

Just Skip It

Helen Carey, Carey Consulting, Kāneʻohe, HI

The illiterate of the 21st century will not be those who cannot read and write, but those who cannot learn, unlearn, and relearn. —Alvin Toffler

Abstract

When developing SAS® programs, we sometimes (or often) want to either submit only a portion of the code or skip some of the code. This paper explains how to do both and offers tips and techniques to make the coding easier to do and easier to identify.

Why Skip Code?

At times you want to run or test some statements and skip others. Some reasons why you may want to skip the code are:

- data set has already been created
- data has been checked or explored
- portion of code has been debugged
- code is no longer needed, but you want to leave it in program

Submit Portion of Code

Let's look at submitting a portion code in the windowing environment.

First select the code that you want to submit. As a reminder, here are three ways to mark the text. (1) You can click the right mouse button and then drag the mouse over the code to be selected and release the button. (2) Another way is to single click the right mouse button at the start (or end) of the code; move the mouse to the other end where you hold the shift key down and click. (3) The final way is to click and hold down the left mouse button in the margin on the first line of text that you want to select. Still holding down the left mouse button, drag the mouse pointer within the margin to the last line that you want to select. Release the left mouse button.

Next, submit the select by either clicking on the run button , pressing the F3 key or right mouse clicking while on the marked text and selecting *Submit Selection*.

Often, when developing a portion of my program, I will cut that portion to a new window and keep changing and submitting that portion of the code. When finished, I copy it back to the original program. This save time by being able to submit the entire window and not have to first select the text.

Skipping Code

If you want to skip over portions of code, these are ways to do it:

- Run cancel statement (RAN CANCEL;)
- Delimited comments (/* */)
- Comment out the %INCLUDE statement (*%INCLUDE 'filename';)
- Macro Skip (%MACRO skip; ... %MEND)
- Define macro variable to be a comment (%LET debug = *;)

Skipping Code with Run Cancel Statement

Including the option CANCEL on the run statement terminates the current step without executing it. SAS prints a message that indicates that the step was not executed.

Example:

```
proc freq data=school;  
  table gender * ethnicity;  
run cancel;
```

```
7
8   proc freq data=school;
9     table gender * ethnicity;
10  run cancel;

WARNING: The procedure was not executed at the user's request.
NOTE: PROCEDURE FREQ used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds
```

Caution. The Cancel option has no effect where the data step has the datalines statement or the run statement has no effect, such as with PROC SQL. Also it has not effect when you use the KILL option with PROC DATASETS.

```
data;
input x y;
datalines;
  1 2
  3 4
run cancel; /* does not work with datalines; */

proc sql;
select * from data1;
run cancel; /* does not work where run has no effect */
```

Because of these exceptions and the easy to identify skipped code with other techniques, I seldom use this way to skip code.

Skipping Code with Delimited Comments (/* */)

One way that is frequently used to skip code is to place /* and */ around the code to make it a comment or, in other words, comment out the code.

This is the code that you want to skip.

```
proc freq data=school;
  table gender * ethnicity;
run;
```

When in the Enhanced Program Editor window, the easier way to add the delimited comments is to mark the text, then press cntrl+/ keys to place /* and */ around each line.

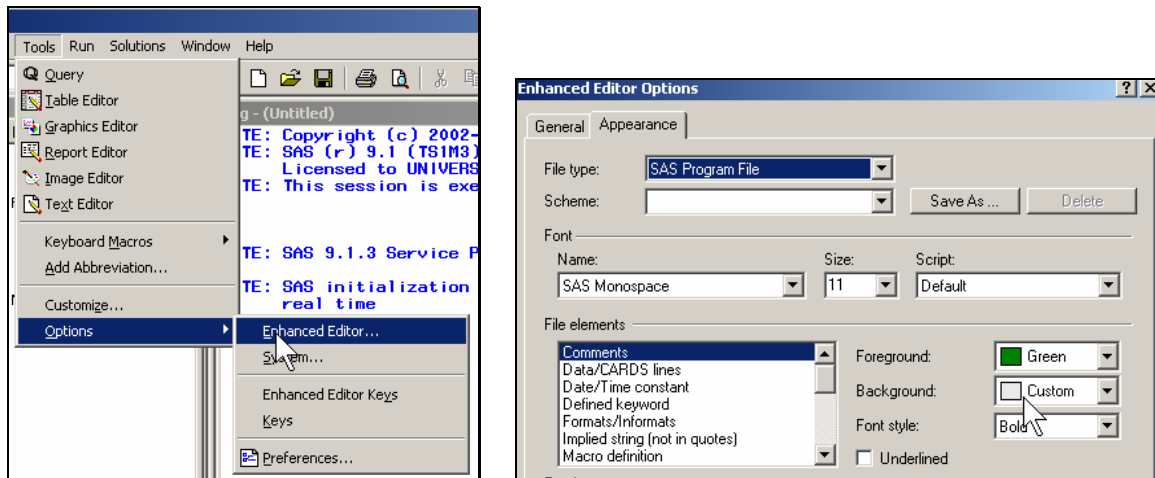
```
/*proc freq data=school;*/
/*  table gender * ethnicity;*/
/*run;*/
```

Check the color coding of the words to see if the code is commented out.

To easily remove the delimited comments, mark the text again and press the shift+cntrl+/ keys.

Enhanced Editor Feature. If you have printed this with a color printer or are viewing it on the screen, you will notice in the above example that the background of the commented code is light yellow. This makes it easier to find the comments in the enhanced editor.

To add the light yellow background to the comment statements, go to the tools bar, select Options, then Enhanced Editor.



In the *Enhanced Editor Options* window, click on the *Appearance* tab. Under *File elements*, choose *Comments*. On the right-side of the dialog box, choose a color for the background. Choose a light color, such as a custom color of light yellow. Otherwise, the default colors are very intense.

Skipping Code By Commenting Out %INCLUDE Statement

If you use %INCLUDE statements, you skip executing the file by commenting out the statement. Just put an asterisk in front to make it a comment statement (*.....).

Let's first review the %INCLUDE statement. The %INCLUDE statement points to a file. When the statement is executed, the indicated file (be it a complete program, macro definition, or a statement fragment) is inserted into the calling program at the location of the call. You can break your larger program into modules or program segment stored in separate files. Then use %INCLUDE statements to execute those files.

You can identify the file to be included directly from within the %INCLUDE statement, such as:

```
%include 'c:\DOE\programs\checkdata.sas';
```

Also it can be done indirectly through the use of a fileref.

```
filename in 'c:\DOE\programs\checkdata.sas';
.....
%include in;
.....
```

In the SAS program editor window, you can open the included file to change or review it by marking the file name on the %INCLUDE or FILENAME statement and pasting the name in the filename area of the open dialog box. (I often use the Ultraedit text editor program (www.ultraedit.com) for editing. One reason is that with that editor, you can easily click on a filename within the text and open the file in another window.)

A disadvantage of using %INCLUDE in a large application is finding and maintaining the filerefs that point to the individual files. One way to make it easier is to place the files that are to be 'included' in one central location. This way they will be easier to find and maintain. On the PC, the central location is a directory. For example:

```
filename project 'c:\DOE\programs';
.....
%include project(checkdata);
```

This makes it easier to move the SAS files to a new directory or drive and just change the location in the filename statement. However, there is not a complete filename to cut and paste in the open dialog box. You could include the complete filename in a nearby comment. Another way to find and open the files quickly in the program editor window is to use the My Favorite Folders window. Add the directory to Favorite Folders by right mouse clicking on Favorite Folders in the tree listing and selecting *New Favorite Folder*.

Creating file to include:

```
/* checking data */  
proc freq data=school;  
    table gender * ethnicity;  
run;
```

Save to file c:\DOE\programs\checkdata.sas

One way to include the file created above:

```
data school;  
    set allschools (where code=12);  
run;  
  
%include 'c:\DOE\programs\checkdata.sas';  
  
proc print data=school;  
run;
```

Another way to include the file is to use the filename pointing to a directory:

```
filename project 'c:\DOE\programs';  
data school;  
    set allschools (where code=12);  
run;  
  
%include project(checkdata);  
  
proc print data=school;  
run;
```

Skip executing the file by commenting out the %INCLUDE statement:

```
data school;  
    set allschools (where code=12);  
run;  
  
*%include 'c:\DOE\programs\checkdata.sas';  
  
proc print data=school;  
run;
```

Skipping Code with SKIP Macros

You can use a macro to bracket the code to be skipped and thus keep it from being compiled and executed.

First, let me explain why the SKIP macro is needed. Bracketing the statements that you want to skip with delimited comments `/* .. */` will not work if the statements include delimited comments (`/* ...*/`). Here is the original program to be skipped.

```
/* Delimited Comment Does Not Work Here */
proc freq data=school;
  /* checking data */
  table gender * ethnicity;
run cancel;
proc options group=macro;
run;
```

Here is the original program with delimited comments around the first 5 lines. However, the 6th line (`proc options`) is also highlighted as a comment. That's because on the first line, there is an `/**/`. The first `/*` ends the delimited comment and the following asterisk (`*`) starts the next statement. Because it begins with an asterisk, it is a comment statement that ends at the semi-colon. Then there is another asterisk, etc. The highlighting makes it easy to see that more statements were taken as comments than intended. This is not the intended result.

```
/* /* Delimited Comment Does Not Work Here */*/
/*proc freq data=school;*/
/* /* checking data */*/
/* table gender * ethnicity;*/
/*run cancel;*/
proc options group=macro;
run;
```

Making each statement a comment statement (`*....;`) is too tedious for more than a few lines of code.

Paul Grant's SUGI 23 paper offers a simple solution—bracket the code with `%MACRO SKIP;` and `%MEND SKIP;` statements. This "hides" any SAS statements from the SAS supervisor. If the macro is never called, then it is never compiled and executed.

Here's an example using the SKIP macro. You can give the macro any name that you want, such as `SKIPFREQ` to indicate the purpose of the code. Notice that I have changed the background of the macro definition, macro keyword, macro text, `%MACRO`, `%MEND` to be cyan to easily recognize the macro. See the topic "Enhanced Editor Feature" above on how to change the appearance of code in the editor.

```
data school;
  set allschools (where code=12);
run;

%macro skip;
proc freq data=school;
  /* checking data */
  table gender * ethnicity;
run;
%mend skip;
```

```
proc print data=school;  
run;
```

So, in general the skip macro looks like:

```
SAS statements to execute  
%macro skip;  
SAS statements to skip  
%mend skip;  
SAS statements to execute
```

You can have multiple SKIP macros. The skip macro can be repeated:

```
SAS statements to execute  
%macro skip;  
SAS statements to skip  
%mend skip;  
SAS statements to execute
```

```
%macro skip;  
More SAS statements to skip  
%mend skip;  
SAS statements to execute
```

Also, the skip macro can be nested:

```
SAS statements to execute  
%macro skip;  
SAS statements to skip  
  %macro skip;  
    More SAS statements to skip  
  %mend skip;  
%mend skip;  
SAS statements to execute
```

If you do not want to print the skipped statements to the log, use the `OPTIONS NOSOURCE` and `OPTIONS SOURCE` statements to turn off and on printing.

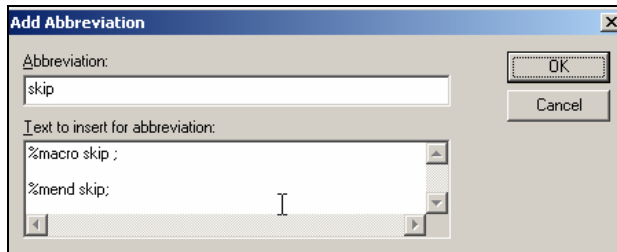
```
SAS statements to execute  
options nosource;  
%macro skip;  
SAS statements to skip  
%mend skip;  
options source;  
SAS statements to execute
```

IF you want to ran the code within the macro and not skip it, you can:

- mark and submit that code without marking the `%MACRO` and `%MEND`;
- comment out the `%MACRO` and `%MEND`;
- call the skip macro with `%SKIP`;

If you call the macro, do not repeat macros using the same macro name or nest the macros. This is to avoid confusion and have the code executed the way that you expect.

Enhanced Editor Feature. Using Enhanced Editor abbreviations make it easier and quicker to add the %MACRO SKIP and %MEND SKIP statements to your program. To add an abbreviation, press ctrl+shift+a or select *Tools→Add Abbreviation...* Fill in the screen.



To use the abbreviation, go to the location in the program where you want to add the statements. Type the abbreviation name, in this case, *skip* and press enter. In my example, I then move the *%mend skip* after the code to be skipped. You could create two separate abbreviations.

```

1 run;
2 skip
3 %macro skip ; %mend skip; (Abbrev)
4 /* checking data */
5 table gender * ethnicity;
6 run;
7
8 proc print data=school;

```

Skipping Code Using Macro Variable

Sometimes you want to only include code for debugging and skip it otherwise. Also, you want to keep the debugging code in the program—just in case. You could comment out that code each time you no longer want to run it. However, there is an easier way by using a macro variable. Put a macro variable in front of every debugging line. When you want to skip the code, define the macro variable to be an asterisk, for the beginning of a comment statement. When you want to use the statement, set the macro variable to nothing.

For example to execute the debugging put statement:

```

%let Debug = ;

data TestResults;
  set class;
  answers = nmiss(of test1-- test4);
  &Debug put 'Debug 1 ' test: answers ;
  . . .

```

Resolves to:

```

put 'Debug 1 ' test: answers ;

```

To skip the debugging statement by making it a comment statement:

```

%let debug = *;

```

```
data TestResults;
  set class;
  answers = nmiss(of test1-- test4);
  &debug put 'Debug 1 ' test: answers ;
  . . .
```

Resolves to:

```
* put 'Debug 1 ' test: answers ;
```

It is now a comment statement and is not executed.

Conclusion

This paper gave ways to submit portions of code and ways to skip portions of code. All of these ways can be a time-saver in the development of your SAS program.

Also, you can see that in SAS, if there is one way to do something, there are least three ways to do it.

References

Grant, Paul (1998), "The 'SKIP' Statement", *Proceedings of the Twenty-Third Annual SAS Users Group International Conference*, Paper 76-23.

Contact Information

Helen Carey
Carey Consulting
808.235.4070
carey@hawaii.edu

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.