Fighting chaos: Tricks to re-use code and become more productive

Daiva & Luke

2015-06-29

Welcome to our Data- and Coding-related workshop

Purpose:

To teach a few tips and tricks for more efficiently managing your data, tracking your computer files, understanding appropriate analytical approaches, and speeding up the process from code to tables.

Welcome to our Data- and Coding-related workshop

Purpose:

To teach a few tips and tricks for more efficiently managing your data, tracking your computer files, understanding appropriate analytical approaches, and speeding up the process from code to tables.

Significance:

Topics we cover will help you get more comfortable with data, reduce the chance of overlooked errors, and give you more control over your work. They are also all important parts of a science movement gaining increasing attention – Reproducible Research.

Caveat: We aren't here to teach statistics

Need help with stats? Use these resources!

- U of T Statistical Consulting Services (click here)
- `http://www.stackoverflow.com`
- `http://stats.stackexchange.com`

Notes and help during this workshop

Go to this website:

`https://etherpad.mozilla.org/codeasmanuscript`

What is a macro?

- SAS has a facility to allow code to be more organized, efficient, and productive for you as the 'coder'
- Two components:
  - Macro variables
  - Macros -> A set of commands that can be re-used in different situations to make coding easier.

What is a macro variable?

- Method of organizing your code to cut down on typing, reduce errors, and make you more productive
- Basically act as 'jars' for other variables

What is a macro variable?

- Method of organizing your code to cut down on typing, reduce errors, and make you more productive
- Basically act as `jars' for other variables

Example code:

```
%let jar = BMI FatIntake Activity;
proc print data = SASdataset;
var &jar; * SAS replaces jar with 'BMI FatIntake Activity';
run;
```

4 steps to making a macro

1. Know what you want the macro to accomplish
   - Data organization
   - Statistical analysis
   - Output printing
   - Any/all of the above
2. Type the code you want to run
   - Data step
   - Proc step (proc corr, proc glm, proc contents, proc print)
3. Add macro commands and variables to Step 2
   - `%macro`
   - `%mend;` (mend = macro end)
4. Add macro arguments (basically macro variables)
5. Save your macros in a separate file

Not using a macro

Want to test association between caffeine intake and 3 different genetic variants (CYP1A2, ADORA2A, DRD2):

```
proc glm data=genes;
    class CYP1A2 sex smoke;
    model caff = CYP1A2 BMI sex smoke;
    lsmeans/ stderr;
run;

proc glm data=genes;
    class ADORA2A sex smoke;
    model caff = ADORA2A BMI sex smoke;
    lsmeans/ stderr;
run;

proc glm data=genes;
    class DRD2 sex smoke;
    model caff = DRD2 BMI sex smoke;
    lsmeans/ stderr;
run;
```

Why is that undesirable?

- High risk of making typos or errors and overlooking them
- Your SAS file can become very long
- Not ideal for sharing with others, especially once you leave your lab

Using a macro variable

Want to test association between caffeine intake and 3 genetic variants (CYP1A2, ADORA2A, DRD2):

```
%let gene = CYP1A2 ADORA2A DRD2;
proc glm data=genes;
    class &gene sex smoke;
    model caff = &gene BMI sex smoke;
    lsmeans/ stderr;
run;
```

Using a macro variable

Want to test association between caffeine intake and 3 genetic variants (CYP1A2, ADORA2A, DRD2):

```
%let gene = CYP1A2 ADORA2A DRD2;
proc glm data=genes;
    class &gene sex smoke;
    model caff = &gene BMI sex smoke;
    lsmeans/ stderr;
run;
```

But there is a problem with the above.

Using a macro

```
%macro glm (data, outcome, predictors, class=);
    proc glm data=&data ;
        class &class;
        model &outcome = &predictors &class;
        lsmeans/ stderr;
        run;
%mend glm;

%glm(data = genes, outcome = caff,
    predictors = CYP1A2);

/* Or... (for running each gene) */
%glm(genes, caff, CYP1A2);
%glm(genes, caff, ADORA2A);
%glm(genes, caff, DRD2);
```

Assignment

- Make a repetitive code without using a macro. Track this file using Git.
- Improve this code by making a macro. You can include both codes in the same .sas file. If you want you can even delete the long-style of code and do a "git diff" to see the changes to the file.
- Give us some "Thoughts" about this workshop as a separate text file. Feel free to include points on how you think you can apply macros in your research.
- Push both your SAS file and Thoughts file to the appropriate location in the Sandbox repo on GitHub.

Thanks!

- Next time: Combining macros with ODS = power!