Project 4: nhanes2

McCourt School of Public Policy, Georgetown University

Overview

This week, we will focus on macros and loops in Stata. We will begin by working with local macros with no data set open. We will open the example data set, nhanes2, to demonstrate locals and loops in practice.

Week 2:

Key Ideas:

- local and global macros
- foreach loops

Key Commands / Concepts:

- help macro
- help foreach

Questions

4.11 Local Macros

- We will begin the questions this week without a data set.
- Open a new do-file in Stata and save it as this week's project.
- Last week we introduced local macros and foreach loops.
- To create a local macro, use the command `local lname lcontents`, where `lname` is the name of the local and `lcontents` is the content of the local macro.
- Consider the example below, that defines a new local macro, then displays the contents to the results window.
- Define your own local macro, with any name and contents you choose, and display the contents to the results window.
- Continue defining new locals and displaying their contents until you are comfortable with how they work.

```
local mylocalexample hello, world
display "The text of the local macro, mylocalexample, is: ||`mylocalexample'||"
```

4.12 Local Macros in Practice

- When might you use a local macro?
- One popular use is to contain a list of control variables.
- Then, you can change your list of control variables in one place, and update all of your commands.
- Consider the example below using the example data set nhanes2.
- Copy this example into your do-file. Once it is working, try updating the list of control variables and re-run the commands.
- Add a new command to `describe` the list of control variables

```
webuse nhanes2, clear
local ctrlvars age bpsystol bpdiast tcresult
display "ctrlvars: `ctrlvars'"
summarize `ctrlvars'
reg weight `ctrlvars'
```

4.13 foreach loops

- Look at the help page for `foreach`.
- There are many styles of `foreach` loop. We will begin with the first one: `foreach lname in any_list`
- A `foreach` loop repeats the commands within the loop, replacing the local macro, `lname`, with each value in `any_list`.
- See the example below, that loops through a list of animals.
- Try writing your own loop to display elements in a list. You could make a list of your favorite foods, or your favorite Stata commands.
- Each line within the loop should be indented by one `tab`.

```
foreach mylname in dog cat chicken mouse {
     display "The next animal is: --`mylname'--"
}
```

4.14 foreach loops in practice

- When might you use a foreach loop?
- Estimate the same regression for several different dependent variables.
- The example below first summarizes several variables that will be used as dependent variables.
- Then each variable is included in a regression on weight and height.
- Run this loop in your do-file.

- Add a `display` command to the loop to show the next dependent variable.

```
summ hgb-iron
foreach yvar in hgb hct tibc iron {
    regress `yvar' weight height
}
```

4.15 foreach loops, example 2

- The contents of the local macro are replaces just as if you had typed those letters into your do-file.
- The local macro can even be used as part of a word or name.
- Consider the following loop, that makes log-versions of several variables.
- Try running this loop without the `display` command. You will see very little output in the display window.
- This is why liberal use of the `display` command is very helpful in writing and testing loops.
- Write a similar loop with the `egen std()` command to make new variables that are the standardized versions of these variables.
- Prefix these new variables with `z_` instead of `ln_`.

```
foreach var in hgb hct tibc iron {
     display "Command to run: ln_`var' = ln(`var')"
     gen ln_`var' = ln(`var')
     label variable ln_`var' "Log of `var'"
}
de ln_*
sum ln_*
```

4.16 foreach loops, example 3

- Remember, any words can be "looped-through", not just variable names.
- In the following loop, the same regression is run for probit, logit, and LPM specifications.
- Add a command to this loop to generate predicted values from each specification.
- The new variables should be named `pr_heartatk_probit`, `pr_heartatk_probit`, and `pr_heartatk_regress`.

```
foreach command in probit logit regress {
     `command' heartatk weight height
}
```

4.17 foreach of varlist

- Suppose you wanted to create a loop that uses variable names.

- In the previous examples, using the `foreach/in` loop, we had to list each variable name.
- What if I want to loop over several variables, and use varlist notation to specify those variables?
- You can use the `foreach/of varlist` style loop.
  - Start with `foreach lname`
  - The third word of the command is `of` instead of `in`
  - Instead of starting the list immediately, the fourth word tells what is coming next, in this case, a `varlist`.
- Consider the example below, demonstrating both `foreach/in` and `foreach/of varlist`.
- Write a `foreach/of varlist` loop to summarize the `z_*` that you created in 4.15.

```
foreach var in ln_hgb ln_hct ln_tibc ln_iron {
     display "The next variable is: `var'"
     summarize `var'
}
de ln_*
foreach var of varlist ln_* {
     display "The next variable is: `var'"
     summarize `var'
}
```

4.18 Combining loops with outreg2

- Using loops can allow you to run many regressions quickly, but it can get difficult to keep track of results.
- `outreg2` can be useful to organize your results.
- Make sure `outreg2` is installed on your current computer by typing `ssc install outreg2`
- Consider the following code to put the regressions from 4.16 into an outreg2 table.
- Notice how the local `command` is used in two places in the loop.
- What is the purpose of the `appendreplace` local macro?

```
local appendreplace replace
foreach command in probit logit regress {
     display "Value of appendreplace: ||`appendreplace'||"
     `command' heartatk weight height
     outreg2 using "results", `appendreplace' ctitle(`command') excel
     local appendreplace append
}
```

4.19 Writing your own regression loop

- In general, when writing loops, start as simple as possible, and add new commands one-by-one.

- After each addition, verify that your loop still works, and it is doing what you want it to do.
- Use the `display` command to verify the commands that you are running.
- Write a loop to display the names of each of your `z_*` variables.
- Add a command to regress each `z_*` variable on `weight` and `height`.
- Include the `appendreplace` local from example 4.18.
- Using a `display` command, verify that it takes the value "replace" for the first iteration, and "append" for every other iteration.
- Add an outreg2 command to save all regressions in columns of the same table.
- Add an option to outreg2 to label each column with the name of the dependent variable.

4.20 Global Macros

- Global macros function just like local macros, except they don't have to be used within a single do-file.
- Once a global macro is defined, it can be used until you close that Stata window.
- This can be useful, but can also be confusing, if you have a very general macro name, like `var` or `ctrlvars`.
- When using global macros, try to give them long, unique names.
- To expand global macros, use a `$gname` in front of the name, instead of the single quote marks.
- Here is an example:

```
global myproject4dataset nhanes2
display "Data set for Project 4: $myproject4dataset"
webuse $myproject4dataset , clear
```