PSYC 930: Make Friends SAS

Overall goal of the course: Get used to working with SAS syntax (instead of windows)

- Skills to learn: read in data from different formats, merge and restructure data, transform variables, arrays and macro programming for greater efficiency, how to save SAS output in different formats, how to conduct general linear model analyses

Why work with syntax instead of point-and-click windows (in any program)?

- Key idea: Save the PROCESS, not the PRODUCT → DOCUMENTATION
  - When conducting analyses – know exactly how any analysis was conducted (i.e., which options were selected, with which versions of the variables, on what dataset)
  - When modifying datasets—know exactly how new variables were created, how aberrant values were fixed, etc
  - You do not ever need to keep 8,000 versions of any dataset—keep the syntax instead
- Make a point to save a 'completed' analysis history to accompany manuscripts
- The initial front-end time investment yields HUGE long-term gains in efficiency
  - Can 'borrow' code to use across multiple projects (especially using macro variables)
  - No longer fear hearing the words "Can you do this again, but with _____?"

Working with SAS Syntax

Welcome to the wonderful world of SAS! SAS has a steeper learning curve than does SPSS, but I think you'll find climbing to the top to be well worth the effort. Most of the data management and analysis that SAS can do SPSS can also do (with some exceptions, of course), but there are several programming-related features that make SAS more pleasant to work with than SPSS.

Enhanced editor uses color-coding to make writing syntax easier:

- **Comments are GREEN**
- **Commands are BLUE**
- **Labels, titles, and libraries are PINKY PURPLE**
- **Entered data is shaded YELLOW**
- **Variables, file names, and other user-entered text is BLACK**
- **If you see RED, something is wrong**

The use of LIBRARIES and temporary directories

- Library = nickname for a physical location where permanent files are stored
- You can define and reference multiple libraries simultaneously
- By default SAS has a temporary "work" library: all files are deleted from the work library when closing SAS unless explicitly saved to a permanent, user-defined library.
- **I always recommend immediately copying files over to the work library and using that temporary copy instead!**
- That way, unnecessary (intermediate) data files are not saved as permanent files
- Also, if you mess up, your original file is still intact – this brings us to a very important difference between SAS and other programs like SPSS: **There is no 'saving the data file' in SAS: transformations happen immediately. So if you mess up, you will not know it until it is too late; thus the importance of using the data set stored in the temporary 'work' library.**
- You can always save the modified data file back into your permanent location as needed, but you don't ever really need to – as long as you keep the syntax, all transformations can be regenerated as needed, and thus you really only need the original file. The exception is when you have a HUGE file in which transformations take a long time to run. In that case it might be worthwhile to save the final product (or the subset you are working with) just to save time.
- Only SAS datasets are recognized in libraries. To refer to other kinds of data, you can use a macro variable as a placeholder for the file location via a %LET statement.

Data files are referred to EXPLICITLY

- SAS has two main types of commands: DATA steps, and PROCs
- All file transformations and variable modifications must happen inside a DATA step. Thus, you always know which file is being modified because it's the one you specify.
- If your data has already been read in, you must essentially re-define it as itself (using the DATA and SET commands) in order to do further transformations on it.
- PROCs (procedures) run things… PROC MEANS, PROC CORR, PROC REG…
- You should always explicitly specify which data file is being used for each PROC. If you do not, by default it will run it on the last one that something was done to.

SAS syntax can be used to generate plots

- SPSS will make graphs for you through windows or syntax, but all the customization (e.g., changing colors, line styles, fonts, labels) must be done through windows. This gets annoying quickly, particularly if you have lots of them to make.

- SAS syntax can be used to make any kind of plot you can think of, and you can write syntax to customize absolutely every feature of the plot. SAS plotting can require a huge learning curve, but with a few relatively basic options you can get nice-looking plots to put in papers and presentations, and you can size them and put them in whatever format directly that you need.

Advanced programming features in SAS

- Macro programming can be used to automate repetitive tasks. Whenever you find yourself doing the same thing over and over again (e.g., running the same series of models on different outcome variables, doing the same series of transformations to different datasets), these are good candidates for macro programs.
- SAS DATA steps can include arrays and loops to automate repetitive variable transformation tasks.
- Another nice feature in SAS is the Output Delivery System (ODS). You can save SAS output as datasets, html, rich text, or pdf files.
- The combination of macro programs + ODS is particularly powerful, because it can enable you to generate many, many analyses, save the tables of output into datasets, manipulate and combine the datasets, and then export them into something easy to read.

The ability to write syntax to make tables and figures is particularly helpful when revising manuscripts, theses, and dissertations…. Change something? One click and all of your data manipulation, analyses, and tables and pictures of results are updated!

Tips for good, readable, and reusable syntax:

- Use EXCESSIVE comments/documentation throughout (you'll thank yourself later)
- Start with who wrote it, for what purpose, when (and when last updated)
- Record which data files get used (use macro variables for file references – stay tuned)
- Add variable labels immediately upon creating new variables
- Separate logical sections with blank lines, comment lines, etc
- Use indentation to help delineate structure (not required, but easier to read)
- Use capital letters for command words (not required, but easier to read)
- Test/allow for all contingencies (and physically LOOK at the data after each step)
- Comments can be used to "shut off" parts of code and yet keep it all in the same file
- Avoid "hard coding" wherever possible by using macro variables (stay tuned)

General tips for de-bugging syntax:

- All commands and variable names spelled correctly?
- Refer to correct version of file?
- Do the variables you are referring to exist yet (in the file you are working with)?
- Comments shut off correctly?
- Quotes balanced?
- Parentheses balanced?
- Do you have a command terminator (semi-colon in SAS)?
- Are the colors ok? Remember: red = wrong
- Check the log – just because it runs doesn't mean it's right
- Is your SAS data set open? It must be closed to use it in most cases.
- In SAS – Did nothing happen? Does it say that something is still "running" at the top of the screen – if so, you are missing a "run;"