8 Error messages and return codes

Contents

8.1 Making mistakes

When an error occurs, Stata produces an error message and a *return code*. For instance,

```
. list myvar
no variables defined
r(111);
```

We ask Stata to list the variable named `myvar`. Because we have no data in memory, Stata responds with the message "no variables defined" and a line that reads "`r(111)`".

The "no variables defined" is called the error message.

The 111 is called the return code. You can click on blue return codes to get a detailed explanation of the error.

8.1.1 Mistakes are forgiven

After "no variables defined" and `r(111)`, all is forgiven; it is as if the error never occurred.

Typically, the message will be enough to guide you to a solution, but if it is not, the numeric return codes are documented in [P] **error**.

8.1.2 Mistakes stop user-written programs and do-files

Whenever an error occurs in a user-written program or do-file, the program or do-file immediately stops execution and the error message and return code are displayed.

For instance, consider the following do-file:

begin myfile.do

```
use https://www.stata-press.com/data/r17/auto
decribe
list
```

end myfile.do

Note the second line—you meant to type `describe` but typed `decribe`. Here is what happens when you execute this do-file by typing `do myfile`:

```
. do myfile

. use https://www.stata-press.com/data/r17/auto
(1978 automobile data)
```

```
. decribe
command decribe is unrecognized
r(199);

end of do-file
r(199);

. _
```

The first error message and return code were caused by the illegal `decribe`. This then caused the do-file itself to be aborted; the valid `list` command was never executed.

8.1.3 Advanced programming to tolerate errors

Errors are not only of the typographical kind; some are substantive. A command that is valid in one dataset might not be valid in another. Moreover, in advanced programming, errors are sometimes anticipated: use one dataset if it is there, but use another if you must.

Programmers can access the return code to determine whether an error occurred, which they can then ignore, or, by examining the return code, code their programs to take the appropriate action. This is discussed in [P] **capture**.

You can also prevent do-files from stopping when errors occur by using the `do` command's `nostop` option.

```
. do myfile, nostop
```

8.2 The return message for obtaining command timings

In addition to error messages and return codes, there is something called a return message, which you normally do not see. Normally, if you typed `summarize tempjan`, you would see

```
. use https://www.stata-press.com/data/r17/citytemp
(City temperature data)
. summarize tempjan
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
     tempjan |        954    35.74895    14.18813         2.2       72.6
```

If you were to type

```
. set rmsg on
r; t=0.00 10:21:22
```

sometime during your session, Stata would display return messages:

```
. summarize tempjan
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
     tempjan |        954    35.74895    14.18813         2.2       72.6
r; t=0.01 10:21:26
```

The line that reads `r; t=0.01 10:21:26` is called the return message.

The `r;` indicates that Stata successfully completed the command.

The `t=0.01` shows the amount of time, in seconds, it took Stata to perform the command (timed from the point you pressed *Enter* to the time Stata typed the message). This command took a hundredth of a second. Stata also shows the time of day with a 24-hour clock. This command completed at 10:21 a.m.

Stata can run commands stored in files (called do-files) and can log output. Some users find the detailed return message helpful with do-files. They construct a long program and let it run overnight, logging the output. They come back the next morning, look at the output, and discover a mistake in some portion of the job. They can look at the return messages to determine how long it will take to rerun that portion of the program.

You may `set rmsg on` whenever you wish.

When you want Stata to stop displaying the detailed return message, type `set rmsg off`.

There is another way to obtain timings of subsets of code that is of interest to programmers. See [P] **timer**.