Making your SAS code more readable

12

By Ron Cody on SAS Users | November 6, 2019

Topics | Learn SAS Programming Tips

I have been programming SAS for a LONG time and have never seen much in the way of programming standards. For example, most SAS programmers indent DATA and PROC statements (I like three spaces). Most programmers do not like to see more than one statement on a line and most agree that there should be blank lines between program boundaries (DATA and PROC steps).

I thought I would share some of my thoughts on programming standards, with the hope that others will chime in with their ideas.

• I am not a fan of camel-case. For example, I prefer Weight_Kg to WeightKg. The reason that some programmers like camel-case is that SAS will automatically split a variable name at a capital letter in some headings.

• I like my TITLE statements in open code, not inside a PROC. To me, that makes sense because TITLE statements are global.

• There should be no conversion messages (character to numeric or numeric to character) in the SAS log. For example use Num = INPUT(Char_Num,12.); instead of Num = 1*Char_Num;. The latter statement forces an automatic character to numeric conversion and places a message in the log.

• I always use the statement ODS NOPROCTITLE;. This eliminates the default SAS procedure name at the top of the output.

• Although fewer and fewer people are reading raw text data, I like my @ signs to all line up in my INPUT statement.

• I like to use the /* and */ comments to define all macro variables. For example:

```
%macro Example (Dsn=,     /* Data set name */
                Varname=, /* Variable name for temperature */
                Convert=, /* C for Celsius to Fahrenheit
                             F for Fahenreight to Celsius  */);
   data &Dsn;
      set &dsn;
      if &Convert = 'C' then &Varname = (1.8*&Varname + 32);
      else if &Convert = 'F' then &Varname = (&Varname - 32)/1.8;
   run;
%mend;
```

Notice that I prefer named parameters in my macros, instead of positional parameters.

If this seems like too much work - SAS Studio has an automatic formatting tool that can help standardize your programs. For example, look at the code below:

Really ugly, right? Here is how you can use the automatic formatting tool in SAS Studio.

When you click this icon, the program now looks like this:

That’s pretty much the way I would write it. By the way, if you don't like how Studio formatted your code, enter a control-z to undo it.

For more tips on writing code and how to get started in SAS Studio – check out my book, Learning SAS by Example: A Programmer’s Guide, Second Edition. You can also download a free book excerpt. To also learn more about SAS Press, check out the up-and-coming titles, and receive exclusive discounts make sure to subscribe to the SAS Books newsletter.

Tags

sas studio | sas programming | SAS Programmers | sas author | SAS 9.4

Share

Ron Cody

Private Consultant

Dr. Ron Cody was a Professor of Biostatistics at the Rutgers Robert Wood Johnson Medical School in New Jersey for 26 years. During his tenure at the medical school, he taught biostatistics to medical students as well as students in the Rutgers School of Public Health. While on the faculty, he authored or co-authored over a hundred papers in scientific journals. His first book, *Applied Statistics and the SAS Programming Language*, was first published by Prentice Hall in 1985 and is now in its fifth edition. Since then, he has published over a dozen books on SAS programming and statistical analysis using SAS. His latest book, *A Gentle Introduction to Statistics Using SAS Studio* was published this year. Ron has presented numerous papers at SAS Global forums, regional conferences, as well as local user groups. He is presently a contract instructor for SAS Institute and continues to write books on SAS and statistical topics.

12 COMMENTS

Bob Peplinski on December 19, 2019 9:27 pm

I like to put a comment right after my procs and data steps to tell me what that does when I collapse all the code blocks. Try to keep it short so it fits on the same line. This way I can collapse all code blocks and the program doesn't feel as overwhelming. EG user and mainly find this helpful where I can collapse the blocks.

+ proc sql; /* this section creates this table*/
+ data want; /* this section does this */

Thanks for the blog and comments everyone. Got a lot out of this.

Reply ›

idea. Thanks.

Reply >

Dale Thompson on November 21, 2019 7:27 pm

First, appreciate this article, the importance of creating "readable" SAS programs and the comments submitted on this topic so far. As a long time programmer/developer in SAS and many other programming languages as I'm sure many others are as well, there are some fundamentals for writing good programs that should always be adhere too regardless of if you're writing in SAS, Java, Javascript, etc., including:
1. Assume you're not always going to be the person having to run, maintain, etc. the program, so write it as such that someone else can easily understand it.
2. Comments are a good thing, especially if you're doing some type of complex calculation, operation, do-loop, etc. This has saved me more than a few times when I developed some very clever code and then had to modify it a couple of years later.
3. White space, indenting, etc. really helps the general "readability" of code.
4. This more of a Program Editor note, but incredibly help...use the color coding functionality of the SAS Editor to highlight different functions, data types, etc. I have the background color change for Comments and String variables and it's amazing how easy it is to notice you're missing a tick mark, semi-colon or end of comment (*/) mark when your whole program changes color ☺
5. Make variable names meaningful...yes, it's a few extra keystrokes, but the ability to quickly and easily understand the meaning of a variable is invaluable. (e.g. "X" vs. "LastName")

Reply >

Ron Cody on November 25, 2019 11:12 am

Hi. Thanks for your comments. Adding comment statements has helped me understand code I wrote years ago. There is a "rule" that after a period of time (6 weeks?) that reading your own code is equivalent to reading someone else's code. It's a shame that so many of the posts showing good programming practices are re-formatted with proportional

Keep the comments coming.

Reply >

Jay Carbonell on November 21, 2019 3:22 pm

I would change the code as this for debugging and readability:

```
if condition then
do;
    var1=vara + varb;
    var2= var1 * varc;
    if var2 &gt; chkpt then
    do;
        var3= var2 / 2;
        var4= var3 + vard;
    end;
end;
```

Reply >

Susan on November 21, 2019 11:24 am

Indenting and aligning can be a great way not only to improve readability but also to facilitate debugging. For instance, I like to indent statements within a do loop and line up my END statement with the corresponding DO as shown below. This not only makes the conditional code easily identifiable, it also helps when you have nested DO statements and are trying to keep track of where each ends (also useful for tracking down missing END statements).

```
if condition then do;
                    var1=vara + varb;
```

```
                                        var4= var3 + vard;
                                    end;
                end;
```

Reply ›

Eric on November 8, 2019 3:36 am

From SQL programmers I have learned to vertically align the commas. Like:

* use fixed width font;

```
proc sql;
   create table aggregate as
      select region
           , month
           , sum(sales) as sales
        from oursystem
       where country = 'France'
    group by 1, 2;
quit;
```

And I use it also for some macros aligning the = symbols:

```
%means2dset(data  = oursystem
          , where = (country = 'France')
          , class = region month
          , var   = sales
          , stat  = sum
          , out   = aggregate);
```

Reply >

Hawk on November 6, 2019 9:49 am

I copied this from one of my favorite SAS programmers that always insisted on "neat looking" SAS code. I always put this on top of each of my codes so another programmer can easily learn the "purpose" of the program.

```
*------------------------------------------------------------------------------------------*;
* SAS code: [file address]\example.sas *;
* Input(s): [file address]\input_example.xlsx *;
* Output(s): [file address]\output_example.pdf
* Objective: Why are we writing this SAS program? *;
* Date: *;
*------------------------------------------------------------------------------------------*;
```

Reply >

Chris Jones on November 6, 2019 2:20 pm

For no real reason, I don't like that style of comment, preferring /* comment */ style.
I've also seen issues when running in batch on Unix if the * comment ; style contains a single apostrophe.

Reply >

Ron Cody on November 11, 2019 3:31 pm

Hi Chris.

You may get two replies from me because I tried earlier and didn't see it.

leave a comment or email me directly.

Best,
Ron

Reply >

Robert Pearson on November 26, 2019 11:18 am

I try to avoid /* */ comments in the "heart" of my programs because if they're included then you lose the ability to easily comment out a step or block of code. I only use that comment type in my header.

Thanks for this post though, this is great. As much as I keep urging my students, I just can't get them to take program formatting seriously

Bob

P.S. since I'm replying to you, I have to mention that only this week I caught for the first time your little INT/Ent quip in your Learning SAS book. As a lover of Tolkien and an admitted purveyor of rimshot jokes, I have to say nicely done.

Reply >

Laurie Fleming on December 10, 2019 7:13 pm

I tend to agree, but if I want to comment out a block of code which contains /* … */, I put %macro donotexecute ; … %mend donotexecute; around it.

Reply >

Your Comment

Your Name

Your Email

Your Website

Save my name, email, and website in this browser for the next time I comment.

Post Comment

This site uses Akismet to reduce spam. Learn how your comment data is processed.

BLOGS

Contact Us

Follow Us