Paper 105

SAS® Macro Magic is not Cheesy Magic!

Let the SAS Macros Do the Magic of Rewriting Your SAS Code

Robert Williams
WellPoint, Inc.

ABSTRACT

Many times, we need to rewrite weekly or monthly SAS programs to change certain key statements such as the conditions inside the WHERE statements, import/export file paths, reporting dates and SAS data set names. If we hard coded these statements, it is cumbersome and a chore to read through the SAS code to rewrite these hard-coded statements. Sometimes, we might miss an important statement that needs to be re-coded resulting in inaccurate data extract and reports. This paper will show how the SAS Magic Macros can streamline and eliminate the process rewriting the SAS codes. Two types of SAS Macros will be reviewed with examples: 1) Defining SAS Macro variables with values using %LET to be rewritten in the SAS statement using & sign. 2) Creating SAS Macro programming using %MACRO and %MEND to write a series of SAS statements to be rewritten in the SAS code. You will be amazed how useful the SAS Magic Macro is for many of your routine weekly and monthly reports. Let the SAS Magic Macro relieve you of the tedious task of rewriting many of the SAS statements!

INTRODUCTION

Many times, we hard-code SAS codes with a specific date range inside a WHERE statement for a weekly or monthly report. We may even hard-code the destination file path for a PROC EXPORT or an ODS output with hard coded report headers. Then at a later time, we may need to change the date range, folder paths or the ODS output report headers. That means, we will have to rewrite the SAS code to accommodate the changes. If it is a short and simple SAS code, it may not too much labor to manually rewrite the code. However, if it were a long SAS code requiring multiple changes throughout the code; it can be labor intensive to read through the code to manually rewrite some of the lines at the appropriate places. With the extra rewriting labor, it can lead to syntax errors or even miss rewriting one or more important lines of code leading to inaccurate data extract and reports. In this paper, I will show two types of SAS Macros that will eliminate the process of rewriting the SAS codes.

1) SAS Macro Magic #1: Defining SAS Macro variables with values using %LET statements to be rewritten in the SAS statement using the & sign.
2) SAS Macro Magic #2: Creating SAS Macro programs using %MACRO and %MEND to write a series of SAS statements to be rewritten in the SAS code.

The best way to show the SAS Macro Magic is by example in a hypothetical situation. I will be using SAS snacks dataset available in everyone's SASHELP library. This dataset provides the daily snack food sales such as quantities sold, retail price of product, date of sale and product name. This paper will be using this dataset to demonstrate the use of SAS Macros to produce three PDF sales reports for popcorn, chips and pork rinds in the month of June 2004 like this:

Monthly Popcorn Sales Report

(Sold between 01JUN2004 and 30JUN2004)

Snack	Quantity Sold	Average Price
Buttery popcorn	161	1.99
Carmelized popcorn	70	2.99
Cheddar cheese popcorn	139	2.87
Low-fat popcorn	167	1.49
GRAND TOTAL	*537*	*2.34*

Display 1. Popcorn sales report with quantity sold and average price for the month of June 2004

CODING DILEMMA: HARD-CODING THE DATA EXTRACT AND REPORT

To produce the above report (Display 1), we will need to write a SAS data step to extract snack sales data sold between 6-1-2004 and 6-30-2004 and write an ODS output for the snack type. A typical hard-coded SAS code would be like this:

```
data work.monthly_snack_sales;
       set SASHELP.SNACKS;
       where date between '01JUN2004'd and '30JUN2004'd;
run;

options nodate;
ods pdf file="C:\SESUG 2014\Popcorn_Sales_Report.pdf" style=festival;
title;
footnote;
title1 'Monthly Popcorn Sales Report';
title2 '(Sold between 01JUN2004 and 30JUN2004)';
PROC Report DATA=work.monthly_snack_sales NOWINDOWS HEADLINE HEADSKIP;
       where index(upcase(product),'POPCORN');
       COLUMN product qtysold price;
       DEFINE product / 'Snack' group;
       DEFINE qtysold / 'Quantity Sold' analysis sum format=comma10.0;
       DEFINE price / 'Average Price' analysis mean format=comma6.2;
       /* Creates grand total line */
       RBREAK AFTER / SUMMARIZE;
       compute after;
              product = 'GRAND TOTAL';
       endcomp;
RUN;
ods pdf close;
```

We will face two dilemmas that require rewriting lines of code for separate reports. The first dilemma would be to change the date range of the data extract next month's report. It will require rewriting the date range in the WHERE statement for the data step. The other dilemma is the changing several of lines in the ODS PDF output process such as the PDF file name, the TITLE lines (for product name and date range) and the WHERE statement for the product type selection. This is even more rewriting of code for each product: popcorn, chips and pork rinds

Instead, we will apply the SAS Macro magic to take care of the rewriting some of these lines.

SAS MACRO MAGIC #1: MACRO VARIABLE

In this technique, we will create and use SAS macro variables to be used in the WHERE statement in the data step. Creating and using the SAS macro variable is a two-step process for each variable using the %LET statement and the & character to rewrite the content of the macro variable.

```
/* This creates SAS macro varibles for begin and end dates */
%let beg_date = 01JUN2004;
%let end_date = 30JUN2004;

data work.monthly_snack_sales;
       set SASHELP.SNACKS;
       where date between "&beg_date"d and "&end_date"d;
run;
```

The above code followed the two step process of using the SAS macro variables:

- Using the %LET to create the begin date and end date range by naming macro variables `beg_date` and `end_date` and assign the values 01JUN2004 and 30JUN2004 respectively.
- Then we invoke both macro variables inside the WHERE statement to be used as the date range. The way to invoke the macro variable is to precede the macro name with ampersand sign: "&" such as **`&beg_date`** and **`&end_date`**. The ampersand sign: "&" will "rewrite" the SAS code with the macro variable values:

01JUN2004 and 30JUN2004 in the WHERE statement. NOTE: Use double-quotes so SAS can notice the & sign to put the values within the quotes so we can use it with the date conversion letter "d".

After submitting the code to run, you will see in the SAS log that the macro variable values has been successfully invoked in the WHERE statement. We just did the magic of re-writing the SAS code in the WHERE statement.

```
20         data work.monthly_snack_sales;
21           set SASHELP.SNACKS;
22           where date between "&beg_date"d and "&end_date"d;
23         run;

NOTE: There were 1050 observations read from the data set SASHELP.SNACKS.
      WHERE (date>='01JUN2004'D and date<='30JUN2004'D);
NOTE: The data set WORK.MONTHLY_SNACK_SALES has 1050 observations and 6
variables.
NOTE: DATA statement used (Total process time):
      real time           0.05 seconds
      cpu time            0.00 seconds
```

You can also find out what values are in the SAS macro variable by using %PUT statements to write to the SAS log.

```
/* This writes what is in the SAS macro variables */
/* so we can "see" it in the SAS log */
%put The begin date is:  &beg_date;
%put The end date is:    &end_date;
```

By submitting this code, you will see the same values written to the SAS log. This is a good way to debug your process to make sure the SAS Macro magic is working.

```
20         %put The begin date is:  &beg_date;
The begin date is:  01JUN2004
21         %put The end date is:    &end_date;
The end date is:    30JUN2004
```

SAS MACRO MAGIC #2: SAS MACRO PROGRAMS

In this technique, we create SAS macro programs with parameters. Like macro variables, macro programs (also known as macros) enable us to "re-write" multiple lines of text into the SAS programs. This makes the SAS program more dynamic and reusable.

The SAS macro program with positional parameters is defined with the following syntax.

```
%MACRO macro-name(parameter-1,parameter-2,…,parameter-n);
      <SAS PROGRAM LINES>
%MEND macro-name;
```

The positional parameters inside the %MACRO line are named as if you were using multiple %LET statements in the order that you assign the values in the macro call. In other words, you must specify the values of the macro parameters in the *same order* in which they are defined.

So, in our example, we will take the ODS statements and create the SAS macro programs with three positional parameters: 1) PDF report file name as `rpt_file_name`, 2) report title as `rpt_title` and 3) the WHERE statement conditions as `rpt_where_statement`. These parameters will allow dynamic processes to create separate PDF's for each product type. Here is how the macro program will be set with the ODS statements. We will call the macro program "snack_reports" with the three positional parameters defined.

```
options mprint;
%MACRO snack_reports(rpt_file_name, rpt_title, rpt_where_statement);
options nodate;
ods pdf file="C:\SESUG 2014\&rpt_file_name._Sales_Report.pdf" style=festival;
title;
```

```
footnote;
title1 "Monthly &rpt_title Sales Report";
title2 "(Sold between &beg_date and &end_date)";
PROC Report DATA=work.monthly_snack_sales NOWINDOWS HEADLINE HEADSKIP;
       where &rpt_where_statement;
       COLUMN product qtysold price;
       DEFINE product / 'Snack' group;
       DEFINE qtysold / 'Quantity Sold' analysis sum format=comma10.0;
       DEFINE price / 'Average Price' analysis mean format=comma6.2;
       /* Creates grand total line */
       RBREAK AFTER / SUMMARIZE;
       compute after;
              product = 'GRAND TOTAL';
       endcomp;
RUN;
ods pdf close;
%MEND snack_reports;
```

There are several key points about this macro program.

- The macro program starts with %MACRO statement and ends with %MEND statement with the same macro name "`snack_reports`".
- From each of the three parameters defined in the %MACRO statement, there are macro variables invoked within the macro program with the ampersand "&" in yellow.
- In the report file statement, we added a period to the end of the macro name so SAS knows that "`&rpt_file_name`," is the macro name, not the rest of the file name.
- We use the double quotes when invoking macro variables so SAS can see they are macro variables that it needs to "re-write" the text inside the quotes.
- The macro variables we created for the data step is reused in the ODS process: `&beg_date` and `&end_date` lll. This is the beauty of using macro variables in multiple places in the SAS code so we don't have to re-write it every time.
- We have to submit the SAS macro program so it can be loaded into SAS. Make sure you submit all lines from %MACRO line to %MEND line.
- The options: `options mprint;`, is to allow the macro program to write to the log. This is helpful for debugging the SAS macro codes.

By now, we have loaded the macro program "`snack_reports`". In order to run the macro program, we have to "call" the macro by preceding the macro program name with "%" like the following.

```
%snack_reports(Chips, Chips, index(upcase(product),'CHIP'));
%snack_reports(Popcorn, Popcorn, index(upcase(product),'POPCORN'));
%snack_reports(Rinds, Pork Rinds, index(upcase(product),'RIND'));
```

Note, how we make sure we supply the text value to each of the positional parameters in the exact order. The first one is for the file name, the second one is for the report title and the third one is for the WHERE statement. By submitting these three lines, we are calling up the macro program three times which will generate three PDF files using the supplied parameters. So, we are using the magic of dynamically writing the ODS statement three times for chips, popcorn and pork rinds as shown below (display 2, display 3, and display 4.).

Monthly Chips Sales Report

(Sold between 01JUN2004 and 30JUN2004)

Snack	Quantity Sold	Average Price
Baked potato chips	176	1.97
Barbeque potato chips	177	1.99
Classic potato chips	880	0.99

Easy dip tortilla chips	172	1.99
Multigrain chips	125	2.87
Ruffled potato chips	176	1.99
Salt and vinegar potato chips	128	2.87
Sun-dried tomato multigrain chips	130	2.87
Tortilla chips	349	0.99
WOW potato chips	58	2.99
WOW tortilla chips	109	2.87
GRAND TOTAL	*2,480*	*2.22*

Display 2. Chips sales report with quantity sold and average price for the month of June 2004.

Monthly Popcorn Sales Report

(Sold between 01JUN2004 and 30JUN2004)

Snack	Quantity Sold	Average Price
Buttery popcorn	161	1.99
Carmelized popcorn	70	2.99
Cheddar cheese popcorn	139	2.87
Low-fat popcorn	167	1.49
GRAND TOTAL	*537*	*2.34*

Display 3. Popcorn sales report with quantity sold and average price for the month of June 2004.

Monthly Pork Rinds Sales Report

(Sold between 01JUN2004 and 30JUN2004)

Snack	Quantity Sold	Average Price
Barbeque pork rinds	130	1.71
Extra hot pork rinds	431	0.99
Fried pork rinds	147	1.49
GRAND TOTAL	*708*	*1.40*

Display 4. Pork rinds sales report with quantity sold and average price for the month of June 2004.

CONCLUSION

The two Macro Magic techniques has saved us a significant amount of manual labor of coding of rewriting hard-coded SAS codes. This example may not seem like it needs a lot of re-coding to do. But these two techniques will be a big help if we have a large number of repetitive SAS statements or conditions to re-write. This is not limited to just making the WHERE statements or ODS statements. You will be amazed how useful these techniques can be applied to dozens or hundreds of routine reports especially on a weekly or monthly basis. *Macro Magic* can be expanded to include other SAS procedures and other DATA steps. Let the *Macro Magic* do the re-writing the SAS codes for you! I am confident that you will add these two valuable techniques to your expanding SAS bag of coding tips.

ACKNOWLEDGEMENTS:

I would like to thank the Southeast SAS Users Group for accepting my abstract and paper as well as for conducting a successful conference.

REFERENCES

SAS Certification Prep Guide: Base Programming for SAS 9 Third Edition Cary, NC: SAS Institute Inc. 2011

SAS Certification Prep Guide: Advanced Programming for SAS 9 Third Edition Cary, NC: SAS Institute Inc. 2011

SAS Help Data Sets Cary, NC: SAS Institute Inc. 2013

The Web's Free 2013 Medical Coding Reference. (http://www.icd9data.com/)

CONTACT INFORMATION

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

Robert Williams
WellPoint, Inc.
1300 Amerigroup Way
Virginia Beach, VA 23464
Robert.Williams@wellpoint.com

SAMPLE SAS CODE

This is the full sample SAS code for this paper including SESUG_DX_LISTING table (Table 1) and the sample claims data. Run the first two DATA steps to import the diagnosis listing and the sample claims data. The rest of the code is for the two *Robo-Coding* techniques.

```
/* This creates SAS macro varibles for begin and end dates */
%let beg_date = 01JUN2004;
%let end_date = 30JUN2004;

/* This writes what values are in the SAS macro variables */
/* so we can "see" it in the SAS log */
%put The begin date is:  &beg_date;
%put The end date is:    &end_date;

/* Pulls the sales data for the date range */
data work.monthly_snack_sales;
      set SASHELP.SNACKS;
      where date between "&beg_date"d and "&end_date"d;
run;

options mprint;
/* Loads up the MACRO that includes the ODS reports */
%MACRO snack_reports(rpt_file_name, rpt_title, rpt_where_statement);
options nodate;
ods pdf file="\\va01pstodfs003.corp.agp.ads\users\VA1\rwilli4\My Documents\VASUG and
SESUG Presentations\SESUG 2014\SAS Coding Sandbox\&rpt_file_name._Sales_Report.pdf"
style=festival;
title;
footnote;
title1 "Monthly &rpt_title Sales Report";
title2 "(Sold between &beg_date and &end_date)";
PROC Report DATA=work.monthly_snack_sales NOWINDOWS HEADLINE HEADSKIP;
      where &rpt_where_statement;
      COLUMN product qtysold price;
      DEFINE product / 'Snack' group;
      DEFINE qtysold / 'Quantity Sold' analysis sum format=comma10.0;
      DEFINE price / 'Average Price' analysis mean format=comma6.2;
```

```
        /* Creates grand total line */
        RBREAK AFTER / SUMMARIZE;
        compute after;
               product = 'GRAND TOTAL';
        endcomp;
RUN;
ods pdf close;
%MEND snack_reports;

/* Calls the SAS Macro snack_reports for each sales product */
%snack_reports(Chips, Chips, index(upcase(product),'CHIP'));
%snack_reports(Popcorn, Popcorn, index(upcase(product),'POPCORN'));
%snack_reports(Rinds, Pork Rinds, index(upcase(product),'RIND'));
```