Title

stata.com

mata — Mata invocation command

Syntax Description Remarks and examples Also see

Syntax

The `mata` command documented here is for use from Stata. It is how you enter Mata. You type `mata` at a Stata dot prompt, not a Mata colon prompt.

Syntax 1	Comment
`mata`	no colon following `mata`
istmt	
istmt	if an error occurs, you stay in
..	`mata` mode
istmt	
`end`	you exit when you type `end`

Syntax 1 is the best way to use Mata interactively.

Syntax 2	Comment
`mata:`	colon following `mata`
istmt	
istmt	if an error occurs, you are
..	dumped from `mata`
istmt	
`end`	otherwise, you exit when you type `end`

Syntax 2 is mostly used by programmers in ado-files.
Programmers want errors to stop everything.

Syntax 3	Comment
`mata` *istmt*	rarely used

Syntax 3 is the single-line variant of syntax 1, but it is not useful.

Syntax 4	Comment
`mata:` *istmt*	for use by programmers

Syntax 4 is the single-line variant of syntax 2, and it exists for the same reason as syntax 2: for use by programmers in ado-files.

Description

The `mata` command invokes Mata. An *istmt* is something Mata understands; *istmt* stands for interactive statement of Mata.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Introduction
The fine distinction between syntaxes 3 and 4
The fine distinction between syntaxes 1 and 2

Introduction

For interactive use, use syntax 1. Type `mata` (no colon), press *Enter*, and then use Mata freely. Type `end` to return to Stata. (When you exit from Mata back into Stata, Mata does not clear itself; so if you later type `mata`-followed-by-enter again, you will be right back where you were.)

For programming use, use syntax 2 or syntax 4. Inside a program or an ado-file, you can just call a Mata function

```
program myprog
        ...
        mata: utility("`varlist'")
        ...
end
```

and you can even include that Mata function in your ado-file

begin `myprog.ado`

```
program myprog
        ...
        mata: utility("`varlist'")
        ...
end

mata:
function utility(string scalar varlist)
{
        ...
}
end
```

end `myprog.ado`

or you could separately compile `utility()` and put it in a `.mo` file or in a Mata library.

The fine distinction between syntaxes 3 and 4

Syntaxes 3 and 4 are both single-line syntaxes. You type `mata`, perhaps a colon, and follow that with the Mata *istmt*.

The differences between the two syntaxes is whether they allow continuation lines. With a colon, no continuation line is allowed. Without a colon, you may have continuation lines.

For instance, let's consider

```
function renorm(scalar a, scalar b)
{
        ...
}
```

No matter how long the function, it is one *istmt*. Using `mata:`, if you were to try to enter that *istmt*, here is what would happen:

```
. mata: function renorm(scalar a, scalar b)
<istmt> incomplete
r(197);
```

When you got to the end of the first line and pressed *Enter*, you got an error message. Using the `mata:` command, the *istmt* must all fit on one line.

Now try the same thing using `mata` without the colon:

```
. mata function renorm(scalar a, scalar b)
> {
>         ...
> }
.
```

That worked! Single-line `mata` without the colon allows continuation lines and, on this score at least, seems better than single-line `mata` with the colon. In programming contexts, however, this feature can bite. Consider the following program fragment:

```
program example
        ...
        mata utility("`varlist'"
        replace `x' = ...
        ...
end
```

We used `mata` without the colon, and we made an error: we forgot the close parenthesis. `mata` without the colon will be looking for that close parenthesis and so will eat the next line—a line not intended for Mata. Here we will get an error message because "`replace `x' = ...`" will make no sense to Mata, but that error will be different from the one we should have gotten. In the unlikely worse case, that next line will make sense to Mata.

Ergo, programmers want to include the colon. It will make your programs easier to debug.

There is, however, a programmer's use for single-line `mata` without the colon. In our sample ado-file above when we included the routine `utility()`, we bound it in `mata:` and `end`. It would be satisfactory if instead we coded

begin `myprog.ado`

```
program myprog
        ...
        mata: utility("`varlist'")
        ...
end

mata function utility(string scalar varlist)
{
        ...
}
```

end `myprog.ado`

Using mata without the colon, we can omit the end. We admit we sometimes do that.

The fine distinction between syntaxes 1 and 2

Nothing said above about continuation lines applies to syntaxes 1 and 2. The multiline mata, with or without colon, always allows continuation lines because where the Mata session ends is clear enough: end.

The difference between the two multiline syntaxes is whether Mata tolerates errors or instead dumps you back into Stata. Interactive users appreciate tolerance. Programmers want strictness. Programmers, consider the following (using mata without the colon):

```
program example2
     ...
     mata
          result = myfunc("‘varlist’")
          st_local("n" result)           /* <- mistake here */
          result = J(0,0,"")
     end
     ...
end
```

In the above example, we omitted the comma between "n" and result. We also used multiline mata without the colon. Therefore, the incorrect line will be tolerated by Mata, which will merrily continue executing our program until the end statement, at which point Mata will return control to Stata and not tell Stata that anything went wrong! This could have serious consequences, all of which could be avoided by substituting multiline mata with the colon.

Also see

[M-3] **intro** — Commands for controlling Mata