Paper 11822-2016

# To Macro or not to Macro: That Is the Question

Claudine Lougee, Dualenic LLC

## ABSTRACT

Do you need a macro for your program? This paper provides some guidelines, based on user experience, and explores whether it's worth the time to create a macro (for example, a parameter-driven macro or just a simple macro variable). This paper is geared toward new users and experienced users who do not use macros.

## INTRODUCTION

How many times have you asked yourself, "It is worth my time to create a SAS® macro for this program?"

Macros can be written for many reasons. In most situations, it is up to the programmer to decide if it is worth the time, effort, and skill to create a macro for a program. There are several ways to write macros. The paper explores simple macro options to consider and does not teach the use of macros. The examples provided in this paper can be used by new programmers or experienced programmers in any job environment with any skill level.

## SIMPLE MACRO VARIABLE

A simple macro variable is a way to add text to a program which may be changed by different users. For example, a location of the SAS library or the output location of a file is a great use of a macro.

Here is an example of a macro variable called "***userloc***":

```
%LET userloc=C:\lougee\SGF2016;
ODS RTF FILE="&userloc.\class_report.rtf";
PROC PRINT DATA=SASHELP.CLASS;
RUN;
ODS RTF CLOSE;
```

In the above example, the report user location is used as a simple macro variable to create the ODS output. The macro variable, ***userloc***, could be used several times throughout the program. If a different user wanted to use the same code, they could easily change the macro variable to a different report location and run the program with ease. If the program had several reports and thousands of lines of code, it would be very useful to only change one line of code instead of using the find and replace method several times and hope to capture all the right places to change within the code and not change the wrong ones unintentionally.

### %LET STATEMENT

The %LET statement can be used for system generated or user-generated macro variables. Adding the existing system generated macro, ***sysdate***, to the code will add the system date to the file name.

```
%LET date=&sysdate.;
ODS RTF FILE="&userloc.\class_report_&date..rtf";
PROC PRINT DATA=SASHELP.CLASS;
RUN;
ODS RTF CLOSE;
```

System generated macros can be used without the let statement and resolve. There would be no need to create a macro variable called “date” by using the system date.

```
%LET date=&sysdate.;
ODS RTF FILE="&userloc.\class_report_&sysdate..rtf";
PROC PRINT DATA=SASHELP.CLASS;
RUN;
ODS RTF CLOSE;
```

System generated or automatic macros can be seen in the log using the following statements:

```
%PUT _automatic_;
```

Global or user-generated macro can be seen in the log with the following statements:

```
%PUT _global_;
%PUT _user_;
```

### ADDING A DATE STAMP TO THE LOG

In this example, a log is created for review every time the program is executed. If the program runs once a day and you want to review the output each time the program executes, a date stamp may be a nice addition.

```
PROC PRINTTO LOG="&userloc.\logresults_&sysdate9..txt" new;
RUN;
PROC PRINTO PRINT="&userloc.\printresults_&sysdate9..txt" new;
RUN;
```

Note of caution: Some programmers do not shut down their programs or computer at night before going home. If you are one of those programmers, the sysdate is the date when the SAS software was initiated. If you want to create the actual day that the report was created, you can use following code which generates the actual date and not the date that SAS software program opened.

Use the following code for today’s date:

```
%LET date =%TRIM(%QSYSFUNC(DATE(), date9.));
```

## PARAMETER DRIVEN MACRO

The parameter driven macro can be used when multiple items may need to change within code. If the same report is requested often with different criteria, such as a specific age qualifier, a parameter driven macro may be the answer.

### %GETAGE MACRO

```
%LET userloc=C:\lougee\SESUG;
%LET folder=Reports;
%LET rptname=class_report;
%LET date=%TRIM(%QSYSFUNC(DATE(), date7.));
```

```
%macro GETAGE(age);
ODS RTF FILE= "&userloc.\&folder.\&rptname._&date..rtf";
TITLE1 "Where Age is &age.";
PROC PRINT DATA=SASHELP.CLASS;
      WHERE age= age.;
RUN;
ODS RTF CLOSE;
%mend;
```

Use the following code to run the macro:

```
%GETAGE(11)
```

**Error! Reference source not found.** provides the results of the GETAGE macro where age 11 was specified in the macro call.

*Where Age is 11*

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 11 | Joyce | F | 11 | 51.3 | 50.5 |
| 18 | Thomas | M | 11 | 57.5 | 85.0 |

**Output 1. Output from the GETAGE macro**

## WHEN TO USE A MACRO

Here are some basic questions to ask when deciding to create a macro:

1. Do I need a macro?
2. Do I know how to create the right code for a macro variable?
3. Is this code I will run again later using similar techniques?
4. Will someone else inherit this code later?
5. Will it help me later if there is a macro within this code?

If you answered at least 3 of the above questions ‘yes’, then I would suggest you go for it!

Maybe if you only answered 1 ‘yes’ above, a macro may still be a great idea.

Here are some basic answers to the above questions:

1. Even if you don’t need a macro, a macro can be fun to create and learn if you don’t know how
2. Everyone creates macro differently, no two macros are created equally (unless you borrowed code)
3. All code can be reused in some way (borrow, borrow, borrow)
4. Even if you nobody inherits your code, someone may borrow your code
5. I’ve never found a macro to hinder the process

## CONCLUSION

In summary, it is typically the programmer's choice to decide if the macro helps or hinders the process. It is always a good practice to place the macros in the beginning of the program, especially if the macro variable is going to change every time the program runs. It is not a good idea to add macros inside of macros which will need to change in the middle of the program every time. No programmer should add macros so elaborate without providing documentation to explain what the macro does or is intended to do. Macros can be used for programs which need little to no maintenance or used as a means create logs or time stamps on data input or program output.


## ACKNOWLEDGMENTS

Thanks to the SAS® Global Forum Executive Board, SAS Global Users Group, and all Global Forum volunteers and attendees for the opportunity to publish and present this paper. Thanks to all mentors, presenters, and instructors who provided me with examples and material for this paper. Thanks to SAS® for allowing these conferences to continue to exist and serve the user group community.


## RECOMMENDED READING

- *SAS® Macro Guide Made Easy (Michele Burlew)*
- *Carpenter's Complete Guide to the SAS® Macro Language (Art Carpenter)*
- *The Little SAS® Book (Delwiche and Slaughter)*

## CONTACT INFORMATION

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


Claudine Lougee
Dualenic, LLC
claudine@dualenic.com