Basic Data Analysis Course on Stata

Introduction to Stata

Md Monowar Hossain

Institute of Statistical Research and Training (ISRT)

December 21, 2024

Log file

- Log files allow you to record the inputs and outputs from Stata.Output sent to the results screen is not automatically saved. So you should always set up a log file to record your work.
- By default, Stata produces log files in its own format (smcl–Stata Markup and Control Language).

log files

- To start a log file, type:

```
log using example.txt, replace
```

- Here replace tells Stata to overwrite any existing version of example.txt.
- We can also type:

```
log using example.txt, append
```

- Here append adds new input/output to end of existing log file example.txt.

- A good approach to permanently storing log files with important results is to rename them after the analysis is complete - e.g. example class1.txt.
- Type log off

to pause recording.

- Type log on

to resume recording.

- Type log close

to end the log file.

Entering data into Stata

- You can enter data into Stata using **Graphical User Interface** (GUI) by clicking the Edit button or by typing edit in the command window.
- Also, we can use a do file to enter data by writing the following syntaxes:

```
input hhid sex str10 location
10030 1 urban
10031 0 rural
10032 1 rural
end
```

where hhid and sex are numeric variables and location is string variable.

Rules for variable naming in Stata

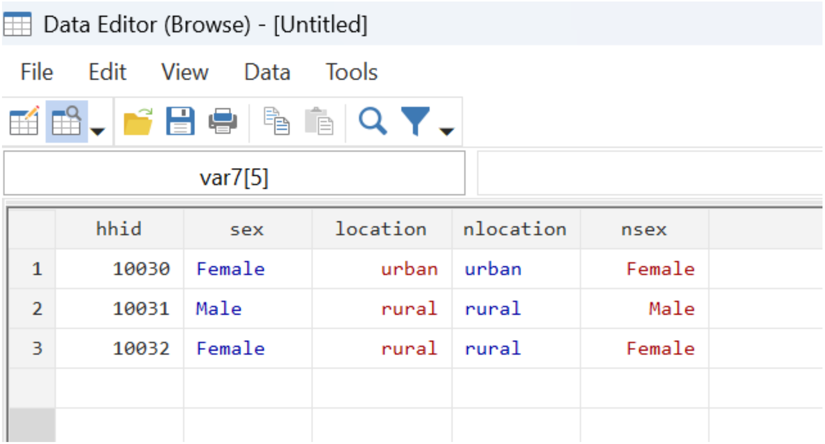
- Stata is case-sensitive, so using all lower case letters in variable names is a good idea.
- They can contain no more than 32 characters.
- They can contain letters, numbers, or underscores (_).
- Spaces or other special characters (like &, *, %, etc.) are not allowed.
- The first character must be a letter or underscore, not a number. Starting variable names with underscores is a really bad idea, since Stata's built-in variables begin with an underscore.

Value labels and the color coded system

In the example used above we have 1 and 0 to represent sex. Say, 0=Male and 1=Female. If we want to label the values by ‘Male’ and ‘Female’, then first we have to define a value label. This label will be created as an object in Stata. Then we will need to attach that value label to a variable. For example,

```
label define gender 0 "Male" 1 "Female"
label values sex gender
```

Here we create a value label gender as an object in the first step. Then we attach it to the variable sex. There is a color coded system in Stata. All the numeric variables appear in black. The string variables appear in red. The labelled variables appear in blue.



	hhid	sex	location	nlocation	nsex
1	10030	Female	urban	urban	Female
2	10031	Male	rural	rural	Male
3	10032	Female	rural	rural	Female

Labelling variables and saving data

Labeling a variable means giving the variable names a definition. For example, we mean ‘household identification number’ by hhid and ‘type of area of residence’ by location. It is always advisable to label the variables for future use. We can do that as follows:

```
label variable hhid "Household identification number"
label variable location "Type of area of residence"
label variable sex "Gender of the respondent"
```

After we are done with the initial formatting, we can save the data we have just entered and labeled. For example,

```
  save "E:\StataClass1\FirstData.dta"
```

Converting data into different types

- String to labeled numeric:

```
encode location, gen(nlocation)
labelbook nlocation
```

- Labeled numeric to string:

```
 decode sex, gen(nsex)
```

Renaming a variable

- rename

Sometimes you have to rename the variables so that the name can give you an idea of what the variable is about. For example,

```
use bd_individual.dta, clear
rename v011 DOB
rename v012 current_age
```

- labelbook

This command provides with the details about a value label, such as ranges of values and some other basic information about labels. For example,

```
labelbook V024
```

- Note that V024 is a value label, not the variable v024.

browse or edit

The command browse brings up a spreadsheet-style data editor in which you can examine the raw data. But often we may only want to view a few variables at a time, especially in large datasets with a large number of variables. In such cases, simply list the variables you want to examine after browse. For example,

```
 browse caseid v002 v003 v012 v024
```

- We can also browse data by putting a condition. For example,

```
browse if v012<21
browse if v024==1
browse caseid v002 v003 v012 v024 if v012<21
```

- We can use edit to correct if any anomaly is found during the examination of data.

list

The browse and edit commands start a pop-up window in which you can examine the raw data. You can also examine it within the results window using the list command-although listing the entire dataset is only feasible if it is small. If the dataset is large, you can use some options to make the output of list more tractable. For example,

```
list
list caseid v002 v003 v012 v024
list if v024==1
list caseid v002 v003 v012 v024 if v024==1
```

assert

With large datasets, it is often impossible to check every single observation using list or browse. An useful command is assert which verifies whether a certain statement is true or false. For example, you might want to check whether ages of the respondents in our dataset is between 15 and 49 as they should be.

```
assert v012>=15 & v012<=49
(blank output means no contradiction found)
assert v024>15 & v024<=49
17863 contradictions in 17863 observations
assertion is false
```

- If the statement is true, assert does not yield any output on the screen. If it is false, assert gives an error message and the number of contradictions.

describe

This command reports some basic information about the dataset and its variables (e.g. size, number of variables and observations, storage types of variables, labels etc.). You can also mention some specific variables to see information on only those. For example,

```
describe
describe caseid v002 v003 v012 v024
```

summarize

This command provides summary statistics, such as means, standard deviations, and so on. For example,

```
summarize v012
summarize v012, detail /*(to get percentiles)*/
summarize v012 if v024==1
```

- codebook

This command provides various information on variables such as range of values, numbers of observations, missing values and unique cases, summary statistics of numerics etc.

```
codebook
codebook caseid v002 v003 v012 v024
```

- tabulate

This command provides one-way or two-way frequency distribution for categorical variables. For example,

```
tabulate v024
tabulate v024 v106
```

THANK YOU