# **BQA WEEK 1**

## **EXERCISE 1:**

## **STATA FOR BQA**

**+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

DESCRIPTION: This exercise introduces additional commands for Stata. It will establish the commands you need to complete the following exercise 2 worksheet, which is linked to the conceptual issues covered in the mini lectures.

### FILES: BQA\_Lab.dta

### STEPS:

1. Create a second version of the **BQA\_Lab.dta** so you retain a copy of the original
2. Open your new copy of this dataset
3. Sort data
4. Add variable label and value labels to variable
5. Generate new variables from values of different variable

### NEW COMMANDS:

* Housekeeping: **set more off, capture**
* Inspecting the data: **display, sort, bysort**
* Modifying variables: **label variable, label define, label values**

### 1.1 Miscellaneous useful stuff

If you disable the “num lock” option, you can use the 9 (page up) and 3 (page down) keys to scroll through your previous commands while in interactive mode. Very useful!

See <http://www.stata.com/statalist/archive/2013-05/msg00330.html> for a range of useful keyboard shortcuts.

The results screen in Stata shows you one screen-full of output at a time, and shows you that it has more to come by displaying **– more –** at the bottom of the screen. See the rest by pressing the space bar or the enter key. In a do-file, including the command **set more off** allows the program to run without waiting for you to scroll through the output.

To interrupt Stata in the middle of a command (for example, if you have mistakenly asked Stata to execute a very time-consuming or endless routine), click on the white-on-red cross on the menu bar, or type “control-break”. It’s my belief (though possibly wrong) that you sometimes need to do this repeatedly to get the interruption to actually work.

Stata has a handy calculator function. Type “display” followed by the function you want to calculate (e.g. **display 4.37 \* 21 / 14.551)**

Stata also has a very powerful capacity to automatically use coefficients or other statistics from regressions in later calculations. But that’s beyond the scope of this primer.

You can cut and paste output from the Results screen to a Word or other document. The option “copy table” may be particularly useful.

### 1.2 Sorting data

The **sort** [varlist] command sorts your data by the variables in the variable list, starting with the first.

The **by** and **bysort** runs a command separately for each value of a variable. **by** requires that the data is sorted by the variable in question and cannot be abbreviated. **bysort** sorts the data for you and may be abbreviated **bys.**

|  |  |
| --- | --- |
| **bysort female: sum lrscale if lrscale < 77** | Specifying **bysort** means the command is repeated for each value of the variable **female** |

### 1.3 Labels

We have referred to labels in several places – this section gives a little more detail, and tells you how to make your own labels for variables and values. If you’re running short on time, this section is less important than some of the others – but if you are going to do some analysis using Stata, you should learn how to label variables and values.

***Variable* labels** are pretty easy: you can label (or re-label) a variable using the **label variable** command. There’s a variable in the data called **female**, which has no label. Give it a label of your own choice, using the following syntax:

* **lab var female “binary variable indicating female gender”**

***Value* labels** are a little trickier; the labelling is done in two steps. The variable “female” has no value labels (to see this, type **tab female**) – the following commands define and apply a set of labels:

|  |  |
| --- | --- |
| **label define female\_lab 0 “male” 1 “female”** | You first have to define the label. This command says what the label is called (female\_lab) and assigns labels to each value (we tell Stata that the value 0 is associated with males, and the value 1 is with females). Note the use of double quotes around each label. |
| **label values female female\_lab** | In this step, we apply the label to the variable. |

Now type **tab female** again, and you’ll see that the values are labelled.

In the example above, the variable to be labelled was called female, and the labels were called female\_lab. Value labels are often given the same name as the variable itself (this is the case in the ESS - see this by typing **describe**, which gives all variable names and all variable and value labels). So you could assign value labels by typing:

**label define female 0 “male” 1 “female”**

**label values female female**

To inspect an existing value label, type **label list** + the name(s) of the *labels* you want listed.

When generating new variables, you can copy the labels over from the original variable, by typing [:**lblname = exp**]. Type “**help generate**” for info.

**See section 1.3** in the ‘**Stata for BQA’ do-file** for more tips on recoding variables and adding value labels to variables.

### 1.4 Labels in Stata

In FiAS, you created a variable called **dom2**, which was a recode of the variable **domicil** (the area in which the respondent lives), that included the following categories:

1. A big city, including suburbs or outskirts of a big city
2. A town or a small city
3. A country village, or a farm or home in the countryside
4. Missing values

Here is the recode for this variable.

**recode domicil (1 2=1) (3=2) (4 5=3) (7/9 = .), generate(dom2)**

The variable currently has no variable or value labels. Give the variable a variable label and some useful value labels, writing your code in your code below and then copying it across to your do-file to check that it runs without error.

**/\* label the variable \*/**

**label variable dom2 "Domicile, respondent's description"**

**/\* define the label before assinging it to a variable\*/**

**label define dom2 1 "A big city, including suburbs or outskirts of a big city" 2 "A town or a small city" 3 "A country village, or a farm or home in the countryside"**

**/\* We can then assign that label to the variable\*/**

**label value dom2 dom2**

### 1.5 Logical operators as “true or false” statements

All the statements using relational operators that we have looked at so far appear as part of IF or IN statements.

However, statements using logical operators also have another function – they take the value 1 for observations where the statement is true, and 0 for observations where the statement is false.

|  |  |
| --- | --- |
| **(agea == 20)** | Takes the value 1 for the observations where age = 20, and 0 otherwise |
| **(cntry != “BE”)** | Takes the value 1 for all observations in countries other than Belgium, and 0 for observations in Belgium |
| **(agea >= 70)** | Takes the value 1 for all observations where age is greater than or equal to 70, and 0 otherwise. |
| **(age > 70 & female == 1)** | Takes the value 1 for all women over age 70, and 0 for everyone else |

This is very useful when generating new binary variables….

### 1.6 Generating new variables from values of different variables

Sometimes we may want to incorporate data from more than one variable into a new variable. For example, suppose we want to generate a single variable which indicates whether a respondent is a man aged 70 or over. Try the following syntax (as with many other things in Stata, there is more than one way to do this).

|  |  |
| --- | --- |
| **capture drop oldman** | In case the variable already exists |
| **gen oldman = .** | Start with a variable which is missing for all observations |
| **replace oldman = 0 if gndr == 2**  **replace oldman = 0 if agea < 70**  **replace oldman = 1 if gndr == 1 & agea >= 70 & agea != .** | Set the variable to zero for women  Set the variable to zero for individuals under age 70  Set the variable to 1 for men over 70 |
| **tab oldman, m**  **tab agea oldman, m**  **tab agea gndr, m** | Checking |

### 1.7 Combining variables to create a new variable (optional activity)

**NB: If you are short of time, I would save this for after the session and move on to Exercise 2.**

Generate a variable called **happyparent** which identifies people who:

* live in a household with children (variable **chldhm**)
* whose self-reported happiness is 9 or 10 (variable **happy**).

Remember to code missing values appropriately, and to check via the appropriate tabulations that your commands have done what you thought they would. Paste the code into a do-file and check that it runs without error.

There should be 4.875 “happy parents” in your data set and 402 individuals for whom the variable is missing.