LOOPS, MACROS AND TEMPVARS

There are various ways to store information in Stata besides permanent variables. This cheat sheet will remind you how to keep your code clean and elegant using temp variables, macros and loops.

by Kelsey Gonzalez

1 LOCAL MACRO

A macro is an abbreviation that you can use to represent a long string or expression. Local macros are only valid until the end of the do-file or stata session.

You define a local macro either of these ways:
local macroname "string"
local macroname = expression

You can call on a macro using a grave accent and a normal accent like `this' . Try using display and in any other command:
display `macroname'

2 GLOBAL MACRO

Global macros persist across instances of stata until you delete them. Defining your own global macros is not recommended because they have the potential to cause conflicts across do-files or ado-files.

You can call on a global macro using $macroname

3 SCALARS

A scalar is usually a numeric expression and functions similarly to a local macro. However, scalars hold higher numerical precision than locals and are able to store binary data.

You define a scalar using the code:
scalar scalarname = 123
scalar scalarname = "hello"

displaying the scalar is simple, just write the scalar name:

display scalarname

4 TEMPORARY VARIABLES

A temporary variable is a variable that is only defined for the current session of Stata. You can use the tempvar in operations and expressions, but it will not appear when the dataset is saved.

The coding is quite simple. Define any number of temporary variables with the code:
tempvar var1 var2

You call on the tempvar the same way you would a local:
gen `var1' = age^2

5 TEMPORARY FILES

In addition to temporary variables, we can also store temporary datasets. This works like the preserve and restore commands, however, those can only hold 1 dataset. You can create multiple tempfiles.

Define any number of temporar variables with the code:
tempfile filename

You call on the tempfile the same way you would a local or a tempvar:
save `filename'
use `filename'

6 BASICS OF A LOOP

Loops refer to commands which execute a group of commands multiple times (Long 2009:92). A rule of thumb while programming is any time you feel like copying and pasting a chunk of code and making tiny changes, you should be using a loop instead. Loops help to avoid errors, update coding, and make debugging easier.

All loops have a basic structure that looks something like this:

```
for___ macro __ X Y Z {
        commands referring to `macro'
}
```

7 FORVALUES

Forvalues is a specific loop for patterned numeric values. Patterns can be set using

1(2)10 = one to ten in steps of two
1/5 = one to five in steps of one
1 4 to 12 = one to twelve in steps of three

The code for a forvalues macro looks like this:

```
forval macro = X(Y)Z {
    commands referring to `macro'
}
```

8 FOREACH OF

Foreach is a general loop which can contain any type of list, though foreach of is much more restrictive that foreach in. In using foreach of, you must include a list type. These include local, global, varlist, newlist, and numlist. Failure to define the correct list type will cause the loop to fail.

The code for a foreach of looks like this:

```
foreach macro of listtype X Y Z {
     commands referring to `macro'
}
```

9 FOREACH IN

Foreach in is a much more commonly used loop that can loop around any type of list, including a combination of types within one list.

The code for a foreach in looks like this:

```
foreach macro in X Y Z {
    commands referring to `macro'
}
```

REFERENCES & ADDITIONAL RESOURCES

Long, J. Scott. 2009. "The Workflow of Data Analysis using Stata". Stata Press. pp. 83-105.

http://www.ssc.wisc.edu/sscc/pubs/stata_prog1.htm

https://www.stata.com/statalist/archive/2004-01/msg00542.html

https://www.ssc.wisc.edu/sscc/pubs/stata_prog1.htm

https://stats.idre.ucla.edu/stata/modules/working-across-variables-using-foreach/