Paper BI-09-2014

%LET Me Help You Improve Your Reporting

Lori Sissing, CAPITAL Services, Sioux Falls, SD

ABSTRACT

Building and maintaining reports can be tedious. Reports often need to be run daily, weekly, or monthly and for multiple portfolios or departments. Many times dates or other elements need to be updated in multiple places throughout the program. Ensuring all necessary elements are changed each time the report is updated can take time, and an element can easily be missed resulting in inaccurate output. This paper describes how using macros and macro variables can decrease programming and updating complexity while saving time. The examples described will give readers a template for improving their own reports.

INTRODUCTION

This paper describes and gives simple examples of how SAS® macros and macro variables can be used to improve reporting. A very brief introduction to macro and macro variables will be provided, but readers with no prior macro experience should also reference the recommended reading sections below for a more in depth discussion of the details of this topic.

USING %LET TO SIMPLIFY UPDATING REPORTS

One of the quickest ways to make reports more efficient is to introduce %LET statements to the program.

```
%LET macro-variable = value;
```

The *value* is a character string that can be up to 65,534 characters long (Li, 2010). Case, quotes, mathematics etc. are maintained in the value but leading and training blanks are removed. A few examples are shown below:

Statement	Resolved value
`%LET macro-variable1 = 1 + 2;`	`1 + 2`
`%LET macro-variable2 = " SAS ";`	`" SAS "`
`%LET macro-variable2 =  leading spaces;`	`leading spaces`

If reports have values that change over time %LET allows the programmer to update all changes at the top of the program and avoid missing a change. For example, in the below program the date is referenced twice.

```
%LET Date='01JAN2014'd;
PROC SQL; CREATE TABLE REPORT AS SELECT*
    FROM data1
    WHERE firstcycledate>=&DATE
          AND firstpaymentdate >=&DATE;
    QUIT;
```

Since %LET is any character string it can be used for any value that changes frequently; date, department, employee, company, etc.

%LET statements can also contain macro functions that resolve to character values. This allows you to make macro variables that update on their own. %SYSFUNC allows most data step functions to be used in the macro environment (Yan & Wang).

One of the most useful ways I have found to use %SYSFUNC is with dates so reports will always return the last x days of data. This way if a report is scheduled to run every Monday, it will always be last week's data and will not require any updating.

For example, with the following &start variable and WHERE statement the report will always run for the last seven days.

```
%let start =%sysfunc(intnx(day,%sysfunc(date()) ,-7),date9.);

WHERE Date>="&START"d;
```

Since &START resolves as a date, it needs to be in double quotes as macros will not resolve in single quotes, and the quotes around the date are necessary for the WHERE statement.

MACROS

Macros and macro parameters alone or in combination with %LET statements can also improve reporting.

MACRO DEFINITIONS

Macro definitions can be used to execute a section of code at various places and under different conditions throughout a program (Sadof). All macro definitions, and all macros covered by this paper, begin with %MACRO and end with %MEND. This type of macro is a name style macro and is most efficient because it starts with % which notifies the word scanner to pass the token to the macro processor. The other two types of macros are command style and statement style but will not be discussed in this paper.

```
%MACRO macroname;
    DATA b;
      SET a;
      RUN;
%MEND macroname;
```

To invoke or execute the macro, it must first be defined. The macro can be called with the following:

```
    %macroname;
```

Macros are most useful when blocks of code need to be run multiple times. Using a macro helps with debugging as the code is more condensed.

MACRO PARAMETERS

Parameters can be added to macros to make them more versatile and reusable. Parameters are values that are passed to the macro at the time of invocation and are indicated in parenthesis inside the macro definition (Sadof).

```
%MACRO macroname (parameter1, parameter2);
    DATA b;
      SET &parameter1;
       IF salary>&parameter2;
      RUN;
%MEND macroname;
```

In the above example you could execute the macro *macroname* one time but invoke it several times for different datasets and different salary cutoffs, for example:

```
%macroname (dsn1, 50000);
%macroname (dsn2, 90000);
```

Parameters are helpful if the same set of code needs to be executed for a list of people or departments. Performance reviews for example could be executed annually for everyone in the department.

```
%MACRO Review (employeename, employeeid);
    DATA &employeename;
      SET Performancedata;
       IF employeeid=&employeeid;
           …details…
      RUN;
%MEND Review;

%Review (Bob, 102);
%Review (Sue, 98);
```

The program to pull in all performance measurements is written once and can be called by each employee aiding in debugging efforts and condensing programs.

CONDITIONAL LOGIC

Conditional logic can be worked into macros to aid in reporting. Reports may need to be run weekly, daily, monthly or other frequencies. If the frequency is always the same, the programmer may simply code in the range and the report will always return the correct data back X amount of days. What if however the report needed to be run for different time frames depending on day of the week or month? Instead of manually changing the date range each time, use conditional logic to incorporate the reports run schedule into the macro. For example if a report summarizes yesterday's progress, but no progress is made on the weekends then Monday's report will always be blank but should really outline Friday's progress. The macro program below shows a simple way to have the program pull data from three days ago (Friday) if it is Monday and otherwise to pull data from yesterday. Scheduling options can be used to have the report run on only Monday- Friday.

```
%MACRO DAY;
    %IF &SYSDAY=Monday %THEN
       %DO;
           %LET start=%sysfunc(intnx(day,%sysfunc(date()) ,-3),date9.);
           …program…
       %END;
    %ELSE
       %DO;
           %LET start=%sysfunc(intnx(day,%sysfunc(date()) ,-1),date9.);
           …program…
       %END;
%MEND DAY;
```

This same logic can be used many ways to decrease the need to change date ranges manually. Below are a few examples.

- Running monthly reports based on days of the month
- Running reports for different lengths (monthly or weekly) depending on department.
- Controlling report layout depending upon audience (executive, manager, etc.)

AUTOCALL LIBARIES

The macros discussed so far are session compiled macros, which must be compiled each time the program is run. Macros are also able to be stored, already compiled, in an external file called an autocall library. Autocall libraries are helpful when the macro will need few updates or if multiple users use the macro.

To create an autocall library, first assign a libref with the LIBNAME statement. Next use a FILENAME statement with the CATALOG argument to assign a fileref that contains the autocall library. This example creates a fileref called STRDMACS that points to the STRDMACS.MYAUTOS catalog.

```
libname STRDMACS 'SAS-data-library';
filename STOREDMACS catalog 'strdmacs.myautos';
```

Now that the autocall library is set up, let us discuss how to call the macro. First, the MAUTOSOURCE system option is needed to enable the autocall facility and SASAUTOS is needed to specify the autocall library. These can be set with the OPTIONS statement as below.

```
OPTIONS MAUTOSOURCE
OPTIONS SASAUTOS=LIBRARY1;
```

With the system options set calling a stored compiled macro is the same as calling a session compiled macro. The advantage of utilizing the autocall library is it saves compilation time each time the program is run and allows multiple coders to access and run the same program without having to pass code back and forth.

USING MACRO VARIABLES TO DOCUMENT REPORT DETAILS

In addition to using macro variables to update reports, they can be used within reports to document the details, time, user, etc. of the reporting. SAS has many built in macro variables that can be used to aid in the documentation; a few will be briefly discussed.

Macro variables can be used to add footnotes that update each time the report is run. As an example:

```
FOOTNOTE "This report was generated on &SYSDAY, &SYSDATE";
```

Which resolves to:

```
This report was generated on Monday, 06OCT14.
```

Other built in variables that are helpful in documenting reports are

- &SYSJOBID: User id of report executer
- &SYSTIME: Specific time SAS job was started
- &SYSVER: SAS Version

Users are also able to create their own macro variables that describe the purpose of the report, who the report is for, when the report was created, etc. The advantage of the variables is they can be used for multiple reports, but if they are used only once and do not change a footnote is sufficient.

CONCLUSION

This paper presented multiple ways SAS macro variables and macros can save programming time, reduce program complexity and create more robust reports. Applying these techniques will save the programmer time and errors.

REFERENCES

- Li, A. X. (2010). When Best to Use the %LET Statement, the SYMPUT Routine, or the INTO Clause to Create Macro Variables. Retrieved July 31, 2014, from Support.SAS.com: http://support.sas.com/resources/papers/proceedings10/028-2010.pdf
- Sadof, M. G. (n.d.). Macros from Beginning to Mend. Retrieved July 29, 2014, from www.ats.ucla.edu: http://www.ats.ucla.edu/stat/sas/library/nesug99/bt066.pdf
- SAS. (n.d.). SAS Support. Retrieved 8 16, 2014, from http://support.sas.com/documentation/cdl/en/mcrolref/61885/HTML/default/viewer.htm#a001328769.htm
- Yan, L., & Wang, H. (n.d.). Exploring the SAS Macro Function % SYSFUNC. Retrieved July 31, 2014, from lexjansen.com: http://www.lexjansen.com/pharmasug/2008/cc/CC11.pdf

RECOMMENDED READING

- Sadof, Michael G. "Macros from Beginning to Mend A Simple and Practical Approach to the SAS® Macro Facility." http://www.ats.ucla.edu/stat/sas/library/nesug99/bt066.pdf
- %LET Statement http://support.sas.com/documentation/cdl/en/mcrolref/61885/HTML/default/viewer.htm#a000543704.htm

CONTACT INFORMATION

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

Name: Lori Sissing
Enterprise: CAPITAL Services
Address: 500 E North 60th Street
City, State ZIP: Sioux Falls SD 57104
E-mail: lori.sissing@capitalsvcs.com
Web: http://www.capitalsvcs.com/

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.