Stata tip: generation of string matrices using local macros

Dmytro Vikhrov
Moody's Analytics
Prague, Czech Republic
Dmytro.Vikhrov@moodys.com

Olga Loiseau-Aslanidi
Moody's Analytics
Prague, Czech Republic
Olga.Loiseau-Aslanidi@moodys.com

Mimeo
August 12, 2016

Abstract.

Unlike Mata, Stata does not allow users to create matrices containing string values. We recommend that practitioners store each row of a string matrix in a separate local. For a given row, columns are added to the local recursively, separated by a space. All the locals created are indexed by a sequential identifier which allows users to manage and organize them in memory efficiently. We show the application of this approach using variable selection in regression analysis.

Keywords: matrix, string value, local macro, subset selection, binary code

1 Introduction

It is practical to organize workflow using structured data arrays and it is common to have string values in such arrays. For example, criteria-based variable selection in regression analysis can be formalized as follows. First, all the permutations (subsets) of the vector of variables are generated and each subset is placed in a separate row of the array. Second, a best model is found by looping over the rows of the array and comparing the values of pre-defined criteria for each model.

When coding these steps in Stata, users find that Stata does not allow string values in matrices. Harrison [2006] acknowledges this issue and suggests an algorithm that relies on the listing of the string characteristics of variables. Cox and Newton [2014] provide a detailed description of indispensable tools for dealing with lists and macros. The list is complemented by various posts in the Stata blog. However, there is no unique solution and this creates demand for algorithms that can supplement existing tools in various applications that include string matrices.

One way to overcome the issue is to use Mata. However, an alternative and probably more time-efficient solution for users not familiar with Mata is to reformulate the task into a two-step procedure that relies on local macros. In step one, a matrix of strings[1] is split into rows and each row is represented by

[1] A typical string matrix usually contains variable names, values or labels.

a local macro with an ID. In step two, each local is populated sequentially with respective column values.

2 How it works: simple example

Suppose a researcher considers two potential models; model one with variables $x_1, \ldots, x_5$ and model two with variables $y_1, \ldots, y_5$.

Table 1: Two models organized in an array containing string values.

Name	Variables				
model x	x_1	x_2	x_3	x_4	x_5
model y	y_1	y_2	y_3	y_4	y_5

Table 1 can be stored in Stata using the two-step procedure described above. In step one, we generate two local macros, namely `model_x` and `model_y`. Since there are two rows, we should create two local macros. In step two, each macro is populated with the column values from the respective rows.

```
foreach r in x y {
   forvalues c = 1(1)5 {
      local model_`r' `model_`r'' `r'`c'
      macro list _model_`r'
   }
}
```

This code produces two local macros, namely `model_x` that contains elements `"x1 x2 x3 x4 x5"` and `model_y` with `"y1 y2 y3 y4 y5"`. The macros are populated sequentially; at each iteration an additional column of Table 1 is added to the local (see the output below for `model_x`).

```
_model_x:       x1
_model_x:       x1 x2
_model_x:       x1 x2 x3
_model_x:       x1 x2 x3 x4
_model_x:       x1 x2 x3 x4 x5
```

Now, the locals can be used in a regression of a dependent variable `lhs_var` on the independent variables contained in `model_x` and `model_y` using the code below.

```
foreach r in x y {
   reg lhs_var `model_`r''
}
```

3 Example with variable labels

Special attention is required when string values contain spaces or special characters. For example, when working with variable labels, a space is no longer a valid separator of the elements within a macro (see Table 2). Hence compound double quotes should be used; see [U] **18.3.5. Double quotes** for more details.[2]

Table 2: Two models containing variable labels organized in an array.

Name	Variable Labels		
labels of model x	lab of x_1	lab of x_2	lab of x_3
labels of model y	lab of y_1	lab of y_2	lab of y_3

To check if Stata understands you, count the number of elements in the locals using the `list sizeof` function from the extended macro functions.

```
foreach r in x y {
   forvalues c = 1(1)3 {
      local lab_model_`r' `"  `lab_model_`r''  `"lab of x`c'"'  "'
   }
   local size_`r':  list sizeof lab_model_`r'
   macro list _lab_model_`r'  _size_`r'
}
```

The code above produces four lines of output:

```
_lab_model_x:    `"lab of x1"' `"lab of x2"' `"lab of x3"'

_size_x:        3

_lab_model_y:    `"lab of x1"' `"lab of x2"' `"lab of x3"'

_size_y:        3
```

The locals `lab_model_x` and `lab_model_y` can be used in labelling graphs (for example, see Cox 2011) or any other analysis performed on the labels of variables.

The advantage of using local macros for programming sting matrices is that a user no longer works with an array of strings as a whole but with each row. Thus, most functions of local macros documented in [P] **macrolists** and [P] **macro** can be applied to individual rows.

We recommend that users create macros on the extensive margin (*i.e.* more macros of relatively small size) rather than the intensive margin (*i.e.* a few macros, each containing many elements). Stata places stricter restrictions on the number of elements within a local than on the number of locals. See [R] **limits** for information on various Stata restrictions.

[2] Cox [2011] provides more examples of counting elements of different types within a local.

4 Variable selection for regression

Suppose we wish to find a model with such a combination of variables that ranks best on a pre-defined selection criterion, for example in-sample fit. For simplicity, assume that the vector of variables consists of x_1, x_2, x_3. The total number of potential models (subsets) created from this vector is 7 ($=2^3-1$) excluding an empty set (regression on a constant). These models are depicted in Table 3 below. We generate the models using binary count system as suggested by Miller [2002] on p. 23.

Table 3: All potential models from the vector of three variables.

Model ID	Regressors	Binary Representation
1	x_3	001
2	x_2	010
3	x_3 x_2	011
4	x_1	100
5	x_1 x_3	101
6	x_1 x_2	110
7	x_1 x_2 x_3	111

Binary code "001" means that in model 1 regressor x_1 and x_2 are omitted, and the model consists of only one regressor x_3. In line with this reasoning, "111" refers to a model containing all the three regressors.

When programming the subset selection in Stata, it is convenient to place each model into a local and use an incremental counter to order the models. Then, the models produced can be used in regression analysis. The code below illustrates this idea.

```
set obs 100 // <---------- Simulate data for regression
   g y = rnormal()
forvalues i = 1(1)3  {
   qui g x`i'= rnormal()
}

qui ds y, not // <--------- Create a list of locals with models
   local initial_vector `r(varlist)'

   forvalues i = 1(1)`=2^3-1' { // <--- Generate binary representation
      qui inbase 2  `i'
      local model = "`r(base)'"
      local size = length("`model'")

      forvalues j = 1(1)`size' { // <--- Match the binary to the variables
         local element = substr("`model'", `j', 1)

         if `element'== 1 {
         local model`i' `model`i'' `:  word `= `size'- `j'+ 1' of `initial_vector''
         }
```

```
    }

macro list _model`i' // <--- Show the models and estimate regressions

reg lhs_var `model`i''
}
```

When the number of potential regressors is relatively large, it makes sense to select models that have at most `p` regressors, where `p` is smaller than the vector of potential regressors. In this case the above code can be easily expanded to include another `if` loop.

5 Conclusion

In this note, we describe how constructing a matrix of strings in Stata can be easily reformulated to work with local macros. Every row of the matrix is placed in an array of indexed local macros. A researcher can work with the macros separately or as an ordered group.

An advantage of this approach is that it allows extra flexibility by applying the functionality of local macros to every row of the string matrix. We showed the application of this approach using subset selection procedure which is applicable in many areas of applied econometrics.

6 References

Cox, N. J. 2011. Stata tip 98: Counting substrings within strings. *Stata Journal* 11(2): 318–320.

Cox, N. J., and H. J. Newton, ed. 2014. *One Hundred Nineteen Stata Tips*. 3rd ed. Stata Press.

Harrison, D. A. 2006. Stata tip 34: Tabulation by listing. *Stata Journal* 6(3): 425–427.

Miller, A. J. 2002. *Subset Selection in Regression*. 2nd ed. New York: Chapman & Hall.