Title

stata.com

quietly — Quietly and noisily perform Stata command

Description Syntax Remarks and examples Also see

Description

`quietly` suppresses all terminal output for the duration of *command*. It is useful both interactively and in programs.

`noisily` turns back on terminal output, if appropriate, for the duration of *command*. It is useful only in programs.

`set output` specifies the output to be displayed. It is useful only in programs and even then is seldom used.

Syntax

Perform command but suppress terminal output

`quietly` [`:`] *command*

Perform command and ensure terminal output

`noisily` [`:`] *command*

Specify type of output to display

`set output` { `proc` | `inform` | `error` }

Remarks and examples

stata.com

Remarks are presented under the following headings:

quietly used interactively

▷ Example 1

`quietly` is useful when you are using Stata interactively and want to temporarily suppress the terminal output. For instance, to estimate a regression of `mpg` on the variables `weight`, `foreign`, and `headroom` and to suppress the terminal output, type

```
. use https://www.stata-press.com/data/r17/auto
(1978 automobile data)
. quietly regress mpg weight foreign headroom
. _
```

Admittedly, it is unlikely that you would ever want to do this in real life.

◁

quietly used in programs

❑ Technical note

quietly is often used in programs. Say that you have the following program to run a regression of y on x, calculate the residuals, and then list the outliers, which are defined as points with residuals below the 5th percentile or above the 95th percentile:

```
program myprog
        regress ‘1’ ‘2’
        predict resid, resid
        sort resid
        summarize resid, detail
        list ‘1’ ‘2’ resid if resid< r(p5) | resid> r(p95)
        drop resid
end
```

Although the program will work, it will also fill the screen with the regression output, any notes that predict feels obligated to mention, and the detailed output from summarize. A better version of this program might read

```
program myprog
        quietly regress ‘1’ ‘2’
        quietly predict resid, resid
        quietly sort resid
        quietly summarize resid, detail
        list ‘1’ ‘2’ resid if resid< r(p5) | resid> r(p95)
        drop resid
end
```

You can also combine quietly with {}:

```
program myprog
        quietly {
                regress ‘1’ ‘2’
                predict resid, resid
                sort resid
                summarize resid, detail
        }
        list ‘1’ ‘2’ resid if resid< r(p5) | resid> r(p95)
        drop resid
end
```

❑

❑ Technical note

noisily is the antonym of quietly, and it too can be used in programs and do-files. In fact, that is its only real use. We could recode our example program to read as follows:

```
program myprog
        quietly {
                regress ‘1’ ‘2’
                predict resid, resid
                sort resid
                summarize resid, detail
                noisily list ‘1’ ‘2’ resid if resid< r(p5) | resid> r(p95)
                drop resid
        }
end
```

Here we have not improved readability.

❑

❑ Technical note

`noisily` is not really the antonym of `quietly`. If the user types `quietly myprog yvar xvar`, the output will be suppressed because that is what the user wants. Here a `noisily` inside `myprog` will not display the output—`noisily` means noisily only if the program was allowed to be noisy when it was invoked.

❑

❑ Technical note

If you think you understand all this, take the following test. Is there any difference between `quietly do` *filename* and `run` *filename*? How about `noisily run` *filename* and `do` *filename*? What would happen if you typed `quietly noisily summarize` *myvar*? If you typed `noisily quietly summarize` *myvar*?

When you are ready, we will tell you the answers.

`quietly do` *filename* is equivalent to `run` *filename*. Typing `run` is easier, however.

`noisily run` *filename* is not at all the same as `do` *filename*. `run` produces no output, and no matter how noisily you run `run`, it is still quiet.

Typing `quietly noisily summarize` *myvar* is the same as typing `summarize myvar`. Think of it as `quietly {noisily summarize` *myvar*`}`. It is the inside `noisily` that takes precedence.

Typing `noisily quietly summarize` *myvar* is the same as typing `quietly summarize` *myvar*—it does nothing but burn computer time. Again it is the inside term, `quietly` this time, that takes precedence.

❑

❑ Technical note

`set output proc` means that all output, including procedure (command) output, is displayed. `inform` suppresses procedure output but displays informative messages and error messages. `error` suppresses all output except error messages. In practice, `set output` is seldom used.

❑

Note for programmers

If you write a program or ado-file, say, `mycmd`, there is nothing special you need to do so that your command can be prefixed with `quietly`. That said, c-class value `c(noisily)` (see [P] **creturn**) will return `0` if output is being suppressed and `1` otherwise. Thus your program might read

```
program mycmd
   ...
   display ...
   display ...
   ...
end
```

or

```
program mycmd
   ...
   if c(noisily) {
      display ...
      display ...
   }
   ...
end
```

The first style is preferred. If the user executes `quietly mycmd`, the output from `display` itself, along with the output of all other commands, will be automatically suppressed.

If the program must work substantially to produce what is being displayed, however, and the only reason for doing that work is because of the display, then the second style is preferred. In such cases, you can include the extra work within the block of code executed only when `c(noisily)` is true and thus make your program execute more quickly when it is invoked `quietly`.

Also see

[P] **capture** — Capture return code

[U] 18 Programming Stata