10 Keyboard use

Contents

10.1 Description

The keyboard should operate much the way you would expect, with a few additions:

- There are some unexpected keys you can press to obtain previous commands you have typed. Also, you can click once on a command in the History window to reload it, or click on it twice to reload and execute; this feature is discussed in the *Getting Started* manuals.
- There are a host of command-editing features for Stata for Unix(console) users because their user interface does not offer such features.
- Regardless of operating system or user interface, if there are *F*-keys on your keyboard, they have special meaning and you can change the definitions of the keys.

10.2 F-keys

Windows users: *F3* and *F10* are reserved internally by Windows; you cannot program these keys.

By default, Stata defines the *F*-keys to mean

F-key	Definition
F1	`help advice;`
F2	`describe;`
F7	`save`
F8	`use`

The semicolons at the end of some entries indicate an implied *Enter*.

Stata provides several methods for obtaining help. To learn about these methods, select **Help > Advice**. Or you can just press *F1*.

`describe` is the Stata command to report the contents of data loaded into memory. It is explained in [D] **describe**. Normally, you type `describe` and press *Enter*. You can also press *F2*.

`save` is the command to save the data in memory into a file, and `use` is the command to load data; see [D] **use** and [D] **save**. The syntax of each is the same: `save` or `use` followed by a filename. You can type the commands or you can press *F7* or *F8* followed by the filename.

You can change the definitions of the *F*-keys. For instance, the command to list data is `list`; you can read about it in [D] **list**. The syntax is `list` to list all the data, or `list` followed by the names of some variables to list just those variables (there are other possibilities).

If you wanted *F9* to mean `list`, you could type

```
. global F9 "list "
```

In the above, `F9` refers to the letter *F* followed by *9*, not the *F9* key. Note the capitalization and spacing of the command.

You type `global` in lowercase, type `F9`, and then type `"list "`. The space at the end of `list` is important. In the future, rather than typing `list mpg weight`, you want to be able to press the *F9* key and then type only `mpg weight`. You put a space in the definition of `F9` so that you would not have to type a space in front of the first variable name after pressing *F9*.

Now say you wanted *F5* to mean list all the data—`list` followed by *Enter*. You could define

```
. global F5 "list;"
```

Now you would have two ways of listing all the data: press *F9*, and then press *Enter*, or press *F5*. The semicolon at the end of the definition of *F5* will press *Enter* for you.

If you really want to change the definitions of *F9* and *F5*, you will probably want to change the definition every time you invoke Stata. One way would be to type the two `global` commands every time you invoke Stata. Another way would be to type the two commands into a text file named `profile.do`. Stata executes the commands in `profile.do` every time it is launched if `profile.do` is placed in the appropriate directory:

Windows:	see [GSW] **B.3 Executing commands every time Stata is started**
Mac:	see [GSM] **B.1 Executing commands every time Stata is started**
Unix:	see [GSU] **B.1 Executing commands every time Stata is started**

You can use the *F*-keys any way you desire: they contain a string of characters, and pressing the *F*-key is equivalent to typing those characters.

❑ Technical note

[*Stata for Unix(console) users.*] Sometimes Unix assigns a special meaning to the *F*-keys, and if it does, those meanings supersede our meanings. Stata provides a second way to get to the *F*-keys. Press *Ctrl+F*, release the keys, and then press a number from `0` through `9`. Stata interprets *Ctrl+F* plus 1 as equivalent to the *F1* key, *Ctrl+F* plus 2 as *F2*, and so on. *Ctrl+F* plus 0 means *F10*. These keys will work only if they are properly mapped in your `termcap` or `terminfo` entry.

❑

❑ Technical note

On some international keyboards, the left single quote is used as an accent character. In this case, we recommend mapping this character to one of your function keys. In fact, you might find it convenient to map both the left single quote (‘) and the right single quote (’) characters so that they are next to each other.

Within Stata, open the Do-file Editor. Type the following two lines in the Do-file Editor:

```
global F4 ‘
global F5 ’
```

Save the file as `profile.do` into your Stata directory. If you already have a `profile.do` file, append the two lines to your existing `profile.do` file.

Exit Stata and restart it. You should see the startup message

```
running C:\Program Files\Stata17\profile.do ...
```

or some variant of it depending on where your Stata is installed. Press *F4* and *F5* to verify that they work.

If you did not see the startup message, you did not save the `profile.do` in your home folder.

You can, of course, map to any other function keys, but *F1*, *F2*, *F7*, and *F8* are already used.

❑

10.3 Editing keys in Stata

Users have available to them the standard editing keys for their operating system. So, Stata should just edit what you type in the natural way—the Stata Command window is a standard edit window.

Also, you can fetch commands from the History window into the Command window. Click on a command in the History window, and it is loaded into the Command window, where you can edit it. Alternatively, if you double-click on a line in the History window, it is loaded and executed.

Another way to get lines from the History window into the Command window is with the *PgUp* and *PgDn* keys. Press *PgUp* and Stata loads the last command you typed into the Command window. Press it again and Stata loads the line before that, and so on. *PgDn* goes in the opposite direction.

Another editing key that interests users is *Esc*. This key clears the Command window.

In summary,

Press	Result
PgUp	Steps back through commands and moves command from History window to Command window
PgDn	Steps forward through commands and moves command from History window to Command window
Esc	Clears Command window

10.4 Editing keys in Stata for Unix(console)

Certain keys allow you to edit the line that you are typing. Because Stata supports a variety of computers and keyboards, the location and the names of the editing keys are not the same for all Stata users.

Every keyboard has the standard alphabet keys (*QWERTY* and so on), and every keyboard has a *Ctrl* key. Some keyboards have extra keys located to the right, above, or left, with names like *PgUp* and *PgDn*.

Throughout this manual we will refer to Stata's editing keys using names that appear on nobody's keyboard. For instance, PrevLine is one of the Stata editing keys—it retrieves a previous line. Hunt all you want, but you will not find it on your keyboard. So, where is PrevLine? We have tried to put it where you would naturally expect it. On keyboards with a key labeled *PgUp*, *PgUp* is the PrevLine key, but on everybody's keyboard, no matter which version of Unix, brand of keyboard, or anything else, *Ctrl+R* also means PrevLine.

When we say press PrevLine, now you know what we mean: press *PgUp* or *Ctrl+R*. The editing keys are the following:

Name for editing key	Editing key	Function
Kill	*Esc* on PCs and *Ctrl+U*	Deletes the line and lets you start over.
Dbs	*Backspace* on PCs and *Backspace* or *Delete* on other computers	Backs up and deletes one character.
Lft	←, *4* on the numeric keypad for PCs, and *Ctrl+H*	Moves the cursor left one character without deleting any characters.
Rgt	→, *6* on the numeric keypad for PCs, and *Ctrl+L*	Moves the cursor forward one character.
Up	↑, *8* on the numeric keypad for PCs, and *Ctrl+O*	Moves the cursor up one physical line on a line that takes more than one physical line. Also see PrevLine.
Dn	↓, *2* on the numeric keypad for PCs, and *Ctrl+N*	Moves the cursor down one physical line on a line that takes more than one physical line. Also see NextLine.
PrevLine	*PgUp* and *Ctrl+R*	Retrieves a previously typed line. You may press PrevLine multiple times to step back through previous commands.
NextLine	*PgDn* and *Ctrl+B*	The inverse of PrevLine.
Seek	*Ctrl+Home* on PCs and *Ctrl+W*	Goes to the line number specified. Before pressing Seek, type the line number. For instance, typing *3* and then pressing Seek is the same as pressing PrevLine three times.
Ins	*Ins* and *Ctrl+E*	Toggles insert mode. In insert mode, characters typed are inserted at the position of the cursor.
Del	*Del* and *Ctrl+D*	Deletes the character at the position of the cursor.
Home	*Home* and *Ctrl+K*	Moves the cursor to the start of the line.
End	*End* and *Ctrl+P*	Moves the cursor to the end of the line.
Hack	*Ctrl+End* on PCs, and *Ctrl+X*	Hacks off the line at the cursor.
Tab	→| on PCs, *Tab*, and *Ctrl+I*	Expand variable name.
Btab	|← on PCs, and *Ctrl+G*	The inverse of Tab.

▷ Example 1

It is difficult to demonstrate the use of editing keys in print. You should try each of them. Nevertheless, here is an example:

```
. summarize price waht
```

You typed `summarize price waht` and then pressed the *Lft* key (← key or *Ctrl+H*) three times to maneuver the cursor back to the `a` of `waht`. If you were to press *Enter* right now, Stata would see the command `summarize price waht`, so where the cursor is does not matter when you press *Enter*. If you wanted to execute the command `summarize price`, you could back up one more character and then press the Hack key. We will assume, however, that you meant to type `weight`.

If you were now to press the letter *e* on the keyboard, an `e` would appear on the screen to replace the `a`, and the cursor would move under the character `h`. We now have `weht`. You press *Ins*, putting Stata into insert mode, and press *i* and *g*. The line now says `summarize price weight`, which is

correct, so you press *Enter*. We did not have to press *Ins* before every character we wanted to insert. The *Ins* key is a toggle: If we press it again, Stata turns off insert mode, and what we type replaces what was there. When we press *Enter*, Stata forgets all about insert mode, so we do not have to remember from one command to the next whether we are in insert mode.

◁

❑ Technical note

Stata performs its editing magic from the information about your terminal recorded in `/etc/termcap`(5) or, under System V, `/usr/lib/terminfo`(4). If some feature does not appear to work, the entry for your terminal in the `termcap` file or `terminfo` directory is probably incorrect. Contact your system administrator.

❑

10.5 Editing previous lines in Stata

In addition to what is said below, remember that the History window also shows the contents of the review buffer.

One way to retrieve lines is with the PrevLine and NextLine keys. Remember, PrevLine and NextLine are the names we attach to these keys—there are no such keys on your keyboard. You have to look back at the previous section to find out which keys correspond to PrevLine and NextLine on your computer. To save you the effort this time, PrevLine probably corresponds to *PgUp* and NextLine probably corresponds to *PgDn*.

Suppose you wanted to reissue the third line back. You could press PrevLine three times and then press *Enter*. If you made a mistake and pressed PrevLine four times, you could press NextLine to go forward in the buffer. You do not have to count lines because, each time you press PrevLine or NextLine, the current line is displayed on your monitor. Simply press the key until you find the line you want.

Another method for reviewing previous lines, `#review`, is convenient for Unix(console) users.

▷ Example 2

Typing `#review` by itself causes Stata to list the last five commands you typed. For instance,

```
. #review
5 list make mpg weight if abs(res)>6
4 list make mpg weight if abs(res)>5
3 tabulate foreign if abs(res)>5
2 regress mpg weight weight2
1 test weight2=0
. _
```

We can see from the listing that the last command typed by the user was `test weight2=0`. Or, you may just look at the History window to see the history of commands you typed.

◁

▷ Example 3

Perhaps the command you are looking for is not among the last five commands you typed. You can tell Stata to go back any number of lines. For instance, typing `#review 15` tells Stata to show you the last 15 lines you typed:

```
. #review 15
15 replace resmpg=mpg-pred
14 summarize resmpg, detail
13 drop predmpg
12 describe
11 sort foreign
10 by foreign: summarize mpg weight
9 * lines that start with a * are comments.
8 * they go into the review buffer too.
7 summarize resmpg, detail
6 list make mpg weight
5 list make mpg weight if abs(res)>6
4 list make mpg weight if abs(res)>5
3 tabulate foreign if abs(res)>5
2 regress mpg weight weight2
1 test weight2=0
. _
```

If you wanted to resubmit the 10th previous line, you could type `10` and press Seek, or you could press PrevLine 10 times. No matter which of the above methods you prefer for retrieving lines, you may edit previous lines by using the editing keys.

◁

10.6 Tab expansion of variable names

Another way to quickly enter a variable name is to take advantage of Stata's tab-completion feature. Simply type the first few letters of the variable name in the Command window and press the *Tab* key. Stata will automatically type the rest of the variable name for you. If more than one variable name matches the letters you have typed, Stata will complete as much as it can and beep at you to let you know that you have typed a nonunique variable abbreviation.

The tab-completion feature also applies to typing filenames. If you start by typing a double quote, `"`, you can type the first few letters of a filename or directory and press the *Tab* key. Stata will automatically type the rest of the name for you. If more than one filename or directory matches the letters you have typed, Stata will complete as much as it can and beep at you to let you know that you have typed a nonunique abbreviation. After the entire filename or directory has been typed, type another double quote.