Title

stata.com

errors — Error codes

Description Remarks and examples Also see

Description

When an error occurs, Mata presents a number as well as text describing the problem. The codes are presented below.

Also the error codes can be used as an argument with `_error()`, see [M-5] **error( )**.

Mata's error codes are a special case of Stata's return codes. In particular, they are the return codes in the range 3000–3999. In addition to the 3000-level codes, it is possible for Mata functions to generate any Stata error message and return code.

Remarks and examples

stata.com

Error messages in Mata break into two classes: errors that occur when the function is compiled (code 3000) and errors that occur when the function is executed (codes 3001–3999).

Compile-time error messages look like this:

```
: 2,,3
invalid expression
r(3000);

: "this" + "that
mismatched quotes
r(3000);
```

The text of the message varies according to the error made, but the error code is always 3000.

Run-time errors look like this:

```
: myfunction(2,3)
             solve():  3200  conformability error
             mysub():     -  function returned error
       myfunction():      -  function returned error
             <istmt>:     -  function returned error
r(3200);
```

The output is called a traceback log. Read from bottom to top, it says that what we typed (the `<istmt>`) called `myfunction()`, which called `mysub()`, which called `solve()` and, at that point, things went wrong. The error is possibly in `solve()`, but because `solve()` is a library function, it is more likely that the error is in how `mysub()` called `solve()`. Moreover, the error is seldom in the program listed at the top of the traceback log because the log lists the identity of the program that detected the error. Say `solve()` did have an error. Then the traceback log would probably have read something like

```
                   *:  3200  conformability error
             solve():     -  function returned error
             mysub():     -  function returned error
       myfunction():      -  function returned error
             <istmt>:     -  function returned error
```

The above log says the problem was detected by * (the multiplication operator), and at that point, solve() would be suspect, because one must ask, why did solve() use the multiplication operator incorrectly?

In any case, let's assume that the problem is not with solve(). Then you would guess the problem lies with mysub(). If you have used mysub() in many previous programs without problems, however, you might now shift your suspicion to myfunction(). If myfunction() is always trustworthy, perhaps you should not have typed myfunction(2,3). That is, perhaps you are using myfunction() incorrectly.

The error codes

3000. (message varies)
There is an error in what you have typed. Mata cannot interpret what you mean.

3001. incorrect number of arguments
The function expected, say, three arguments and received two, or five. Or the function allows between three and five arguments, but you supplied too many or too few. Fix the calling program.

3002. identical arguments not allowed
You have called a function specifying the same variable more than once. Usually this would not be a problem, but here, it is, usually because the supplied arguments are matrices that the function wishes to overwrite with different results. For instance, say function *f*(*A*, *B*, *C*) examines matrix *A* and returns a calculation based on *A* in *B* and *C*. The function might well complain that you specified the same matrix for *B* and *C*.

3010. attempt to dereference NULL pointer
The program made reference to **s*, and *s* contains NULL; see [M-2] **pointers**.

3011. invalid lval
In an assignment, what appears on the left-hand side of the equals is not something to which a value can be assigned; see [M-2] **op_assignment**.

3012. undefined operation on pointer
You have, for instance, attempted to add two pointers; see [M-2] **pointers**.

3020. class child/parent compiled at different times
One class is being used to extend another class, and the parent has been updated since the class was compiled.

3021. class compiled at different times
A function using a predefined class has not been recompiled since the class has last been changed.

3022. function not supported on this platform
You have tried to use a function that is defined for some operating system or flavor supported by Stata but not for the one you are currently using. For example, you may have tried to use a Mac-only function in Stata for Windows.

3101. matrix found where function required
A particular argument to a function is required to be a function, and a matrix was found instead.

3102. `function found where matrix required`
A particular argument to a function is required to be a matrix, vector, or scalar, and a function was found instead.

3103. `view found where array required`
In general, view matrices can be used wherever a matrix is required, but there are a few exceptions, both in low-level routines and in routines that wish to write results back to the argument. Here a view is not acceptable. If V is the view variable, simply code $X = V$ and then pass X in its stead. See [M-5] **st_view()**.

3104. `array found where view required`
A function argument was specified with a matrix that was not a view, and a view was required. See [M-5] **st_view()**.

3200. `conformability error`
A matrix, vector, or scalar has the wrong number of rows and/or columns for what is required. Adding a 2×3 matrix to a 1×4 would result in this error.

3201. `vector required`
An argument is required to be $r \times 1$ or $1 \times c$, and a matrix was found instead.

3202. `rowvector required`
An argument is required to be $1 \times c$ and it is not.

3203. `colvector required`
An argument is required to be $r \times 1$ and it is not.

3204. `matrix found where scalar required`
An argument is required to be 1×1 and it is not.

3205. `square matrix required`
An argument is required to be $n \times n$ and it is not.

3206. `invalid use of view containing op.vars`
Factor variables have been used in a view, and the view is now being used in a context that does not support the use of factor variables.

3208. `more than 2 billion rows or columns (LAPACK)`
A call was made to a `LAPACK()` function with a matrix containing more than $2^{31} - 1$ rows or columns. The LAPACK functions used by Mata cannot accept matrices larger than this.

3209. `more than 281 terarows or teracolumns`
A call was made to a function with a matrix containing more than $2^{48} - 1$ rows or columns. This is not allowed.

3250. `type mismatch`
The *eltype* of an argument does not match what is required. For instance, perhaps a real was expected and a string was received. See *eltype* in [M-6] **Glossary**.

3251. `nonnumeric found where numeric required`
An argument was expected to be real or complex and it is not.

3252. `noncomplex found where complex required`
An argument was expected to be complex and it is not.

3253. `nonreal found where real required`
An argument was expected to be real and it is not.

3254. `nonstring found where string required`
An argument was expected to be string and it is not.

3255. `real or string required`
An argument was expected to be real or string and it is not.

3256. `numeric or string required`
An argument was expected to be real, complex, or string and it is not.

3257. `nonpointer found where pointer required`
An argument was expected to be a pointer and it is not.

3258. `nonvoid found where void required`
An argument was expected to be void and it is not.

3259. `nonstruct found where struct required`
A variable that is not a structure was passed to a function that expected the variable to be a structure.

3260. `nonclass found where class required`
A variable that is not a class was passed to a function that expected the variable to be a class.

3261. `non class/struct found where class/struct required`
A variable that is not a class or a structure was passed to a function that expected the variable to be a class or a structure.

3300. `argument out of range`
The *eltype* and *orgtype* of the argument are correct, but the argument contains an invalid value, such as if you had asked for the 20th row of a 4×5 matrix. See *eltype* and *orgtype* in [M-6] **Glossary**.

3301. `subscript invalid`
The subscript is out of range (refers to a row or column that does not exist) or contains the wrong number of elements. See [M-2] **subscripts**.

3302. `invalid %fmt`
The `%fmt` for formatting data is invalid. See [M-5] **printf()** and see [U] **12.5 Formats: Controlling how data are displayed**.

3303. `invalid permutation vector`
The vector specified does not meet the requirements of a permutation vector, namely, that an n-element vector contain a permutation of the integers 1 through n. See [M-1] **permutation**.

3304. `struct nested too deeply`
Structures may contain structures that contain structures, and so on, but only to a depth of 500.

3305. `class nested too deeply`
Classes may contain classes that contain classes, and so on, but only to a depth of 500.

3351. `argument has missing values`
In general, Mata is tolerant of missing values, but there are exceptions. This function does not allow the matrix, vector, or scalar to have missing values.

3352. `singular matrix`
The matrix is singular and the requested result cannot be carried out. If singular matrices are a possibility, then you are probably using the wrong function.

3353. `matrix not positive definite`
The matrix is non–positive definite and the requested results cannot be computed. If non–positive-definite matrices are a possibility, then you are probably using the wrong function.

3360. `failure to converge`
The function that issued this message used an algorithm that the function expected would converge but did not, probably because the input matrix was extreme in some way.

3492. `resulting string too long`
A string that the function was attempting to produce became too long. Because the maximum length of strings in Mata is 2,147,483,647 characters, it is unlikely that Mata imposed the limit. Review the documentation on the function for the source of the limit that was imposed (for example, perhaps a string was being produced for use by Stata). In any case, this error does not arise because of an out-of-memory situation. It arises because some limit was imposed.

3498. (message varies)
An error specific to this function arose. The text of the message should describe the problem.

3499. _____ `not found`
The specified variable or function could not be found. For a function, it was not already loaded, it is not in the libraries, and there is no `.mo` file with its name.

3500. `invalid Stata variable name`
A variable name—which name is contained in a Mata string variable—is not appropriate for use with Stata.

3598. `Stata returned error`
You are using a Stata interface function and have asked Stata to perform a task. Stata could not or refused.

3601. `invalid file handle`
The number specified does not correspond to an open file handle; see [M-5] **fopen( )**.

3602. `invalid filename`
The filename specified is invalid.

3603. `invalid file mode`
The file mode (whether read, write, read-write, etc.) specified is invalid; see [M-5] **fopen( )**.

3610. `file from more recent version of Stata`
An attempt was made to read a file created by a newer version of Stata than you are currently using. The file is in a format that your version of Stata does not understand.

3611. `too many open files`
The maximum number of files that may be open simultaneously is 50, although your operating system may not allow that many.

3612. `file too large for 32-bit Stata`
You are running 32-bit Stata on a 64-bit computer, and the file you wish to process is larger than 4 gigabytes.

3621. `attempt to write read-only file`
The file was opened read-only and an attempt was made to write into it.

3622. `attempt to read write-only file`
The file was opened write-only and an attempt was made to read it.

3623. `attempt to seek append-only file`
The file was opened append-only and then an attempt was made to seek into the file; see [M-5] **fopen( )**.

3698. `file seek error`
An attempt was made to seek to an invalid part of the file, or the seek failed for other reasons; see [M-5] **fopen( )**.

3900. `out of memory`
Mata is out of memory; the operating system refused to supply what Mata requested. There is no Mata or Stata setting that affects this, and so nothing in Mata or Stata to reset in order to get more memory. You must take up the problem with your operating system.

3901. `macro memory in use`
This error message should not occur; please notify StataCorp if it does.

3930. `error in LAPACK routine`
The linear-algebra LAPACK routines—see [M-1] **LAPACK**—generated an error that Mata did not expect. Please notify StataCorp if you should receive this error.

3995. `unallocated function`
This error message should not occur; please notify StataCorp if it does.

3996. `built-in unallocated`
This error message should not occur; please notify StataCorp if it does.

3997. `unimplemented opcode`
This error message should not occur; please notify StataCorp if it does.

3998. `stack overflow`
Your program nested too deeply. For instance, imagine calculating the factorial of n by recursively calling yourself and then requesting the factorial of 1e+100. Functions that call themselves in an infinite loop inevitably cause this error.

3999. `system assertion false`
Something unexpected happened in the internal Mata code. Please contact Technical Services and report the problem. You do not need to exit Stata or Mata. Mata stopped doing what was leading to a problem.

Also see

[M-5] **error( )** — Issue error message

[M-2] **intro** — Language definition