Introduction to Geographic Information Science

Week 02 Stata Commands

Christopher G. Prener, Ph.D.

Spring, 2016

Contents

1 Comments

1.1 Syntax

```
// [comment text]
/* [comment text] */
```

1.2 Basic Examples

```
// This is a single line comment

/* This is a
multi-line comment */
```

Note that multi-line comments may include one or more carriage returns.

1.3 Divider Examples

```
// ================================================================================
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

1.4 Notes

Comments can only be used in do-file text. They cannot be entered interactively into the command box in the main window. Use comments to annotate your work, document the purpose of a do-file, and track changes that you make to your code. Dividers can be entered into your do-files to separate groups of commands. These increase the readability and organization of your do-files.

Additional details can be found in the Stata documentation for do-files on page 3 (link).

2 Working Directory

2.1 Syntax

```
pwd
cd ["]folder_path["]
cd ["]folder_name["]
```

2.2 Windows OS Examples

The pwd command ('print working directory') is used to display the current working directory:

```
. pwd
E:\Users\username\Desktop
```

The cd command ('change directory') is used change to a new working directory:

```
. cd E:\Users\username\Documents
E:\Users\username\Documents
```

If your working directory (in this case the 'Documents' directory) contained a sub-directory named 'Working', you would use the following syntax:

```
. pwd
E:\Users\username\Documents
. cd Working
E:\Users\username\Documents\Working
```

If you wanted to navigate from a sub-directory to a parent directory, you would use the following syntax:

```
. pwd
E:\Users\username\Documents\Working
. cd ..
E:\Users\username\Documents
```

Note that the pwd command is not required in either of the last two examples. It is only included for illustrative purposes.

2.3 Mac OS Examples

The pwd command ('print working directory') is used to display the current working directory:

```
. pwd
/Users/username/Desktop
```

The cd command ('change directory') is used change to a new working directory:

```
. cd /Users/username/Documents
/Users/username/Documents
```

If your working directory (in this case the 'Documents' directory) contained a sub-directory named 'Working', you would use the following syntax:

```
. pwd
/Users/username/Documents
. cd Working
/Users/username/Documents/Working
```

If you wanted to navigate from a sub-directory to a parent directory, you would use the following syntax:

```
. pwd
/Users/username/Documents/Working
. cd ..
/Users/username/Documents
```

Note that the pwd command is not required in either of the last two examples. It is only included for illustrative purposes.

2.4 File Paths or Names With Spaces

File paths or names that contain one or more spaces require double quotes ("") around the path:

```
. cd "E:\Users\username\Three Word Folder"
E:\Users\username\Three Word Folder
```

This requirement holds true for any commands that utilize file or folder paths or names.

2.5 Notes

Additional details can be found in the Stata documentation for the cd command (link).

3 Opening Stata Datasets

3.1 Syntax

```
use ["]file_name["]
```

3.2 Example

To open a file in your working directory:

```
. use data.dta
```

3.3 Notes

See Section 2.4 for notes on files or directory paths with spaces in them - these apply to the `use` command. The file name can also be substituted for a full or relative file path if necessary. Examples of this as well as additional details can be found in the Stata documentation for the `use` command (link).

4 Saving Stata Datasets

4.1 Syntax

```
save ["]file_name["] [, options]
```

4.2 Options

The `replace` option allows you to overwrite an existing file.

4.3 Example

To save a file under a new filename (the equivalent of 'Save As'):

```
. save newFileName.dta
```

To save a file under its current file name, thereby overwriting the current data (the equivalent of 'Save'):

```
. save currentFileName.dta, replace
```

4.4 Notes

See Section 2.4 for notes on files or directory paths with spaces in them - these apply to the `save` command. The file name can also be substituted for a full or relative file path if necessary. Examples of this as well as additional details can be found in the Stata documentation for the `save` command (link).

5 Importing CSV Files

5.1 Syntax

```
import delimited ["]file_name["] [, options]
```

5.2 Options

The most important option for basic uses is the `varnames(#)` option. This specifies the row that contains variable names.

5.3 Example

To import a CSV file in your working directory that has no variable names included:

```
.  import delimited data.csv
(2 vars, 5 obs)
```

To import a CSV file in your working directory that has variable names saved in the first row:

```
.  import delimited data.csv, varnames(1)
(2 vars, 5 obs)
```

It is important to review your data before you import it to understand whether or not you need to specify variable names when you import the data. Data saved as CSV files can be opened using Microsoft Excel or another spreadsheet application.

If you import data and notice that the variable names Stata creates appear to actually be data, that is a tip-off that there may not be variable names included in the spreadsheet and you should not use the `varnames(1)` option. Likewise, if you import the data and notice that variables have been named `v1`, `v2`, etc., you will likely find that the variable names have been saved in the first observation. You should therefore use the `varnames(1)` option.

5.4 Notes

See Section 2.4 for notes on files or directory paths with spaces in them - these apply to the `import delimited` command. The file name can also be substituted for a full or relative file path if necessary. Examples of this as well as additional details can be found in the Stata documentation for the `import delimited` command (link).

6 Importing Excel Files

6.1 Syntax

```
import excel ["]file_name["] [, options]
```

6.2 Options

There are two common options for importing Excel files. The first is the `sheet("sheetname") option`. By default, new workbooks in Excel have one sheet named "Sheet1". I recommend starting with this as your default import option: `sheet("Sheet1")`. The second option is the `firstrow` option, which instructs Stata to treat the first row of data in the spreadsheet as the variable names.

6.3 Example

To import an Excel file in your working directory that has data saved in 'Sheet1' and variable names saved in the first row:

```
. import delimited data.xls, sheet("Sheet1") varnames(1)
```

As with CSV files, it is important to review your data before you import it to understand whether or not you need to specify variable names when you import the data.

6.4 Notes

This command will work for both Excel 1997/2003 (.xls) files as well as Excel 2007 and later (.xlsx) files. When you use the `import excel` command, specify the file type (.xls or .xlsx when you declare the file name). Stata imposes a size limit of 40 MB for Excel 2007 and later (.xlsx) files.

See Section 2.4 for notes on files or directory paths with spaces in them - these apply to the `import excel` command. The file name can also be substituted for a full or relative file path if necessary. Examples of this as well as additional details can be found in the Stata documentation for the `import excel` command (link).

7 Clearing Stata's Memory

7.1 Syntax

```
clear
```

7.2 Example

Entered by itself with no options, the `clear` command will erase all data from Stata's memory. This is a necessary command to run before importing new data if you are not saving the data currently in Stata's memory.

7.3 Notes

If you have not saved your work and you run the `clear` command, you will lose that work permanently. Stata will not warn you before executing this command.

Additional details can be found in the Stata documentation for the `clear` command (link).

8 Common Errors

8.1 Error 4: Data in Memory Would Be Lost

When opening data using the use command, or when importing data using the import delimited or import excel commands, you may received this error:

```
.  import delimited E:\Users\username\Documents\data.csv
no; data in memory would be lost
r(4);
```

This error indicates that you have unsaved data in Stata's memory. Save your data or use the clear command.

8.2 Error 198: Invalid Syntax

When referencing a file using the use, save, import delimited, or import excel commands, you may receive this error:

```
.  cd E:\Users\username\Documents\Week 2
invalid syntax
r(198);
```

This error indicates that there is likely a space in the given file or directory path. Enclosing the full path in double quotes ("") should rectify this error.

8.3 Error 198: Option Not Allowed

If an option is misspelled or is not actually a valid option for a command, you will see the following error. In this case, the option for the summarize command has been misspelled - it is `detail`, not `detailed`:

```
.  summarize pop, detailed
option detailed not allowed
r(198);
```

8.4 Error 199: Command Not Recognized

If a command is misspelled, it will generate the following error. In this case, the command describe has been misspelled:

```
.  dscribe type
command dscribe is unrecognized
r(199);
```

If the command was written as part of a user-installed package, this error could also indicate that package has not been installed locally.

8.5 Error 601: Data Not Found

When referencing a file using the `use`, `save`, `import delimited`, or `import excel` commands, you may receive this error:

```
. use E:\Users\username\Documents\data.dta
file E:\Users\username\Documents\data.dta not found
r(601);
```

This error indicates one of two conditions: (1) the file is not in the directory specified or, if only a file name was specified, is not in the working directory, or (2) there is a misspelling in either the file name or file path.