Tips and Tricks for Getting the Most Out of the SAS® Macro Facility BB-13-2015

Scott A. Miller
Credit Risk Analytics
Wells Fargo

Tips and Tricks

- These are ideas
  - Use what works for you
  - Adapt them
- Read full paper online
  - More tips
  - Additional detail
  - Full source code

Definitions

- Macro Variable
  - Stores a string up to 65,534 characters
  - Referenced with &

- Macro
  - A program run by macro facility
  - Referenced with %

Parameterize Programs with Macro Variables

- Most programs have parameters
  - Date ranges
  - Max or Min values
  - Excluded categories
  - etc.
- These things change!

Parameterize Programs with Macro Variables

- In the first few lines of program
  - Create macro variables for parameters

- Benefits:
  - Better organization – find parameters in one place
  - Reduced redundancy – easier to modify

Parameterize Programs with Macro Variables

```
*****  BEGIN -- program parameters  *****;
%let  Monthly_begin_date = '01JAN2015'd;
%let  Monthly_end_date = '31JAN2015'd;

%let  Systolic_BP = 150;
%let  Diastolic_BP = 90;

%let  Busiest_US_Airports = "ATL" "LAX" "ORD" "DFW" "JFK";
*****  END -- program parameters  *****;
```

Parameterize Programs with Macro Variables

- A Macro Variable can hold a list

```
%let Busiest_US_Airports = "ATL" "LAX" "ORD" "DFW" "JFK";

proc sql;
   select *  from flights
   where airport in (&Busiest_US_Airports.);
quit;

data test;
   set flights (where=(airport in (&Busiest_US_Airports.)));

   array airports {5} $3 (&Busiest_US_Airports.);
run;
```

Selecting a Name

- Avoid common SAS keywords
  - Otherwise, tracing your code will be painful

- Some Examples:

  DATA, RUN, FILE, END, IN, OUT,

  DAY, MONTH, YEAR, MIN, MAX,

  SUM, LT, GT, EQ, NE, AND, OR

Selecting a Name

- Macro and Macro variable names
  - up to 32 characters long
- Which is better?
  - DT1 &
    DT2
  - Month_Begin_Date &
    Month_End_Date

Automate Date Macro Variables

- Date parameters are typically updated each time the program is run

- It's easy to automate
  - Save effort
  - Eliminate errors

Automate Date Macro Variables

- Get current date with Today function
- Compute the interval you need using Intnx function

Automate Date Macro Variables

- Example: first day of previous month

```
intnx ('month', today(), -1, 'end')
```

- From the Log:

```
WHERE ( date <= 20331 )
```

Automate Date Macro Variables

- Make human-readable date values

```
"'" || put (begin_date, date9.) || "'d"
```

- From the Log:

```
WHERE ( date <= '31AUG2015'D )
```

Automate Date Macro Variables

Put all the pieces together…

```
data _null_;
   report_date = today ();

   Monthly_end_date =
          intnx ('month', report_date, -1, 'end');

   call symput ('Monthly_end_date',
          "'" || put (Monthly_end_date, date9.) || "'d");
run;
```

Monthly_end_date
'31AUG2015'd

Set Macro Vars From CSV File

- Cons to storing parameters in .SAS file
  - Update parameters by non-programmer?
  - Update by automatic process?
  - Code locked down
- One alternative is to store parameters in .CSV file

Set Macro Vars From CSV File

- Step one: read CSV file into data set

  var_name, var_contents

- Step two: use call symput to create macro variables

```
call symput (var_name,
             var_contents);
```

Positional vs Keyword Macro Parameters

- Positional parameters are based on order listed

```
%Payment (1234, 0.05, 36)
```

  - Mistakes are hard to catch

Positional vs Keyword Macro Parameters

- Keyword parameters are specified by name

```
%Payment (loanamt=1234,
          months=0.05,
          rate=36)
```

  - Mistakes are easier to catch
  - Code is self-documenting

Define Macro Variable Scope

- Global Variable Scope
  - Value is shared through entire SAS program
- Local Variable Scope
  - Independent of same-name variables outside macro
  - Value is lost when macro finishes

Show Macro Variable Scope

```
%let  value = 50;

%macro Something;
     %do value = 1 %to 20;
     %end;
%mend Something;
%Something
```

```
%put VALUE = &value.;
```

LOG WINDOW

VALUE = 21

Show Macro Variable Scope

```
%let  value = 50;

%macro Something;
     %local value;
     %do value = 1 %to 20;
     %end;
%mend Something;
%Something

%put VALUE = &value.;
```

LOG WINDOW

```
VALUE = 50
```

Within a Macro…
Use Local Variable Scope

- Same variable name in multiple places?
  - Debugging nightmare!
- How to avoid?
  - 1. Never re-use the same variable name?
  - 2. Use %local to restrict scope

Save and Restore Options within Macro

- Your macro may need to change a System Option
  - Save previous value
  - Change option
  - Restore original value when macro completes!

Save and Restore Options within Macro

- 1. Use GetOption function to retrieve value

```
%let OLD_YC =
      %sysfunc (GetOption (YEARCUTOFF));
```

- 2. Perform macro work
- 3. Use OPTIONS statement to restore value

```
OPTIONS YEARCUTOFF=&OLD_YC.;
```

Keep “ERROR” and “WARNING” False Hits out of Log

- Problem detection code…

```
*  Validity check - write problems to log  *;
data _null_;
  set sashelp.heart;

  if (not missing (AgeAtDeath))
                and (AgeAtDeath < AgeAtStart) then do;
     error "ERROR: AgeAtDeath...";
     stop;
  end;
run;
```

Keep “ERROR” and “WARNING” False Hits out of Log

- Searching LOG for “ERROR” will find

LOG WINDOW

```
35   *  Validity check - write problems to log  *;
36   data _null_;
37     set sashelp.heart;
38
39     if (not missing (AgeAtDeath))
40                 and (AgeAtDeath < AgeAtStart) then do;
41        error "ERROR: AgeAtDeath...";
42        stop;
43     end;
44   run;

NOTE: There were 5209 observations read from the data set
SASHELP.HEART.
```

Keep “ERROR” and “WARNING” False Hits out of Log

- Use %UpCase to break up the words

LOG WINDOW

```
35   *  Validity check - write problems to log  *;
36   data _null_;
37     set sashelp.heart;
38
39     if (not missing (AgeAtDeath))
40               and (AgeAtDeath < AgeAtStart) then do;
41        %UpCase(er)ror "%UpCase(ER)ROR: AgeAtDeath...";
42        stop;
43     end;
44   run;

NOTE: There were 5209 observations read from the data set
SASHELP.HEART.
```

Extend a Macro with a Default Parameter Value

- You have a useful macro
  - Example: macro to shuffle playing cards
    - One deck of cards

- Modify the macro
  - Add parameter – # of decks
  - Don’t want to break existing code

Extend a Macro with a Default Parameter Value

- Provide a default value for new parameter

```
%macro Shuffle (NumDecks=1);
```

- Old code still works!
- New code can override default value

Pull Data from Yearly Data Sets

- Data stored in one data set per year
- PROS: Better Performance for subset
- CONS: More complicated

Pull Data from Yearly Data Sets

- Example: 18 months of data is needed

2013	J	F	M	A	M	J	J	A	S	O	N	D
2014	J	F	M	A	M	J	J	A	S	O	N	D
2015	J	F	M	A	M	J	J	A	S	O	N	D

2013	J	F	M	A	M	J	J	A	S	O	N	D
2014	J	F	M	A	M	J	J	A	S	O	N	D
2015	J	F	M	A	M	J	J	A	S	O	N	D

- You may need 2 or 3 data sets

Pull Data from Yearly Data Sets

- Step 1: use do loop to build data set list
- Step 2: store list in macro variable
- Step 3: use macro variable in SET statement

Pull Data from Yearly Data Sets

```
data _null_;
   start_year = year (&First_Day_of_Report.);
   end_year = year (&Last_Day_of_Report.);
   length list_of_datasets $250;

   do year = start_year to end_year;
         list_of_datasets = trim (list_of_datasets) ||
                            " annua
   end;
   call symputx ('list_of_datasets'
                 list_of_datasets);
run;
```

list_of_datasets

annual_data_2013
annual_data_2014
annual_data_2015

Pull Data from Yearly Data Sets

```
data use_annual_data;
   set &list_of_datasets.;

   where data_date between &First_Day_of_Report.
                       and &Last_Day_of_Report.;

      ***    Add your code here    ***;
run;
```

list_of_datasets

annual_data_2013
annual_data_2014
annual_data_2015

Contact Information

Name: Scott A. Miller
Enterprise: Wells Fargo
Location: Des Moines, IA
SMS: 515-537-4288
E-mail: smiller933 @ yahoo.com

Print All Macro Variables to Log

- You can easily print a list of all macro variables to the log
  - Explore what's available
  - Debugging

Print All Macro Variables to Log

```
%PUT _ALL_;
```

LOG WINDOW

```
AUTOMATIC SYSADDRBITS   32
AUTOMATIC SYSDATE       19JAN16
AUTOMATIC SYSDATE9      19JAN2016
AUTOMATIC SYSDAY        Tuesday
AUTOMATIC SYSHOSTNAME   MYCOMPNAME
AUTOMATIC SYSVLONG      9.03.01M2P081512
```

Print All Macro Variables to Log

- View only a subset with…

```
%PUT _AUTOMATIC_;

%PUT _GLOBAL_;

%PUT _LOCAL_;
```