Title

stata.com

naming — Advice on naming functions and variables

Description Syntax Remarks and examples Also see

Description

Advice is offered on how to name variables and functions.

Syntax

A *name* is 1–32 characters long, the first character of which must be

```
A–Z     a–z     _
```

and the remaining characters of which may be

```
A–Z     a–z     _     0–9
```

except that names may not be a word reserved by Mata (see [M-2] **reswords** for a list).

Examples of names include

```
x                   x2          alpha
logarithm_of_x      LogOfX
```

Case matters: `alpha`, `Alpha`, and `ALPHA` are different names.

Variables and functions have separate name spaces, which means a variable and a function can have the same name, such as `value` and `value()`, and Mata will not confuse them.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Interactive use

Use whatever names you find convenient: Mata will tell you if there is a problem.

The following sections are for programmers who want to write code that will require the minimum amount of maintenance in the future.

Naming variables

Mostly, you can name variables however you please. Variables are local to the program in which they appear, so one function can have a variable or argument named `x` and another function can have a variable or argument of the same name, and there is no confusion.

If you are writing a large system of programs that has global variables, on the other hand, we recommend that you give the global variables long names, preferably with a common prefix identifying your system. For instance,

```
multeq_no_of_equations
multeq_eq
multeq_inuse
```

This way, your variables will not become confused with variables from other systems.

Naming functions

Our first recommendation is that, for the most part, you give functions all-lowercase names: `foo()` rather than `Foo()` or `FOO()`. If you have a function with one or more capital letters in the name, and if you want to save the function's object code, you must do so in `.mlib` libraries; `.mo` files are not sufficient. `.mo` files require that filename be the same as the function name, which means `Foo.mo` for `Foo()`. Not all operating systems respect case in filenames. Even if your operating system does respect case, you will not be able to share your `.mo` files with others whose operating systems do not.

We have no strong recommendation against mixed case; we merely remind you to use `.mlib` library files if you use it.

Of greater importance is the name you choose. Mata provides many functions and more will be added over time. You will find it more convenient if you choose names that StataCorp and other users do not choose.

That means to avoid words that appear in the English-language dictionary and to avoid short names, say, those four characters or fewer. You might have guessed that `svd()` would be taken, but who would have guessed `lud()`? Or `qrd()`? Or `e()`?

Your best defense against new official functions, and other community-contributed functions, is to choose long function names.

What happens when functions have the same names

There are two kinds of official functions: built-in functions and library functions. Community-contributed functions are invariably library functions (here we draw no distinction between functions supplied in `.mo` files and those supplied in `.mlib` files).

Mata will issue an error message if you attempt to define a function with the same name as a built-in function.

Mata will let you define a new function with the same name as a library function if the library function is not currently in memory. If you store your function in a `.mo` file or a `.mlib` library, however, in the future the official Mata library function will take precedence over your function: your function will never be loaded. This feature works nicely for interactive users, but for long-term programming, you will want to avoid naming your functions after Mata functions.

A similar result is obtained if you name your function after a community-contributed function that is installed on your computer. You can do so if the community-contributed function is not currently in memory. In the future, however, one or the other function will take precedence and, no matter which, something will break.

How to determine if a function name has been taken

Use `mata which` (see [M-3] **mata which**):

```
: mata which det_of_triangular()
function det_of_triangular() not found
r(111);

: mata which det()
  det():  lmatabase
```

Also see

[M-2] **reswords** — Reserved words

[M-1] **intro** — Introduction and advice