

Training on

KoBoToolbox for Data Collection

&

Introduction to Data Analysis Using Stata

Part 2: STATA

May 25, 2022

Introduction to data analysis using STATA

STATA is a powerful command driven statistical package with data management, statistical analysis and graphics capabilities.

It is a complete, integrated software package that provides all your data science needs—data manipulation, visualization, statistics, and automated reporting.

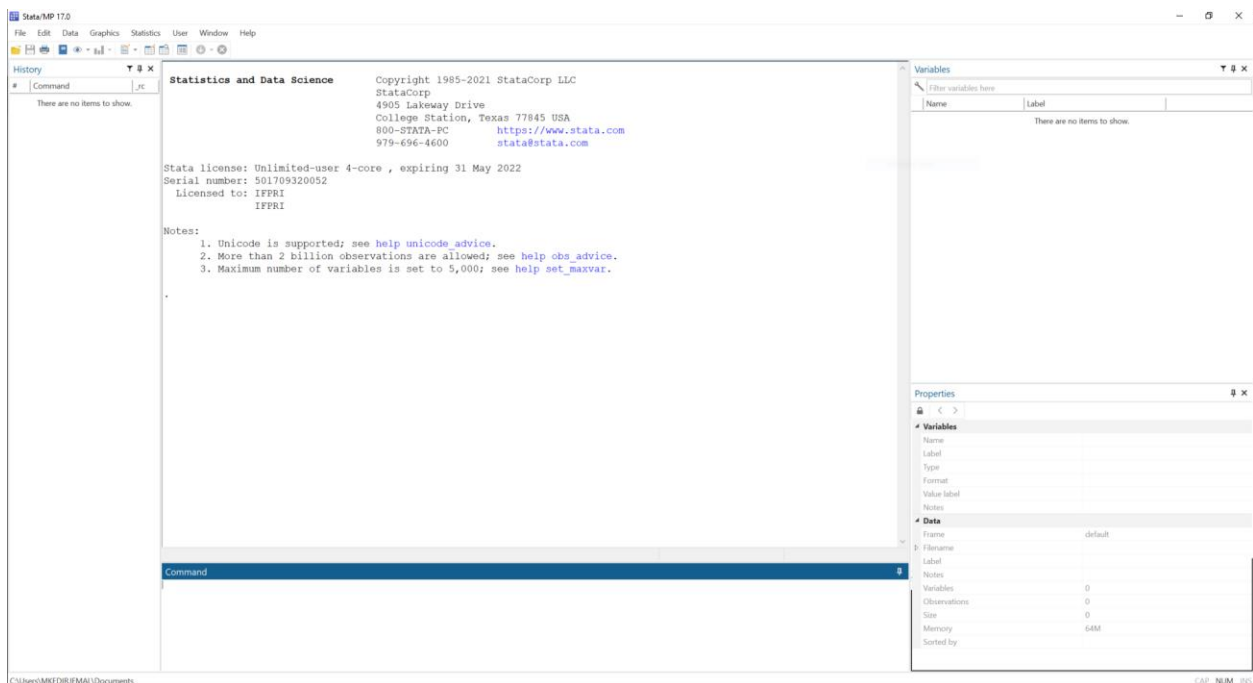
It is a fast, accurate, and easy to use software. STATA's commands for performing tasks are intuitive and easy to learn.

The training will introduce you to some of the basic tasks you can do in Stata, including opening a dataset, investigating its contents, calculating some descriptive statistics, and creating some graphs. Many of these topics will only be touched upon briefly. It should give you an idea of what Stata can do and how it works. In order to become familiar with Stata's commands, menus, and scripting, we will walk through briefly.

The Stata Interface

Stata has a graphical user interface (GUI) for command entry via menus and dialogs. Stata may also be used in a Command window, type the commands that follow a dot (.) in the boxed listings below into the small window labeled Command.

Stata (version 17): screen appearance:





Results window: Stata shows the results in the larger window

History: keep track of command operations used

Variables: located on the top right that lists the variables in the dataset being used

Properties: displays properties of the variables and datasets

Icons in menu options: Data Editor (edit and browse) , Do file Editor 

Stata has a point-on-click menu from which different commands can be selected. Since Stata is a command-driven package, the easiest way to learn it is by typing in the commands.

There are three ways to enter commands:

- Menu driven graphic user interface
- In the command window, type the command and hit enter to execute it
- Do-file: write commands/script in a “do-file” and execute the do-file.

It is a good practice to use do files when performing long or repetitive tasks. Since do files are stored as permanent records, they are editable in the future. Manipulation driven by menus or commands is useful for testing commands/menus on the fly, and then using the command in a do file later.

Exercise

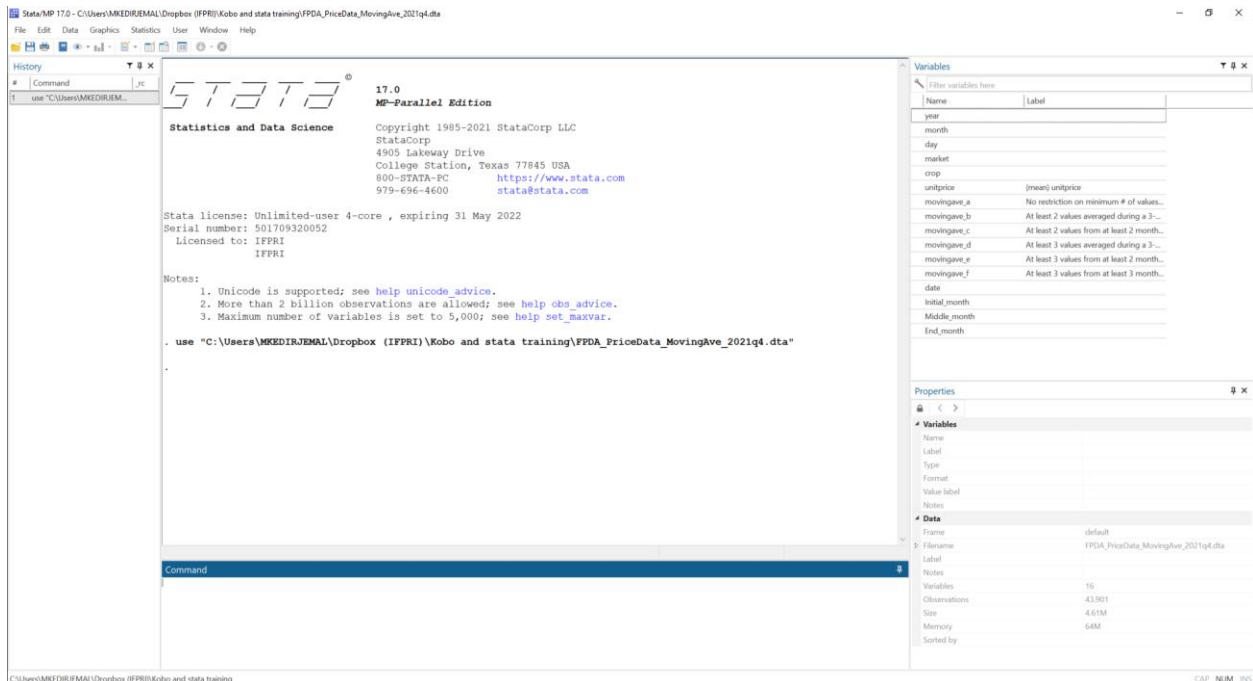
This training will use sample data from a price survey collected in five PNG markets in 2021. Use commands that follow a dot (.) in the boxed listings below into the small window labeled Command. A "Syntax note" will be displayed when something notable happens in a command's structure.

Open Stata and familiarize yourself with its main windows (results, command, review and variable windows).

Start by loading the 'fpda_price_sample_2021.dta' dataset, located in the training folder. Note that Stata datasets always have '.dta' extension.


To access existing Stata dataset using the menus,

1. Select File > your working directory...
2. Click on 'fpda_price_sample_2021' in the directory folder.



Data overview

In the Data Editor (Browse), we can see a quick overview of the data.

By clicking the Data Editor (Browse)  button, or by selecting Data > Data Editor > Data Editor (Browse) from the menu, or by typing 'browse' in the command window.

The Data Editor opens up and you can see that Stata regards the data as one rectangular table. Each column represents a variable, and each row represents an observation. The variables are described by descriptive names, while the observations are numbered.

When you select Data > Data Editor > Data Editor (Editor), you are presented with the same view, but you can now edit the data, much like a spreadsheet.

You can also paste in data this way, if you prefer. However, owing to repeatability difficulties, this is not advised.

The screenshot shows the Stata Data Editor (Browse) window for a dataset named 'FPDA_PriceData_MovingAve_2021q4.dta'. The main window displays a table with columns: year, month, day, date, market, crop, unitprice, movingave_a, movingave_b, movingave_c, movingave_d, and movingave_f. The data spans from 2021 to 2009. The right-hand side of the window shows the 'Variables' list and the 'Properties' panel for the selected variable 'movingave_f'. The Properties panel shows: Name: movingave_f, Label: At least 3 values from at least 3 mo, Type: float, Format: %9.0g, Value label: %9.0g, Notes: (empty), Frame: default, Filename: FPDA_PriceData_MovingAve_2021q4.dta, Label: (empty), Notes: (empty), Variables: 16, Observations: 43,901, Size: 4.61M, Memory: 64M, Sorted by: (empty).

Using syntax in the command window: The biggest challenge is learning how to communicate with Stata. The code for programs and commands is often composed of "command name," "variable name(s)," and "options". Help files contain details on the syntax of each command.

For example type 'browse' in the command window and hit enter to open Data Editor (Browse) window

Data Manipulation

You can use Stata as a calculator by using the display command. Let's use the "display" command, which can be shortened to "di," to show examples of these: di 10+15 and di 8/3 +16. For strings use quotes, for example di "hello stata". Stata commands are case-sensitive. All Stata command names must be in lower case. For example, display is not the same as Display, so the latter will not work.

Below are the most important arithmetic, logical and relational operators in Stata. Among the most common are & (and), ! (or), and ! (not).

Logical Operators in Stata

And

&

Or	
Not	! or ~
Multiplication	*
Division \	\
Addition	+
Subtraction	-
Less Than	<
Greater Than	>
Less Than or Equal	<=
More Than or Equal	>=
To The Power Of	^
Wildcard	*

Describing Data

Browse: there are several ways in Stata to investigate and describe data. You have begun browsing the data in the previous section.

//type browse in the command window

`browse`

Most of the time, we only want to view a few variables at a time, especially in large datasets with many variables. To examine such variables, simply write them after browse:

```
browse market crop year
```

List: we have seen that the browse and edit commands open a pop-up window in which you can examine the raw data. If the dataset is not too large, you can also examine it in the results

window by using the *list* command. In large datasets, you can use some options to make the output more tractable. As an alternative, we can list all variables but only a few observations. For example the observation 1 to 50:

```
//type list market in the command window
```

```
list market
```

```
list in 1/100 //lists observation from 1 to 100
```

```
list in 10/29 //lists observation from 10 to 29
```

```
list in 100/1 //lists observation from 100 to last observation (lower case 'l')
```

```
list in -10/1 //lists observation from last 10 observation (lower case 'l')
```

Sub-setting the data can also be achieved using an "if" qualifier. There are two possible qualifiers in the qualifier: either "true" or "false" (that is, 1 or 0). For example, list only price data collected in 2022.

```
list if year == 2022
```

If you have a large dataset, you may not be able to check every observation using list or browse. Several additional commands are available in Stata for examining data.

Assert: One useful command is *assert*, which verifies whether a certain statement is true or false. Check that all unitprice values are positive, for example:

```
assert unitprice > 0
```

```
assert unitprice < 0
```

When the statement is true, assert does not produce any output on the screen. The assert command gives an error message and the number of contradictions if it is false.

Describe: this command provides a brief overview of the dataset and its variables (size, number of variables, observations, storage types of variables, etc.).

```
describe
```

Codebook: It provides additional information about the variables, such as summary statistics for numeric, examples of data points for strings, etc. Without a list of variables, codebook will give information on all variables in the dataset.

```
//the command displays the frequency of each market
```

```
codebook market
```

Summarize: using the *summarize* command, you can see a broader picture of all the variables in the dataset like summary statistics, such as means, standard deviations, and so on. You can also specify a variable (e.g. *sum unitprice*) after the command to summarize only specific variables. If you would like more precise information (e.g. percentiles) then you can add the detail option to the end of that command, i.e. *sum unitprice, detail*.

```
//in the command window type sum
```

```
sum
```

```
sum unitprice
```

```
sum unitprice, detail
```

Tabulate: You can use this command, for example, to create a frequency table or a cross-tabulation of two variables. "Tab" shows you how many times each answer has been given to a particular variable. This method is suitable for variables with relatively few entries, such as categorical data. You will receive an error message if you attempt to use "tab" on a variable with hundreds of entries. Typing 'tab market' will show how many markets are there in the data. It is also easy to obtain crosstabs which give a breakdown of a variable by another. The 'tab market crop', for example, will display the crop and number of observations for each market.

```
tab market
```

```
tab crop
```

```
tab market crop
```

With the tabulate command and the *sum(varname)* option, it is possible to get a quick idea of the descriptive statistics of some subgroups. For example, the average sweet potato unitprice per market:

```
tab market if crop=="Sweet Potato",sum(unitprice)
```

Another tabulate command is "tabstat". This is used for continuous variables, and it is mainly used to determine mean values. For example *tabstat unitprice* will give the average unit price in the dataset. The 'if' command can be used to subset the data to get the average price of each crop by a market.

```
tabstat unitprice, by(crop)
```

```
tabstat unitprice if market=="Lae", by(crop)
```

Using the command options we can also access other useful statistics such as the median *tabstat unitprice, by (crop) stats(med)*, or the variance *tabstat unitprice, by (crop) stats(var)*. For a full list of the available statistics, type *help tabstat*.


```
tabstat unitprice if market=="Goroka", by(crop) stats(med)
```

```
tabstat unitprice if market=="Lae", by(crop) stats(var)
```

Inspect: this command is very useful. It summarizes missing values (if any) and plots their distribution.

```
//in the command window type inspect
```

```
inspect
```

Bysort: Sometimes you may want to break down summaries based on a specific variable. For example, you might like to see how unitprice variable changes over time. This is where *bysort* comes into play. Suppose you want to see how the mean unitprice varies by market for each crop. The command here would be: *bysort market crop: summ unitprice*.

```
//type bysort
```

```
bysort market crop: sum unitprice
```


Graph: Stata has very comprehensive graphics capabilities (type "help graph" for more details). You can graph a simple histogram with the command:

```
graph twoway histogram unitprice
```

```
graph twoway scatter unitprice year if crop=="Sweet Potato" & market=="Lae"
```

Do and log files

With Stata, you can keep track of what you've done with datasets. Rather than typing commands one by one interactively, you can enter them all at one time in a do-file, then run the do-file once. Upon completion of the do-file, the results of each command can be reviewed in a

log file. Stata has a built-in editor - simply click the pad-and-pencil icon  along the top of the screen. Text editors such as Word and Notepad can be used to create Do-files. Most do-files follow the same format:

```
cd "[path]/[working directory]/"
capture log close
log using "[path]/[working directory]/class.log", replace text
```

```
// Stata basics training:- do-file template
// File name: Sample_do_template1.do-file
// Task: to display basic do-file structure
```

```
clear all
set more off
set memory 100m
```

```
use "[pathway]/fpda_price_sample_2021.dta", clear
* Crop price data collected by FPDA in 7 markets in 2021
```

```
**LIST OF COMMANDS
```

```
Save "[pathway]/[filename]", replace
```

```
log close
```

Here are the different commands:

Clear - clears any data currently in Stata's memory, closes open files, windows, and dialog boxes while clearing data, labels, and stored results. Then you can run a new do-file and clear all previous work.

cd "[path]/[working directory]/" sets your working directory, the location from which you will retrieve and save your data/files. So, if you type 'use fpda_price_sample_2021.dta, Stata will look it up in this folder. In order to access a different directory during the session, just enter the full path to its location.

In case of a space in the file or directory name, be sure to include the file path between inverted commas.

capture log close – closes an already-opened log. The code log using function will not work if an open log already exists. Using *capture* prevents an error message from appearing and allows the do-file to continue if a log is already closed. Without using *capture*, the *log close* command will fail if a log has not been opened.

log using sample1.log, replace text - creates a log file of all the results. Re-running an updated do-file will overwrite the old log file with the updated results if you choose the *replace* option. Use the *append* option instead if you want the new file to be added at the end of previous versions.

set more off - Stata will not pause and display the ---more--- message in the results window for you to review each page on-screen and press a key to get more. Stata will instead run the entire do-file without pausing.

set memory 100m - specify 100m as Stata's default memory might not be enough for large datafiles. When you attempt to open a large file, you receive the following error message: no room to add more observations. If you need additional memory, you can increase it. Use the *describe* command to find out the size of the file before the data opens into Stata.

Use "[pathway]/[filename]" - to open or retrieve data from the current directory

Save "[pathway]/[filename]" - saves data to working directory

log close – closes the log file.

Comments/Notes

In your do-file, keep detailed notes so that you can see what you were trying to accomplish with each command or set of commands. You can add comments in several ways:

// - Stata ignores two consecutive slashes (or a slash followed by an asterisk *), so you can type the title or brief description of each section. But unlike two consecutive slashes, which can appear anywhere in a line, an asterisk can only appear at the beginning.

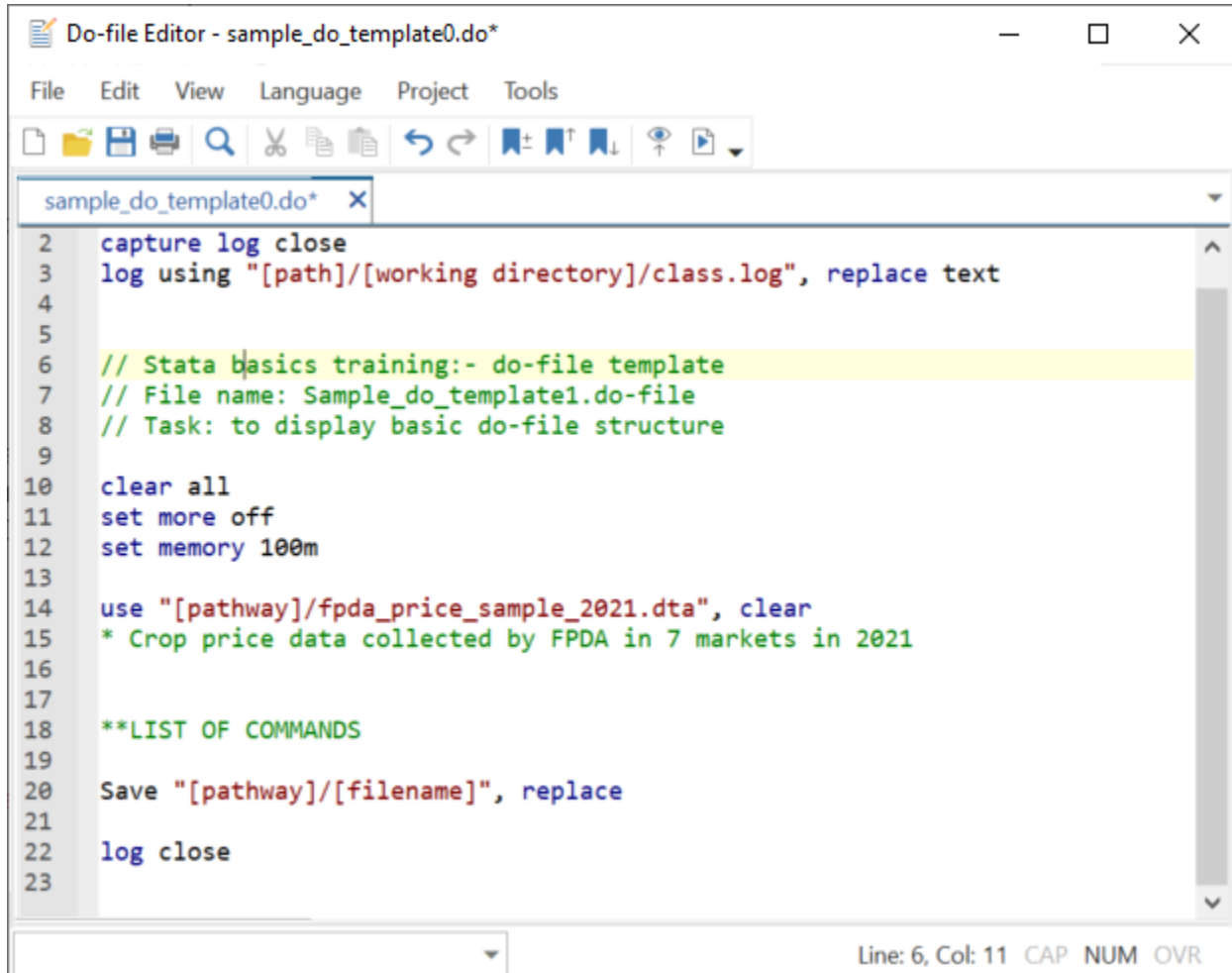
In addition to comments, you can use slashes or asterisks to temporarily disregard commands - if you wish to re-insert the command later, simply delete the slashes or asterisk.

/* */ - a note can be inserted inside these pseudo-parentheses after a command. Using this is also useful for temporarily blocking a set of commands - place /* at the beginning of the first and */ at the end, and Stata will simply ignore them all.

*** - describe a task or line of code.

* - notes or explanations of what a line of code does.

NOTE: If you use three consecutive slashes, the command will ignore the rest of the line and move on to the next.



```
Do-file Editor - sample_do_template0.do*
File Edit View Language Project Tools
sample_do_template0.do* x
2 capture log close
3 log using "[path]/[working directory]/class.log", replace text
4
5
6 // Stata basics training:- do-file template
7 // File name: Sample_do_template1.do-file
8 // Task: to display basic do-file structure
9
10 clear all
11 set more off
12 set memory 100m
13
14 use "[pathway]/fpda_price_sample_2021.dta", clear
15 * Crop price data collected by FPDA in 7 markets in 2021
16
17
18 **LIST OF COMMANDS
19
20 Save "[pathway]/[filename]", replace
21
22 log close
23
Line: 6, Col: 11 CAP NUM OVR
```

Labels:

You can put labels on datasets, variables or values – this helps to make it clear exactly what the dataset contains. A dataset label of up to 80 characters can be used to tell you the data source, it's coverage, and so on. This label will then appear when you describe the dataset. For example, try the following:

Labels are designed to help the user of a dataset understand and present their findings. These are often essential, for example if you have a categorical variable gender with two values 1 and 2, and no label you are in trouble as you will not know which refers to male and which to female.

Generally these will be provided in your data, but not necessarily. Often the variable name itself will be self explanatory, for example in the fpda_price_sample_2021 dataset the meaning of the

variable “market” is obvious. If it is not so obvious, a look at descriptive or the data browser makes it clear that it refers to PNG markets. It is easy to rename a variable.

```
//type rename into the command window.
```

```
rename market marekt_name
```

As well as the actual names, you can also label a variable. These can be used to provide additional information that is not apparent from the variable name.

So far none of the variables in the dataset have these. Adding a variable label to a variable is also straightforward.

```
//type label variable unitprice "".
```

```
Label variable unitprice "Unit price in kina per kg''.
```

From now on, practice all of the preceding commands in the do file saved in the training folder.

Source and references/credits

<https://www.stata.com/why-use-stata/>