Title

stata.com

shell — Temporarily invoke operating system

Description

`shell` (synonym: "`!`") allows you to send commands to your operating system or to enter your operating system for interactive use. Stata will wait for the shell to close or the *operating_system_command* to complete before continuing.

`winexec` allows you to start other programs (such as browsers) from Stata's command line. Stata will continue without waiting for the program to complete.

`xshell` (Stata for Mac and Unix(GUI) only) brings up an `xterm` window in which the command is to be executed.

Syntax

{ `shell` | `!` } [*operating_system_command*]

`winexec` *program_name* [*program_args*]

{ `xshell` | `!!` } [*operating_system_command*]

Command availability:

	Stata for ...			
Command	Windows	Mac	Unix(GUI)	Unix(console)
`shell`	X	X	X	X
`winexec`	X	X	X	–
`xshell`	–	X	X	–

Remarks and examples

stata.com

Remarks are presented under the following headings:

Stata for Windows
Stata for Mac
Stata for Unix(GUI)
Stata for Unix(console)

Stata for Windows

`shell`, without arguments, preserves your session and invokes the operating system. Stata's Command window will disappear, and a Windows command prompt will appear, indicating that you may not continue in Stata until you exit the Windows command prompt. To reenter Stata, type `exit` at your operating system's prompt. Your Stata session is reestablished just as if you had never left.

Say that you are using Stata for Windows and you suddenly realize you need to do two things. You need to enter your operating system for a few minutes. Rather than exiting Stata, doing what you have to do, and then restarting Stata, you type `shell` in the Command window. A Windows command prompt appears:

```
C:\data>
```

You can now do whatever you need to do in Windows, and Stata will wait until you exit the Windows command prompt before continuing.

Experienced Stata users seldom type out the word `shell`. They type "`!`". Also you do not have to enter your operating system, issue a command, and then exit back to Stata. If you want to execute one command, you can type the command right after the word `shell` or the exclamation point:

```
. !rename try15.dta final.dta
```

If you do this, the Windows command prompt will open and close as the command is executed.

Stata for Windows users can also use the `winexec` command, which allows you to launch any Windows application from within Stata. You can think of it as a shortcut for clicking on the Windows **Start** button, choosing **Run...**, and typing a command.

Assume that you are working in Stata and decide that you want to run a text editor:

```
. winexec notepad
```

(*The Windows application Notepad will start and run at the same time as Stata*)

You could even pass a filename to your text editor:

```
. winexec notepad c:\docs\myfile.txt
```

You may need to specify a complete path to the executable that you wish to launch:

```
. winexec c:\windows\notepad c:\docs\myfile.txt
```

The important difference between `winexec` and `shell` is that Stata does not wait for whatever program `winexec` launches to complete before continuing. Stata will wait for the program `shell` launches to complete before performing any further commands.

Stata for Mac

`shell`, with arguments, invokes your operating system, executes one command, and redirects the output to the Results window. The command must complete before you can enter another command in the Command window.

Say that you are using Stata for Mac and suddenly realize that there are two things you have to do. You need to switch to the Finder or enter commands from a terminal for a few minutes. Rather than exiting Stata, doing what you have to do, and then switching back to Stata, you type `shell` and the command in the Command window to execute one command. You then repeat this step for each command that you want to execute from the shell.

Experienced Stata users seldom type out the word `shell`. They type "`!`".

```
. !mv try15.dta final.dta
```

Be careful not to execute commands, such as `vi`, that require interaction from you. Because all output is redirected to Stata's Results window, you will not be able to interact with the command from Stata. This will effectively lock up Stata because the command will never complete.

When you type `xshell vi myfile.do`, Stata invokes an `xterm` window (which in turn invokes a `shell`) and executes the command there. Typing `!!vi myfile.do` is equivalent to typing `xshell vi myfile.do`.

❑ Technical note

On macOS, `xterm` is available when *X11* is installed. To install *X11*, you must first download XQuartz from http://xquartz.macosforge.org/.

❑

Stata for Mac users can also use the `winexec` command, which allows you to launch any native application from within Stata. You may, however, have to specify the absolute path to the application. If the application you wish to launch is a macOS application bundle, you must specify an absolute path to the executable in the bundle.

Assume that you are working in Stata and decide that you want to run a text editor:

```
. winexec /Applications/TextEdit.app/Contents/MacOS/TextEdit
```

(*The macOS application TextEdit will start and run at the same time as Stata*)

You could even pass a filename to your text editor:

```
. winexec /Applications/TextEdit.app/Contents/MacOS/TextEdit
> /Users/cnguyen/myfile.do
```

If you specify a file path as an argument to the program to be launched, you must specify an absolute path. Also using `~` in the path will not resolve to a home directory. If an application cannot be launched from a terminal window, it cannot be launched by `winexec`.

The important difference between `winexec` and `shell` is that Stata does not wait for whatever program `winexec` launches to complete before continuing. Stata will wait for the program `shell` launches to complete before performing any further commands. `shell` is appropriate for executing shell commands; `winexec` is appropriate for launching applications.

Stata for Unix(GUI)

`shell`, without arguments, preserves your session and invokes the operating system. The Command window will disappear, and an `xterm` window will appear, indicating that you may not do anything in Stata until you exit the `xterm` window. To reenter Stata, type `exit` at the Unix prompt. Your Stata session is reestablished just as if you had never left.

Say that you are using Stata for Unix(GUI) and suddenly realize that you need to do two things. You need to enter your operating system for a few minutes. Rather than exiting Stata, doing what you have to do, and then restarting Stata, you type `shell` in the Command window. An `xterm` window will appear:

```
mycomputer$ _
```

You can now do whatever you need to do, and Stata will wait until you exit the window before continuing.

Experienced Stata users seldom type out the word `shell`. They type "`!`". Also you do not have to enter your operating system, issue a command, and then exit back to Stata. If you want to execute one command, you can type the command right after the word `shell` or the exclamation point:

```
. !mv try15.dta final.dta
```

Be careful because sometimes you will want to type

```
. !!vi myfile.do
```

and in other cases,

```
. winexec xedit myfile.do
```

`!!` is a synonym for `xshell`—a command different from, but related to, `shell`—and `winexec` is a different and related command, too.

Before we get into this, understand that if all you want is a shell from which you can issue Unix commands, type `shell` or `!`:

```
. !
mycomputer$ _
```

When you are through, type `exit` to the Unix prompt, and you will return to Stata:

```
mycomputer$ exit
. _
```

If, on the other hand, you want to specify in Stata the Unix command that you want to execute, you need to decide whether you want to use `shell`, `xshell`, or `winexec`. The answer depends on whether the command you want to execute requires a terminal window or is an X application:

... does not need a terminal window:	use `shell` ... (synonym: `!`...)
... needs a terminal window:	use `xshell` ... (synonym: `!!`...)
... is an X application:	use `winexec` ... (no synonym)

When you type `shell mv try15.dta final.dta`, Stata invokes your shell (`/bin/sh`, `/bin/csh`, etc.) and executes the specified command (`mv` here), routing the standard output and standard error back to Stata. Typing `!mv try15.dta final.dta` is the same as typing `shell mv try15.dta final.dta`.

When you type `xshell vi myfile.do`, Stata invokes an `xterm` window (which in turn invokes a shell) and executes the command there. Typing `!!vi myfile.do` is equivalent to typing `xshell vi myfile.do`.

When you type `winexec xedit myfile.do`, Stata directly invokes the command specified (`xedit` here). No `xterm` window is brought up nor is a shell invoked because, here, `xterm` does not need it. `xterm` is an X application that will create its own window in which to run. You could have typed `!!xedit myfile.do`. That would have brought up an unnecessary `xterm` window from which `xedit` would have been executed, and that would not matter. You could even have typed `!xedit myfile.do`. That would have invoked an unnecessary shell from which `xedit` would have been executed, and that would not matter, either. The important difference, however, is that `shell` and `xshell` wait until the process completes before allowing Stata to continue, and `winexec` does not.

❑ Technical note

You can set Stata global macros to control the behavior of `shell` and `xshell`. The macros are

`$S_SHELL`	defines the shell to be used by `shell` when you type a command following `shell`. The default is something like "`/bin/sh -c`", although this can vary, depending on how your Unix environment variables are set.
`$S_XSHELL`	defines shell to be used by `shell` and `xshell` when they are typed without arguments. The default is "`xterm`".
`$S_XSHELL2`	defines shell to be used by `xshell` when it is typed with arguments. The default is "`xterm -e`".

For instance, if you type in Stata

```
. global S_XSHELL2 "/usr/X11R6/bin/xterm -e"
```

and then later type

```
. !!vi myfile.do
```

then Stata would issue the command `/usr/X11R6/bin/xterm -e vi myfile.do` to Unix.

If you do make changes, we recommend that you record the changes in your `profile.do` file.

❑

Stata for Unix(console)

`shell`, without arguments, preserves your session and then invokes your operating system. Your Stata session will be suspended until you `exit` the shell, at which point your Stata session is reestablished just as if you had never left.

Say that you are using Stata and you suddenly realize that you need to do two things. You need to enter your operating system for a few minutes. Rather than exiting Stata, doing what you have to do, and then restarting Stata, you type `shell`. A Unix prompt appears:

```
. shell
(Type exit to return to Stata)
$ _
```

You can now do whatever you need to do and type `exit` when you finish. You will return to Stata just as if you had never left.

Experienced Stata users seldom type out the word `shell`. They type '`!`'. Also you do not have to enter your operating system, issue a command, and then exit back to Stata. If you want to execute one command, you can type the command right after the word `shell` or the exclamation point. If you want to edit the file `myfile.do`, and if `vi` is the name of your favorite editor, you could type

```
. !vi myfile.do
```

Stata opens your editor.
When you exit your editor:

```
. _
```

Reference

Huber, C. 2014. How to create animated graphics using Stata. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2014/03/24/how-to-create-animated-graphics-using-stata/.

Also see

[D] **cd** — Change directory

[D] **copy** — Copy file from disk or URL

[D] **dir** — Display filenames

[D] **erase** — Erase a disk file

[D] **mkdir** — Create directory

[D] **rmdir** — Remove directory

[D] **type** — Display contents of a file