

From Researcher to Programmer: 5 SAS® Tips I Wished I Knew Then

Crystal Carel, MPH^{1,2}

¹STEEEP Analytics, Baylor Scott & White Health, Dallas, TX

²Northern Illinois University, College of Health & Human Sciences

ABSTRACT

Having crossed the spectrum from an epidemiologist & researcher (where ad hoc is life & focusing on research) to a SAS® programmer (writing reusable code for automation, meaning no manual interventions), I have learned a few things that I wished I knew then as a researcher to not only be a better SAS® programmer but also to save time & effort as a researcher by having well organized, accurate code (that I didn't accidentally remove) & code that will work if ran again on another date.

This paper will present 5 SAS® tips that are common practice among SAS® programmers. I will provide researchers who use SAS® with tips that are handy, useful, & provide code (where applicable) to be able to try out at home. Using the tips provided will make any SAS® programmer smile when they get presented with your code (not guaranteed, but your results should not vary by using these tips).

TIP 1: CLEAN UP & ORGANIZE CODE

This tip may sound simplistic, silly, and a no brainer. However, this is the piece that never gets the respect it's due. Cleaning up and organizing code is crucial to save time in the long run, save sanity of the people that inherit your code, and makes your code make sense when you've slept for a while. Step back and make sure you are making sense of your code.

You had to learn how to write, you had to learn how to read, now you must do the exact same in coding.

WHEN WRITING CODE:

- Pick a style and stick with it. UPPER CASE, lower case, or Mixture Of The Two (PROPCASE).
- There isn't a clear cut, one size fits all.
 - Since SAS® is not case specific with functions you can pick your flavor and go with it.
- Code in order!
 - Don't forget to organize your code from start to end after you have developed code!

WHEN READING CODE:

- Insert a `/*COMMENT BOX*/` at the beginning of each data step to explain the step and what is happening.
- `/*COMMENT BOX*/` helps when reviewing code, understanding logic, and saves countless hours without having to dissect your code.
 - If you later comment out code and want to save that section, put a date and removal reason in your code.
 - This way you still have the code and you will remember why you commented it out.

EXAMPLES WRITING & READING CODE:

UPPER CASE & COMMENT:

```
/*WHERE VAR=1*/  
DATA WORK.UPPER;  
SET WORK.OLD;  
WHERE VARIABLE = 1;  
RUN;
```

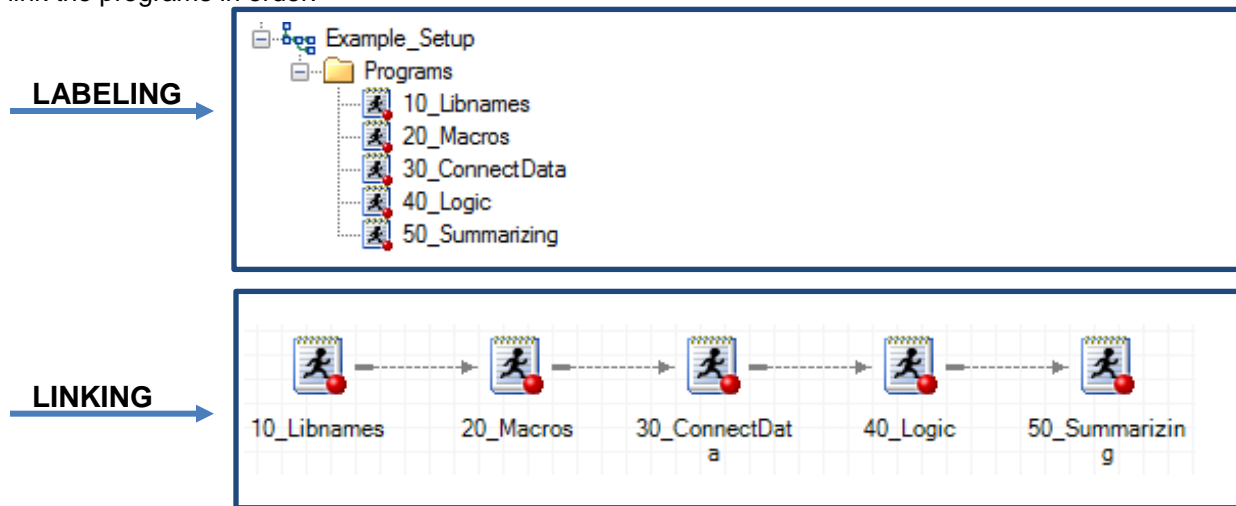
LOWER CASE & COMMENT:

```
/*WHERE VARIABLE IS >1*/  
data work.lower;  
set work.OLD;  
if variable ge 1;  
run;
```

MIXTURES OF CASES & COMMENT:

```
/*CREATE NEW FROM OLD*/  
Data Work.New;  
Set Work.OLD;  
Run;
```

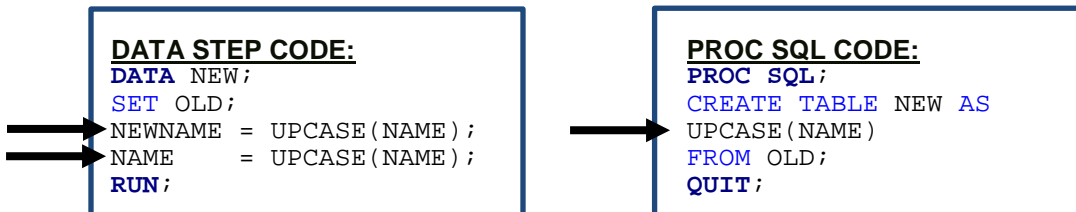
For SAS® EG users: Create programs for each type of data piece, add sequential numbers, and label programs according. Finally, link the programs in order.



- Right click on the program and select “Link Program to...”

TIP 2: UPPERCASE – TO SAVE YOUR EYES AND PROGRAM

- For character variables (such as name, address, or variables that you will use to search for certain ‘strings’), you can almost guarantee that your data has an upper case where you expect lower case and vice versa.
- Convert character variables by using UPCASE(variable-name).
 - Example: John J. Doe, JOHN J. Doe, and JOHN J. DOE will always be JOHN J. DOE when using UPCASE.



TIP 3: SORT ONCE FOR GOOD LUCK – SORT TWICE FOR GOOD MEASURE

- When sorting data, remember NODUP & NODUPKEY can produce very different results.
 - SAS HELP says to ‘consider to use NODUPKEY option instead of NODUP’ however be cautious.
- Whichever way, this is an area where it’s easy to ‘lose’ data you didn’t want to lose.
- Sort once if you are using NODUP and want to keep all the unique rows of data.

```
PROC SORT DATA=OLD
OUT=NEW NODUP;
BY VAR1;
RUN;
```

- Sort twice if using NODUPKEY and need to sort, but want no duplicates based on BY groupings.
 - You may need to sort on one variable, but do not want duplicates based on other variables.

```
/*SORT ON VAR1*/
PROC SORT DATA=OLD
OUT=MIDDLE;
BY VAR1;
RUN;

/*ELIMINATE DUPS BASED ON VAR2 AND VAR3*/
PROC SORT DATA=MIDDLE
OUT=NEW NODUPKEY;
BY VAR2 VAR3;
RUN;
```

TIP 4: MACROS – NOT AS SCARY AS IT SOUNDS

- **Macro variables** can be created for any variable that is consistently change such as a variable name, date, or even a file name by a %LET statement.
- At the beginning of your code, try a %LET statement for items that are always changing.
 - This way it'll be up front and you won't miss it.

```
SAS CODE:
%LET NAME_OF_MACRO_VARIABLE = VARIABLE;
%PUT R = &NAME_OF_MACRO_VARIABLE;

SAS LOG:
R = VARIABLE
```

- If the same code is repetitious and only a few pieces change each time, then a **macro** would work perfectly for that section.
 - This can be a dataset, variable, date, etc. that you change each time but the structure is consistent.

```
%MACRO REPEAT(OLD_DATASET=, NEW_DATASET=);
PROC SORT DATA= &OLD_DATASET
            OUT= &NEW_DATASET;
BY &VAR;
RUN;
%MEND REPEAT;
%REPEAT (OLD_DATASET=DATA_A, NEW_DATASET=DATA_B, VAR=NAME);
%REPEAT (OLD_DATASET=DATA_A, NEW_DATASET=DATA_C, VAR=LOCATION);
```

- We made 2 new data sets:
 - Using the %MACRO/%MEND – this sets up the start/end of the macro.
 - REPEAT is the name of the macro.
 - OLD_DATASET is name of the old data set. NEW_DATASET is the name of the new dataset we are creating.
 - DATA_B is sorted on NAME. DATA_C is sorted on LOCATION.
 - The %REPEAT is calling the macro that was set up.

TIP 5: DATES – DATES - DATES

- This was the area that was the hardest for me as a researcher to programmer to figure out.
 - Dates can make or break a report. They may as well be considered the **bane** of all coding.
- If your code has constantly changing dates, then setting up a macro variable for your date can help with either data steps or reporting.
- Dates are easier to understand when you have code to try out & compare the differences
 - **Copy/paste/run the code below on a couple different days & see the results.**
- First figure out what today's date is:

```
DATA EXAMPLE;
FORMAT CALDATE DATE9.;
CALDATE = TODAY ();
RUN;
```

- Next manipulate today's date to see various formats & future dates based from today:

```
PROC SQL NOPRINT;
SELECT INTNX("DAY", CALDATE, 0,"E") FORMAT = DATE9. AS DATEA
,INTNX("MONTH", CALDATE, 0,"B") FORMAT = MMDDYY10. AS DATEB
,(CALCULATED DATEA) FORMAT = YEAR4. AS DATEC
,INTNX("YEAR", CALDATE, 7,"B") FORMAT = DATE9. AS DATED
,INTNX("MONTH", CALDATE, -2,"B") FORMAT = YYMMN6. AS DATEE
INTO :DATEA,
:DATEB,
:DATEC,
:DATED,
:DATEE
FROM WORK.EXAMPLE;
QUIT;
```

To see what dates you have made, use the %PUT statement and check the SAS LOG:

```
SAS CODE:
%PUT DATEA= &DATEA;

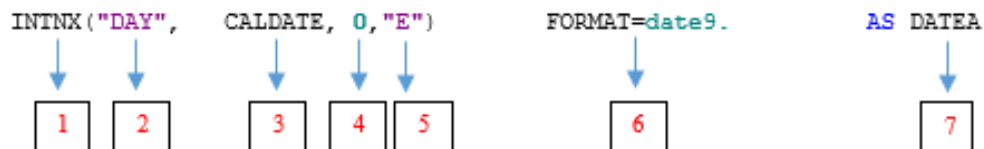
SAS LOG:
DATEA = 28AUG2017; ←The date of today.
```

- You can manipulate this off any date/time period – be careful during that Dec. 31st runs the same as Jan. 1st.
 - Try out a lot of scenarios and always double check your SAS LOG!
- Now you are ready to use the macro date you created in your code!
- Using your code, insert the date macro variable you just created:

```
/*CREATE DATASET WHERE DATE IS EQUAL TO DATEA*/
DATA WORK.NEW;
SET WORK.OLD;
WHERE DATE = "&DATEA"D;
RUN;
```

- To make the date macro variable work, insert an '&' before the name of the macro.
- Wrap the macro variable in double quotes & add a 'D' after the double quotes.
- Your dataset will only have data associated where date was equal to the date macro variable.

To better explain what the calculations are doing – here is an explanation step by step:



1. INTNX function being used which increments dates/times.
2. Interval Time Period = day/week/month/year – always in “quotes” (single or double).
3. Variable that the calculation will use as a ‘base’ date.
4. Number can be positive (+), negative (-), or 0.
 - Positive means future, negative means prior, 0 means current.
5. Internal Time Period calculation will be made from: E=End B=Begin, S=Same, M=Middle, etc.
6. FORMAT = this is the format to be displayed. There is a whole host of options.
7. New Variable Name being created.

CONCLUSION

These tips have popped up along my research to programming career and I still abide to them. By practicing these tips, you will be more equipped to hand off your code for automation. These tips will also help yourself remember what you did later, why you did it, and be able to follow the process from beginning to end. You should feel more confident about your product and have a more packaged feel based on these tips.

REFERENCES

1. Carel, Crystal G. "From Researcher to Programmer: 5 SAS® Tips I Wished I Knew Then". Proceedings of the SAS Global Forum Conference – 2017. Orlando, FL.
2. Levin, Lois. "SAS® Programming Guidelines." Proceedings of the SAS® User Group International Conference – 2006. San Francisco, CA. Available at: <http://www2.sas.com/proceedings/sugi31/123-31.pdf>
3. Sayles, Harlen. "Getting and Staying Organized: Tips for Improving the SAS® Data Analysis/Analyst Experience." Proceedings of the Midwest SAS® Users Group Conference – 2015. Omaha, NE. Available at: <http://www.mwsug.org/proceedings/2015/SA/MWSUG-2015-SA-07.pdf>
4. SAS® Institute Inc., SAS® 9.3 Functions and CALL Routines: Reference. Cary, NC: SAS® Institute Inc. Accessed January 13, 2017. Available at: <http://support.sas.com/documentation/cdl/en/lefunctionsref/63354/HTML/default/viewer.htm#p0ilulfezd14ykn17295t8tnh4xc.htm>
5. Kelsey, Britta. "The Mystery of the PROC SORT Options NODUPRECS and NODUPKEY Revealed." Proceedings of the SAS® User Group International Conference – 2005. Philadelphia, PA. Available at: <http://www2.sas.com/proceedings/sugi30/037-30.pdf>
6. Delwiche, Lora & Slaughter, Susan. "SAS® Macro Programming for Beginners," Proceedings of the SAS® User Group International Conference – 2004. Montréal, Canada. Available at: <http://www2.sas.com/proceedings/sugi29/243-29.pdf>
7. Li, Arthur (2010). "Get a Grip on Macros in just 50 Minutes!" Proceedings of the Western Users of SAS® Software Conference – 2010. San Diego, CA. Available at: http://www.lexjansen.com/wuss/2010/Applications/2956_7_APP-Li.pdf
8. Carel, Crystal. "Let Dates Drive your Data," Proceedings of the South Central SAS® Users Group Conference – 2015. Baton Rouge, LA. Available at: http://www.scsug.org/wp-content/uploads/2015/11/Carel-Let_Dates_Drive_Your_Data.pdf
9. SAS® Institute Inc., SAS®/ETS 9.2 User's Guide Online, "Interval Functions INTNX and INTCK". Cary, NC: SAS® Institute Inc. Accessed February 21, 2017. Available at: http://support.sas.com/documentation/cdl/en/etsug/60372/HTML/default/viewer.htm#etsug_tsdata_sect038.htm

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Crystal Carel
Baylor Scott & White Health
214-265-3674
Crystal.Carel@BSWHealth.org

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

The authors of the paper/presentation have prepared these works in the scope of their employment with Baylor Scott & White Health. SCSUG Inc is granted a non-exclusive right in the copyright of the work to the SCSUG Inc to publish the work in the Publication, in all media, effective if and when the work is accepted for publication by SCSUG Inc.