13 Using the Do-file Editor—automating Stata

The Do-file Editor

Stata comes with an integrated text editor called the Do-file Editor, which can be used for many tasks. It gets its name from the term *do-file*, which is a file containing a list of commands for Stata to run (called a batch file or a script in other settings). See [U] **16 Do-files** for more information. Although the Do-file Editor has advanced features that can help in writing such files, it can also be used to build up a series of commands that can then be submitted to Stata all at once. This feature can be handy when writing a loop to process multiple variables in a similar fashion or when doing complex, repetitive tasks interactively.

To get the most from this chapter, you should work through it at your computer. Start by opening the do-file editor, either by clicking on the **Do-file Editor** button, , or by typing doedit in the Command window and pressing *Enter*.

The Do-file Editor toolbar

The Do-file Editor has 15 buttons. Many of the buttons share a similar purpose with their look-alikes in the main Stata toolbar.

If you ever forget what a button does, hover the mouse pointer over a button, and a tooltip will appear.

New: Open a new do-file in a new tab in the Do-file Editor.

Open: Open a do-file from disk in a new tab in the Do-file Editor.

Save: Save the current do-file to disk.

Print: Print the contents of the Do-file Editor.

Find: Open the *Find* dialog for finding text.

Cut: Cut the selected text and put it in the Clipboard.

Copy: Copy the selected text to the Clipboard.

Paste: Paste the text from the Clipboard into the current document.

Undo: Undo the last change.

Redo: Undo the last undo.

Toggle Bookmark: Turn on or off the bookmark on the current line. Bookmarks are a way to move quickly within the do-file. They are quite useful in long do-files or when debugging.

Previous Bookmark: Go to the previous bookmark (if any).

Next Bookmark: Go to the next bookmark (if any).

Show File in Viewer: Show the contents of the do-file in a Viewer window. This is worthwhile when editing files that contain SMCL tags, such as log files or help files.

Execute (do): Run the commands in the do-file, showing all commands and their output. If text is highlighted, the button becomes the **Execute Selection (do)** button and will run only the selected lines, showing all output. We will refer to this as the **Do** button.

Using the Do-file Editor

Suppose that we would like to analyze fuel usage for 1978 automobiles in a manner similar to what we did in [GSW] **1 Introducing Stata—sample session**. We know that we will be issuing many commands to Stata during our analysis and that we want to be able to reproduce our work later without having to type each command again.

We can do this easily in Stata: simply save a text file containing the commands. When that is done, we can tell Stata to run the file and execute each command in sequence. Such a file is known as a Stata *do-file*; see [U] **16 Do-files**.

To analyze fuel usage of 1978 automobiles, we would like to create a new variable containing gallons per mile. We would like to see how that variable changes in relation to vehicle weight for both domestic and imported cars. Performing a regression with our new variable would be a good first step.

To get started, click on the **Do-file Editor** button to open the Do-file Editor. After the Do-file Editor opens, type the commands below into the Do-file Editor. Purposely misspell the name of the `foreign` variable on the fifth line. (We are intentionally making some common mistakes and then pointing you to the solutions. This will save you time later.)

```
* an example do-file
sysuse auto
generate gp100m = 100/mpg
label var gp100m "Gallons per 100 miles"
regress gp100m weight foreing
```

Here is what your Do-file Editor should look like now:
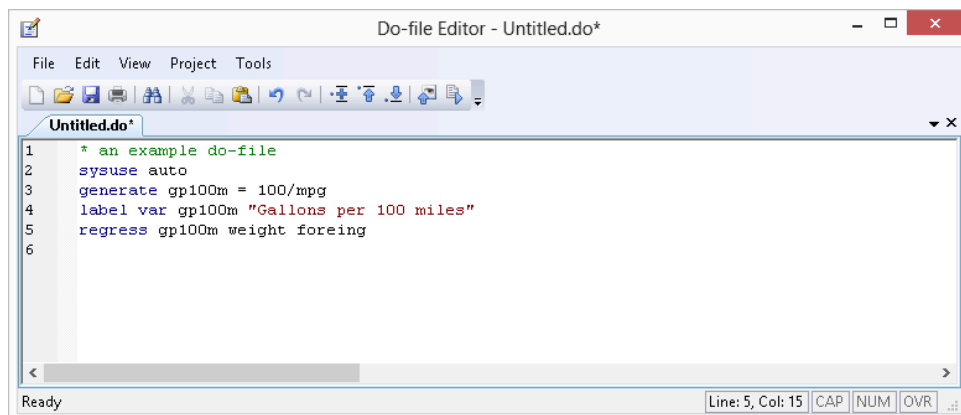
You will notice that the color of the text changes as you type. The different colors are examples of the Do-file Editor's *syntax highlighting*. The colors and text properties of the syntax elements can be changed by selecting **Edit > Preferences...** from the Do-file Editor menu bar and then clicking on the **Syntax Color** tab in the resulting window.

Click on the **Do** button, , to execute the commands. Stata executes the commands in sequence, and the results appear in the Results window:

```
. do C:\Users\mydir\AppData\Local\Temp\STD08000000.tmp

. * an example do-file
. sysuse auto
(1978 Automobile Data)

. generate gp100m = 100/mpg

. label var gp100m "Gallons per 100 miles"

. regress gp100m weight foreing
variable foreing not found
r(111);

.
end of do-file
```

The do "C:\ . . . " command is how Stata executes the commands in the Do-file Editor. Stata saves the commands to a temporary file and issues the do command to execute them. Everything worked as planned until Stata saw the misspelled variable. The first three commands were executed, but an error was produced on the fourth. Stata does not know of a variable named foreing. We need to go back to the Do-file Editor and change the misspelled variable name to foreign in the last line:
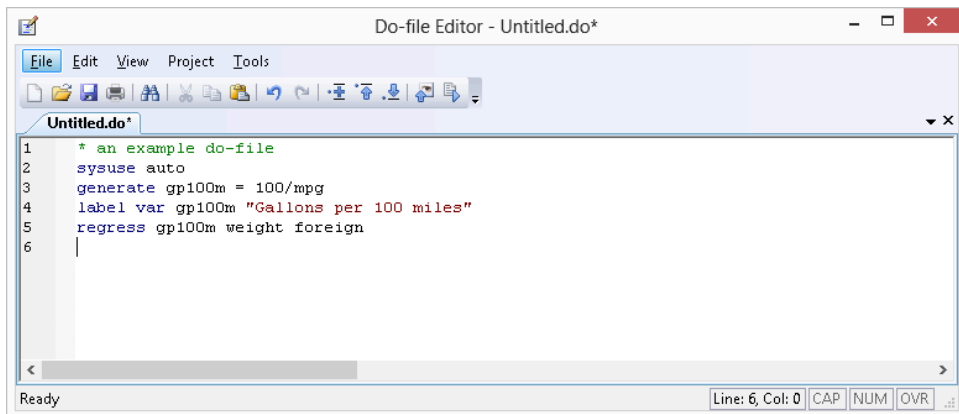
We click on the **Do** button again. Alas, Stata now fails on the first line—it will not overwrite the dataset in memory that we changed.

```
. do C:\Users\mydir\AppData\Local\Temp\STD08000000.tmp

. * an example do-file
. sysuse auto
no; data in memory would be lost
r(4);

.
end of do-file
```

We now have a choice for what we should do:

- We can put a `clear` command in our do-file as the very first command. This automatically clears out Stata's memory before the do-file tries to load the `auto` dataset. This is convenient but dangerous because it defeats Stata's protection against throwing away changes without warning.
- We can type a `clear` command in the Command window to manually clear the dataset and then process the do-file again. This process can be aggravating when building a complicated do-file.

Here is some advice: Automatically clear Stata's memory while debugging the do-file. Once the do-file is in its final form, decide the context in which it will be used. If it will be used in a highly automated environment (such as when certifying), the do-file should still automatically clear Stata's memory. If it will be used rarely, do not clear Stata's memory. This decision will save much heartache. We will add a `clear` option to the `sysuse` command to automatically clear the dataset in Stata's memory before the do-file runs:
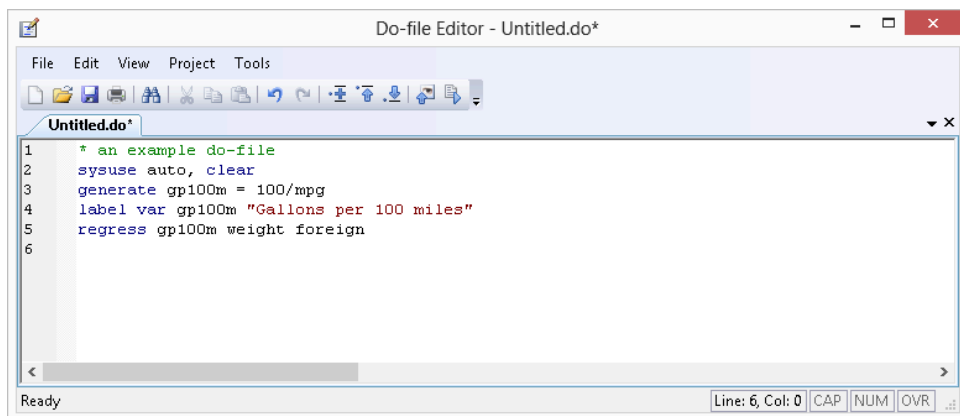
The do-file now runs well, as clicking on the **Do** button shows:

```
. do C:\Users\mydir\AppData\Local\Temp\STD08000000.tmp

. * an example do-file
. sysuse auto, clear
(1978 Automobile Data)

. generate gp100m = 100/mpg

. label var gp100m "Gallons per 100 miles"

. regress gp100m weight foreign

      Source |       SS       df       MS              Number of obs =      74
-------------+------------------------------           F(  2,    71) =  113.97
       Model | 91.1761694     2  45.5880847            Prob > F      =  0.0000
    Residual | 28.4000913    71  .400001287            R-squared     =  0.7625
-------------+------------------------------           Adj R-squared =  0.7558
       Total | 119.576261    73  1.63803097            Root MSE      =  .63246

------------------------------------------------------------------------------
      gp100m |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |   .0016254   .0001183    13.74   0.000     .0013896    .0018612
     foreign |   .6220535   .1997381     3.11   0.003     .2237871    1.02032
       _cons |  -.0734839   .4019932    -0.18   0.855    -.8750354    .7280677
------------------------------------------------------------------------------

.
end of do-file
```

You might want to select **File > Save As...** to save this do-file from the Do-file Editor. Later, you could select **File > Open** to open it and then add more commands as you move forward with your analysis. By saving the commands of your analysis in a do-file as you go, you do not have to worry about retyping them with each new Stata session. Think hard about removing the `clear` option from the first command.

After you have saved your do-file, you can execute the commands it contains by typing do *filename*, where the *filename* is the name of your do-file.

The File menu

The **File** menu of the Do-file Editor includes standard features found in most text editors. You may choose any of these menu items: create a **New > File**, **Open** an existing file, **Save** the current file, save the current file under a new name with **Save As...**, or **Print...** the current file. There are also buttons on the Do-file Editor's toolbar that correspond to these features.

You can also select **Insert File...** to insert the contents of another file at the current cursor position in the Do-file Editor.

Finally, you can create a **New > Project...** to keep track of collections of files used in a project. These can be do-files, data files, graph files, or any other files you like. For more information on the Project Manager, see [P] **Project Manager**.

The Edit menu

The **Edit** menu of the Do-file Editor includes the standard **Undo**, **Redo**, **Cut**, **Copy**, **Paste**, **Delete**, and **Find** capabilities. There are also buttons on the Do-file Editor's toolbar for easy access to these capabilities. There are several other **Edit** menu features that you might find useful:

- You may select the current line with **Select Line**.
- You may delete the current line with **Delete Line**.
- **Find > Go To Line...** will allow you to jump to a specific line number. The line numbers are displayed at the left of the Do-file Editor window.
- **Advanced** leads to a submenu with some programmer's friends:
  - **Shift Right** indents the selection by one tab.
  - **Shift Left** unindents the selection by one tab.
  - **Make Selection Uppercase** converts the selection to all capital letters.
  - **Make Selection Lowercase** converts the selection to all lowercase letters.
  - **Convert Line Endings to Mac OS X/Unix Format (\n)** converts the line endings for the current file to Mac OS X/Unix format.
  - **Convert Line Endings to Windows Format (\r\n)** converts the line endings for the current file to Windows format.

Matching and balancing of braces { }, is also available from the **Edit** menu. When you select **Edit > Find > Match Brace**, the Do-file Editor looks at the character immediately to the left and right of the cursor. If either is a brace, the editor will find the matching character and place the cursor immediately in front of it. If there is no match, the cursor will not move.

When you select **Edit > Find > Balance Braces**, the Do-file Editor looks to the left and right of the current cursor position or selection and creates a selection that includes the narrowest level of matching braces. If you select **Balance Braces** again, the editor will expand the selection to include the next level of matching braces. If there is no match, the cursor will not move. Balancing braces is useful for working with blocks of code defined by loops or if commands. See [P] **foreach**, [P] **forvalues**, [P] **while**, and [P] **if** for more information.

Balance Braces is easier to explain with an example. Type {now {is the} time} in the Do-file Editor. Place the cursor between the words is and the. Select **Edit > Find > Balance Braces**. The Do-file Editor will select {is the}. If you select **Balance Braces** again, the Do-file Editor will select {now {is the} time}.

Editing tip: You can click on the left margin near a line number to select the entire line and the end-of-line characters. Doing so makes it easy to delete lines or cut lines and paste them elsewhere. You can click and drag within the line-number column to select a range of complete lines.

The View menu

The View menu of the Do-File Editor allows you to zoom in and out or display special characters such as tabs and line endings.

The Tools menu

You have already learned about the **Do** button. Selecting **Tools > Execute (do)** is equivalent to clicking on the **Execute (do)** button.

Selecting **Tools > Execute (do) to Bottom** will send all the commands from the current line through the end of the contents of the Do-file Editor to the Command window. This method is a quick way to run a part of a do-file.

Do is equivalent to Stata's do command; see [U] **16 Do-files** for a complete discussion.

You can also preview files in the Viewer by selecting **Tools > Show File in Viewer** or by clicking on the **Show File in Viewer** button, 🖼. This feature is useful when working with files that use Stata's SMCL tags, such as when writing help files or editing log files.

Saving interactive commands from Stata as a do-file

While working interactively with Stata, you might decide that you would like to rerun the last several commands that you typed interactively. From the Review window, you can send highlighted commands or even the entire contents to the Do-file Editor. You can also save commands as a do-file and open that file in the Do-file Editor. You can copy a command from a dialog (rather than submit it) and paste it into the Do-file Editor. See [GSW] **6 Using the Data Editor** for details. Also see [R] **log** for information on the cmdlog command, which allows you to log all commands that you type in Stata to a do-file.

Projects

For advanced users managing many files as part of a project, Stata has a Project Manager that uses the Do-file Editor. For more information on the Project Manager, see [P] **Project Manager**.