The Stata Journal (2011)
11, Number 2, pp. 323–324

Stata tip 100: Mata and the case of the missing macros

William Gould
StataCorp
College Station, TX
wgould@stata.com

Nicholas J. Cox
Department of Geography
Durham University
Durham City, UK
n.j.cox@durham.ac.uk

People who come from Stata programming to Mata often look around and ask, "Where is the macro substitution?" The short and discouraging answer is that there is none. The longer and much more encouraging answer is that what you would do by macro substitution in Stata is easily done in other ways in Mata.

This tip presupposes some acquaintance with macros in Stata. If macros are new or not very familiar to you, [U] **18 Programming Stata** will give you a lot of background.

Let's make clear what we mean with an example. Consider a loop in Stata that runs

```
forvalues j = 1/42 {
        summarize P`j´
}
```

`forvalues` (see [P] **forvalues**) loops over integer sequences, here all of them from 1 to 42. So as Stata executes the loop, the local macro `j` takes on the values 1, 2, ..., 41, 42, which are substituted in turn within the text `P‘j’`. The effect will be that Stata sees, in turn,

```
summarize P1
summarize P2
```

and so on, up to `summarize P42`. Assuming that you do have variables `P1`–`P42`, then each will be `summarize`d in turn.

Now let us suppose that we want to do something similar in Mata. As an artificial example sufficient to show the main idea, let's read in those variables and spit out their means from Mata. However, Mata has no idea of a local or global macro. If it sometimes appears to understand such macros, that is only because references to them are interpreted by Stata before they are ever seen by Mata. But we can do the same kind of string manipulation in Mata, using its own functions.

Here is one solution:

```
for(j = 1; j <= 42; j++) {
        name_to_use = sprintf("P%g", j)
        mean(st_data(., name_to_use))
}
```

The integer `j` is mapped to its string equivalent by being inserted as one or more characters in a string that could be displayed. We are not going to display that string, but we are going to use its contents, which in the same way will be in turn `"P1"`, `"P2"`, and so on.

Here is another way to get `name_to_use` within the same loop:

```
name_to_use = "P" + strofreal(j)
```

It does not matter which command line appears in code. Either way, the idea so far is to construct the desired variable name in Mata as a string scalar, `name_to_use`. In each case, look at the help (equivalently, the Mata manual) if you want more detail on functions such as `sprintf()` or `strofreal()`.

But we can short-circuit that longer code by combining two lines of code into one, which incidentally gets us even closer to the idea of string substitution:

```
mean(st_data(., sprintf("P%g", j)))
```

or

```
mean(st_data(., "P" + strtoreal(j)))
```

The difference is just a matter of style. Some users will prefer the step-by-step solution. The choice is yours. So the main idea is simple: use Mata's display or string functions to do the manipulations you need.