke (as being a smoker is coded 1).

Note here the use of "==", this is a requirement in **`if`** statements if you are asking if one expression is equivalent to another expression. Other logical operators are given below.

Arithmetic	Logical	Relational
+ addition	~ not	> greater than
- subtraction	! not	< less than
* multiplication	| or	>= greater or equal to
/ division	^ and	<= less or equal to
^ power	== equal to	
	~= not equal to	
	!= not equal to	

Operators that you may not be familiar with are "or", which is denoted with "|", found next to the left shift key on a keyboard, (e.g. you wanted to restrict the command to females or smokers), and "not", which is denoted with "!=" (e.g. you wanted to restrict the command to participants who did not smoke).

Examples of other restrictions:

```
summarize age if gender == “Female” | gender == “female” | smoker==1
```

```
summarize age if smoker != 1
```

Missing values in Stata are coded as very large values close to infinity - this becomes an issue with some Stata commands, as if you restrict a command to observations that are greater than a given amount (e.g. **`summarize age if dia_bp > 80`**) then missing observations will also be included in the restriction. As such, it is often necessary to add another restriction if we wish to exclude missing values:

```
summarize age if dia_bp > 80 & dia_bp < .
```

The inclusion of “**`& dia_bp < .`**” means that missing values of diastolic blood pressure won’t also be included in the summary of age.

The by qualifier

It is also possible to have Stata perform a command on all the different levels of a categorical variable using the **`bysort`** command. For example, if you want to summarize height by smoking status:

```
bysort smoker: summarize age
```

The restriction **`bysort smoker:`** tells Stata that you want to summarize age individually for all levels of the categorical variable *smoker*.

Section 3 - Manipulating Data

3.1 – Generating Variables

New variables can be created easily in Stata:

generate *new_variable_name* = exp

Where **exp** is an expression, such as the addition of two variables. Remember the new variable name must start with a letter and not be named the same as any existing variables.

For example, we may want to create a new variable that is the average of the systolic and diastolic blood pressures:

```
generate average_bp = (var1+dia_bp)/2
```

The new variable **average_bp** will be missing for any observations that are missing in either **var1** (systolic blood pressure) or **dia_bp** (diastolic blood pressure). The **in** and **if** restrictions can be also be used with **generate**.

3.2 – Replacing Variables

Variables can also be replaced using a similar method:

replace *old_variable_name* = exp

Where **exp** is an expression, and the old variable must already exist.

For example, we have noticed that height is in cm and we would like it in metres:

```
replace height = height/100
```

Now we have height in metres, we could also generate a new variable for body mass index – weight in kg/height in metres squared:

```
generate bmi = weight/height^2
sum bmi
```

The summary of BMI tells us the average BMI is about 27 kg/m^2, which seems about right. The **in** and **if** restrictions can be used to restrict the replacement to a subset of observations.

In our dataset some missing values have been coded as 9. We want to replace the values of 9 in smoker with numeric missing values (.), so that Stata knows these values should be treated as missing values and not included in any analyses:

```
replace smoker = . if smoker == 9
```

3.3 – Renaming Variables

Renaming variables is straightforward:

rename *old_variable_name new_variable_name*

The only limitations to naming variables are that you cannot name a variable the same as an existing variable and that you have to start a variable name with a letter.

In our dataset, the systolic blood pressure is named var1, so we will rename it sys_bp:

```
rename var1 sys_bp
```

3.4 – Labelling Variables

Labelling variables is useful to remember what the variables mean. Currently, none of the variables are labelled, meaning if we come back to the dataset in a year we may forget what each variable actually is. Labelling variables is also straightforward:

label variable *variable_name “Label of your choice”*

We would like to relabel the sys_bp variable with something more informative than “var1”:

```
label variable sys_bp “Systolic BP (mmHg)”
```

Remember that strings must be enclosed in double quotation marks.

3.5 – Labelling Values

With categorical variables, we often have data that is numeric (e.g. 0, 1) where the numbers represent strings (e.g. “no”, “yes”). We can overlay the numeric coding with the strings that the numbers actually represent using a two-stage command. For instance, in the smoker variable, 0 means “no” and 1 means “yes”. For now we will ignore the value 9.

When labelling values, first we define a value label, then we apply it to the variable:

label define *label_name* #1 “String #1” #2 “String #2”
label values *variable_name label_name*

So for the variable smoker, we define a label that states 0 means “No” and 1 means “Yes”:

```
label define smoker_label 0 “No” 1 “Yes”
```

Then we apply this label to the variable smoker:

```
label values smoker smoker_label
```

Now when we see the different levels of smoker, we will see no and yes instead of 0 and 1.

```
tab smoker
```

3.6– Converting String to Numeric Data

Sometimes data will be saved as string variables when you need it to be numeric. Stata's command to recode the string variable to numeric is:

encode *variable_name*, generate(*new_variable_name*)

notice that the **generate(*new_variable_name*)** option is required (not in square brackets). This command generates ***new_variable_name*** taking each unique string value of ***variable_name***, coding it as a numeric value, and labelling the numeric values of the ***new_variable_name*** with the string value of the ***variable_name*** as the label.

Currently, gender is a string, and we would like it to be numeric:

```
encode gender, generate(gender2)
tab gender gender2
```

As you can see, gender and gender2 are identical. The difference is that gender2 is numeric with labels:

```
tab gender gender2, nolabel
```

The **nolabel** option here shows the tabulates the variable without the value labels, allowing us to see the numeric values of gender2.

We also need to make sure "Female" and "female" have the same numeric value. This can be done by replacing the values using the **replace** command:

```
replace gender2 = 1 if gender2 == 3
```

We also need to make sure "Male" and "male" have the same numeric value:

```
replace gender2 = 2 if gender2 == 4
tab gender gender2
```

Now gender2 is a binary variable, with 1 representing females and 2 representing males.

Note: we could have made all strings in gender lowercase, getting around having to replace any values:

```
replace gender = lower(gender)
```

We could then encode gender as before. There are generally many ways to accomplish the same goals in Stata.

3.7 – Removing (dropping) Data

Variables and observations can be removed from the dataset (dropped). This is a permanent removal, which is why it is important to save a master dataset at the beginning of any analysis so the original data could always be retrieved, and it is important to be sure about dropping any data.

Dropping variables is simple:

`drop` *`varlist`*

removes the variables in the variable list from the dataset.

A similar command allows you to keep only the variables that are specified:

`keep` *`varlist`*

removes the variables NOT in the variable list from the dataset.

Dropping observations works in a similar manner, except that you need to specify which variables to drop using **`in`** and **`if`** restrictions:

```
drop if age > 40
```

removes all observations where the age is more than 40 years.

```
drop in 1/100
```

removes the first 100 observations.

Keeping observations is equally similar:

```
keep if smoker == 1
```

removes all participants that were not smokers (as smoking is coded as 1 in this dataset).

If you would like to remove all data from memory use the command:

```
clear
```

which removes all data from Stata's memory, allowing you to start again with a different dataset.

Section 4 – Exercise Dataset

The “Exercise.dta” dataset has been created in a similar way to the “Training.dta” dataset, except different aspects need altering. Please follow the instructions below – if you get into any difficulties while in the session please ask and someone will assist you.

1. Load the “Exercise.dta” dataset

2. Summarize all the variables to familiarize yourself with the data

3. Check the values for weight – if any seem like they may represent missing values change these values to missing. Hint: missing values in Stata are denoted with a full stop, weight is in stone, and the maximum recorded weight is around 100 stone.

4. Rename var1 and var2 to more sensible names. Hint: check the variable labels.

5. Convert age into years, height into metres and weight into kg. Hint: 1 stone is 6.35kg, and the variable is decimal, so 10.5 stone is 10 and a half stones, not 10 stone 5 pounds.

6. Relabel these variables to reflect the new units

7. Generate a new variable for the body mass index (BMI = weight/height^2) and label it

8. Label the values of gender “male” and “female”. Hint: remember to define the label, then apply it to the variable.

9. Generate a new variable, smoker2, which converts the smoker variable from a string to numeric value. Make sure “No”/“no” and “Yes”/”yes” have the same value in smoker2.

10. (Optional) Perform a linear regression on systolic BP, looking at whether age, gender, smoking and BMI are associated. Hint: using the **`help`** command can help when finding the name of a command.