

Setting up Stata

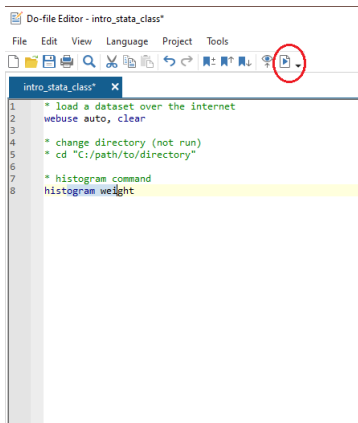
Imagine you are conducting a big statistical analysis. Stata lets you use every command you need. However, what happens when you want to save your work progress for later? Most (if not all) projects cannot be finished in one session. At least not, if you want some sleep in-between. But, Stata does not save your command history beyond closing the program. So what can you do – besides never closing Stata?

do-files: Scripts for your Commands

- ▶ You should always write your code in a **do-file**. It just makes everything easier for you!
- ▶ do-files are text files where users can store and run their commands for re-use, rather than retyping the commands into the **Command Window**. This guarantees two things:
 - ▶ Reproducibility: By looking at your code and using the same data, other researchers can get your exact results.
 - ▶ Debugging and changing commands is much easier.
- ▶ The file extension `.do` is used for do-files.
- ▶ To open a new do-file type the following in your **Command Window**:
`doedit`
- ▶ A text file editor will open up, ready for further instructions.

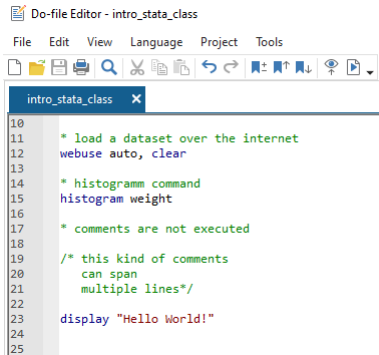
Running Commands from the do-file

- ▶ To run a specific command from the do-file, highlight part or all of the command, and then hit *Ctrl-D* (Mac:*Shift+Cmd+D*).
- ▶ Multiple commands can be selected and executed.
- ▶ If you want to execute your whole do-file from the beginning, be sure to not highlight anything and hit *Ctrl-D*.
- ▶ Stata will execute your code until the end of the file OR until it runs into an **error**. You can use that to your advantage. If you want to execute the code from the start until one specific line, just insert a ? (or any other random symbol) under that line. Stata will run into an error here and stop executing.



Writing a do-file

- ▶ In your do-file, Stata will display all your commands **blue**.
- ▶ But there is more to a reproducible statistical analysis than the commands. In order for others and your future-self to be able to understand what you are doing, you should **Comment** your code.
- ▶ They are colored **green**, so that you can distinguish them easily from commands. Stata does not execute comments.
- ▶ Words in quotes (directories, file names, strings) are colored **red**
- ▶ New versions of Stata feature an enhanced editor with tab auto-completion for Stata commands and previously typed words.



The screenshot shows the Stata Do-file Editor window titled "Do-file Editor - intro_stata_class". The window has a menu bar with "File", "Edit", "View", "Language", "Project", and "Tools". Below the menu bar is a toolbar with icons for file operations (new, open, save, print, find, cut, copy, paste, undo, redo), navigation (first, previous, next, last), and other functions (run, zoom in, zoom out, help). The editor area shows a file named "intro_stata_class" with the following content:

```
10
11      * load a dataset over the internet
12      webuse auto, clear
13
14      * histogram command
15      histogram weight
16
17      * comments are not executed
18
19      /* this kind of comments
20         can span
21         multiple lines*/
22
23      display "Hello World!"
24
25
```

Comments

- ▶ **Comments** are not executed, so they provide a way to **document your do-file**.
- ▶ We want to stress that **Comments** are an important part of your work. Especially beginners tend to neglect this feature. Trust us: if you want to be able to understand what you coded three weeks ago, you have to comment your code!
- ▶ `*` indicates that the whole line will be a comment.
- ▶ `/* */` will comment everything that is in-between these symbols. It lets you write comments over multiple lines.
- ▶ `//` indicates that everything that follows is a comment. You can use this to write a comment after a command in the same line.

```
5  * load a Stata dataset from the internet
6  webuse auto, clear
7
8  * change directory (not run)
9  * cd "C:\..."
10
11 * histogram command
12 histogram weight // histogram comment 2
13
14 * comments are not executed
15
16 /*
17  this kind of comment
18  can span
19  multiple lines
20  */
21
```

Know your limit

- ▶ You will see later that some Stata commands can become pretty long. So how does Stata know when your command ends?
- ▶ A sign or symbol that indicates the end of a command is called a **delimiter**.
- ▶ Stata will normally assume that a line break indicates the end of a command. So per default *Return* is your delimiter. You just can't see it.
- ▶ You can change your delimiter by using the command `#delimit` followed by the symbol that you want to use. For example, we can use `;` as a delimiter by typing:
`#delimit ;`
- ▶ To switch it back type `#delimit CR`

```
23
24 * use the #delimit command to continue a command over multiple lines
25 #delimit ;
26
27 summarize weight
28     length
29     ;
30 # delimit CR
31
32 * alternatively, use "///" to continue a command over multiple lines
33 summarize weight ///
34     length
35
```

It's not the end

- ▶ Instead of switching your delimiters back and forth, you can extend commands over multiple lines by placing `///` at the end of each line, except the last.
- ▶ Make sure to put a space before it.
- ▶ When executing, highlight each line in the command(s).

```
23
24 * use the #delimit command to continue a command over multiple lines
25 #delimit ;
26
27 summarize weight
28     length
29     ;
30 # delimit CR
31
32 * alternatively, use "///" to continue a command over multiple lines
33 summarize weight ///
34     length
35
```


Giving your do-file a head start

- ▶ As an eager economist, you want to start working right away. With our **do-file header**, you can do just that:

```
12
13 * do-file header
14 clear all // clear memory
15 set more off, perm // turn off step-wise display of long output (for ever)
16 set varabbrev off // turn off variable abbreviation
17
18 * Start log file
19 capture log close //closes any previously opened log-files
20 log using stata_intro, text replace // start a new log file
21 log off // suspend the log file for a while
22 log on // continue recording
23
24 * Set path
25 global path "C:\..."
26
27
```

- ▶ Just include your preferred **path** OR **working directory** and you are ready to go.

What's in the header?

- ▶ The **header** clears Stata's memory and makes sure that your output is not shown in a step-wise manner.
- ▶ Starting a **log file** at the beginning of a session is helpful, because it automatically records everything you do.
 - ▶ If you want to not record parts of your code just write `log off`
 - ▶ To start recording again, type `log on`
- ▶ You can also set **paths** so that Stata knows where your materials are. If you use different folders, just copy your folder structure to different paths:

```
global path "C:\...\MyStataProject"  
global folder1 "${path}\folder1"
```

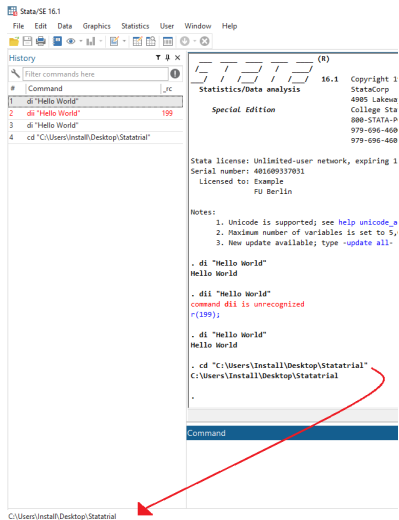
- ▶ This command creates a path to a folder named `"folder1"` within the folder `"MyStataProject"`. Make sure these folders exist on your hard drive.

Working Directory

- ▶ At the bottom left corner of the Stata window – below the **Review Window** – is the address of your **Working Directory**.
- ▶ Stata will only load from and save files to your Working directory.
- ▶ By using the command `cd` and specifying a directory from your computer you can change your Working directory.
- ▶ Let's try it out: Create a folder on your desktop and name it *Statatrial*. Right-click on that folder and copy its directory. Go into your **Command Window** and use

```
cd "CopiedDirectory\Statatrial"
```

Did Stata change your working directory successfully? Check in the corner of your window.



Stata packages

- ▶ Stata commands are part of packages. Your installed version of Stata comes with many packages that you need.
- ▶ However, there are many useful user-written packages that are not part of the main software.
- ▶ Try to find a command for estimating fixed effects ordered logit models following Baetschmann et al. (2015):
`findit baetschmann`
- ▶ You can install the package manually.

