**Welcome to SAS**

**Overall goals of the course:**

* Get used to working with SAS syntax for data management: read in data from different formats, merge and restructure data, transform variables, generate descriptive statistics

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

* Key idea: Save the PROCESS, not the PRODUCT 🡪 DOCUMENTATION
* When running 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 17 versions of any dataset – keep the syntax instead
* The initial front-end time investment yields HUGE long-term gains in efficiency
* Can ‘borrow’ code to use across multiple projects
* No longer fear hearing the words “Can you do this again, but with \_\_\_\_\_\_\_?”

**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 possible contingencies (and actually 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)

**Notable features of SAS:**

* Enhanced editor uses color-coding to make writing syntax easier:
* **Comments are GREEN**
* **Commands are BLUE**
* **Labels, titles, and libraries are PURPLE**
* **Anything in PURPLE is CASE-SENSITIVE and SPACE-SENSITIVE**
* **Entered data is shaded YELLOW**
* **Variables, file names, and other user-defined 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.
* Data files are referred to EXPLICITLY
* SAS has two main types of commands: DATA steps, and PROCs (procedures)
* 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 run analyses… 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 use the last file that something was done to, which may not be what you intended (especially if the order of the code gets re-arranged).
* Advanced programming features in SAS
* SAS has excellent built-in functions for working with string and date variables.
* SAS DATA steps can include arrays and loops to automate repetitive transformations.
* Macro programming can be also 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.
* Another nice feature in SAS is the Output Delivery System (ODS). You can save SAS output as SAS datasets, html, rich text, or pdf files.
* The combination of macro programs + ODS is powerful, because it can enable you to generate many, many analyses, save and organize the tables of output into datasets, and then export the results into something easy to read.
* 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.
* The ability to write syntax to make tables and figures is particularly helpful when revising manuscripts or other products…. Change something? One click and all of your data manipulation, analyses, and tables and figures of results are updated!

**About creating and modifying variables:**

* Variable ‘types’:
* 2 main types you’ll use: numeric or string (text)
* **Store data as numeric whenever possible** – string variables are a pain to work with (and they are case-sensitive and space-sensitive, too)
* Do not mix strings and numbers in one variable
* Variable NAMES, variable LABELS, and variable VALUES:
* Variable NAMES are short – **try to stay under 8 characters** (even though you are allowed up to 32) for two reasons: for use in other programs with 8-character restrictions, and so you have room to add characters after transforming the variable
* Variable NAMES are not case-sensitive (but will display that way)
* Variable NAMES must start with a letter or underscore, numbers are ok, but try to avoid periods or special characters (incompatible in many other programs), avoid ending in underscores; no spaces allowed
* Variable LABELS are long (up to 256 characters) – make them as long as you need to document exactly what the variable means; I recommend including the name, label, and value information ALL in the label for convenience – for example:   
   LABEL dead= “Dead: Whether subject is dead as of 2004, 0=no, 1=yes”
* Variable VALUES are not the same as in SPSS – they do not get stored with the data, but you can use PROC FORMAT to provide values for displays in output
* Missing values codes:
* Missing value codes are usually unnecessary – if it’s missing, leave it blank
* If you must use missing value codes, remember to define them ahead of time (otherwise SAS won’t recognize them), **pick numbers that will never show up** in the data (i.e., pick -99 instead of 99), and use the same codes across all your variables
* Better strategy: If you need to keep track of different reasons for missingness (e.g., not there that day = -99, wouldn’t answer the question = -98), it is better to create a **separate** categorical variable to keep track of missingness reasons

**General tips for de-bugging SAS syntax:**

* Are the colors ok (and remember, red=wrong)? If not, check:
* Are the comments shut off correctly?
* Are the quotes balanced?
* Are the parentheses balanced?
* All SAS commands end in a semi-colon – are you missing any?
* Are all commands and variable names spelled correctly?
* Are you referring to correct, current version of the file? In the right library?
* Do the variables you are referring to exist yet (in the file you are working with)?
* Have you defined all the macro variables that are being used?
* Is your data set open? It must be closed to use or modify it in most cases.
* Did nothing happen? Does it say that something is still “running” at the top of the screen – if so, you are missing a “run;”
* Check the log – just because it runs doesn’t mean it’s right – also check the data set to see if the transformations occurred as you had intended