CMU MSP 36-602: SAS Macros III

Howard Seltman, April 8, 2019

1) Review

a. The SAS macro facility consists of macro variables and macros (functions). These are supplemented by a few DATA step functions such as `SYMGET()` and `CALL SYMPUTX()`.

b. Macro variables are global (or local inside macro functions) **text-only variables** created with `%LET myMacroVar = my contents;` and everything between the "=" and the ";" except leading and trailing spaces is stored in the variable. Macro variables do ***not*** persist across SAS sessions.

c. Macro variables are used in the form `&myMacroVar.` (with the period often optional), and **the variable's contents are substituted into the code before the code is run**. This includes within a DATA step, so %LET cannot be used to store a dataset variable to a macro variable. You can use `CALL SYMPUTX()` to store a dataset variable to a macro variable.

d. Outside of a "step" you can use code like `%PUT myMacroVar equals &myMacroVar..;` to write messages to the log.

e. Macro functions like %LENGTH(), %SUBSTR(), and %SCAN() may be used in any code, and these also carry out their functions before your code is submitted.

f. The %STR() function may be needed if you want to include special characters in your variable contents.

g. Code like `%SYMDEL myMacroVar;` deletes macro variables.

h. You can use `%PUT _USER_;` to see all the macro variables you created.

i. You can write your own macro "functions" with the syntax:

```
%MACRO myMacroName[(myArgName1[=[myDflt1]][,myArgName2[=[myDflt2]][…]])];
  ---any text lines---
%MEND [myMacroName];
```

j. The purpose of every macro fucntion is to **generate text** that can be submitted to the SAS system, generally DATA and PROC steps. Smaller text outputs, e.g., a set of variable names are also possible. Side effects, e.g., printing information with %PUT, may also be part of a macro result.

k. Macros are invoked using syntax like `%myMacro(myArg1Txt, optArg=myArg2Txt)`

l. Semicolons are not part of macro invocation. Macros are evaluated and their return text are inserted into the "input stream" before any surrounding code is run.

m. There are two methods for storing macros between sessions (compiled and as source code). Otherwise, they need to be re-run before being used in a new SAS session.

2) Example: converting the monthly and cumulative accounting example of Macros I to a macro

a. Setup

```
*%LET wd = /folders/myfolders/36602;
%LET wd = .;
LIBNAME fiscMac "&wd/fiscalData";
OPTIONS MSTORED SASMSTORE=fiscMac;

/* Macros to add monthly data to cumulative data and print
   year-to-date results.
   Assume: ExpensesMMMYYYY.dat with variables category and
   amount (at least) are in the current folder; "cumDS" is a
   cumulative dataset with variables category, amount, month,
   and year (at least).
*/

/* New cumulative dataset /*
%MACRO newCum(cumDS) / STORE SOURCE DES="Make new cum. dataset";
  DATA &cumDS;
    LENGTH category $12 amount 8 month $3 year 8;
    STOP;
  RUN;
%MEND newCum;


/* Year to date expense plot */
%MACRO ytdExpensePlot(year, cumDS) / STORE SOURCE
    DES="Plot YTD expense means";
  TITLE "Expenses to date for &year (&SYSDATE9)";
  /* Sort by month to allow BY in UNIVARIATE */
  PROC SORT DATA=&cumDS (WHERE=(year=&year)) OUT=oneYear;
    BY month;
  RUN;
  /* Get monthly sums */
  PROC UNIVARIATE DATA=oneYear NOPRINT OUTTABLE=monthlySums;
    BY month;
    VAR amount;
  RUN;
  /* Add numeric month */
  DATA monthlySums;
    SET monthlySums;
    nMonth = MONTH(INPUT(mon||"2000", MONYY.));
  RUN;
  /* Sort months numerically */
  PROC SORT DATA=monthlySums;
    BY nMonth;
  RUN;
  /* Plot Monthly Sums */
  PROC SGPLOT DATA=monthlySums;
    SERIES X=month Y=_SUM_;
      YAXIS LABEL = "Total Expenses";
  RUN;
  TITLE;
%MEND ytdExpensePlot;
```

```
/* Year to date expense report */
%MACRO ytd(year, cumDS) / STORE SOURCE DES="Print YTD means";
  TITLE "Expenses to date for &year (&SYSDATE9)";
  PROC MEANS DATA=&cumDS (WHERE=(year=&year));
  RUN;
  TITLE;
%MEND;

/* Delete temporary datasets */
  PROC DATASETS LIBRARY=WORK NOLIST;
    DELETE oneYear monthlySums;
  RUN;
  QUIT;
%MEND;

/* Year to date expenses by categories */
%MACRO ytdCategoryPlot(year, cumDS) / STORE SOURCE
    DES="Plot YTD categories";
  TITLE "Expenses by category for this year to date (&SYSDATE9)";
  PROC SORT DATA=&cumDS (WHERE=(year=&year)) OUT=oneYear;
    BY category;
  RUN;
  /* Get monthly sums */
  PROC UNIVARIATE DATA=oneYear NOPRINT OUTTABLE=monthlySums;
    BY category;
    VAR amount;
  RUN;
  /* Plot Monthly Sums */
  PROC SGPLOT DATA=monthlySums;
    VBAR category / response = _SUM_;
      YAXIS LABEL = "Total Expenses";
  RUN;
  TITLE;
  /* Delete temporary datasets */
  PROC DATASETS LIBRARY=WORK NOLIST;
    DELETE oneYear monthlySums;
  RUN;
  QUIT;
%MEND;


/* Group of all available expense reports */
%MACRO showAll(year, cumDS)  / STORE SOURCE
    DES="All tables and plots";
  %ytd(year=&year, cumDS=&cumDS);
  %ytdExpensePlot(year=&year, cumDS=&cumDS);
  %ytdCategoryPlot(year=&year, cumDS=&cumDS);
%MEND;
```

```
/* Main monthly macro */
/* Add data from a month, and run all reports */
%MACRO addMonthData(month, year, cumDS, fileLoc) / STORE SOURCE
    DES="Add month and summarize";
  %LOCAL inname;
  %LET inname = "&fileLoc/Expenses&month&year..dat";
  DATA E&month&year; /* temporary dataset */
    INFILE &inname;
    LENGTH category $12. month $3.;
    INPUT category$ amount;
    year = &year;
    month = "&month";
  RUN;

  DATA &cumDS;
    SET &cumDS E&month&year;
  RUN;

  %showAll(year=&year, cumDS=&cumDS);
%MEND;



/* Macro to remove data, if needed */
%MACRO drop(month, year, cumDS) / STORE SOURCE
    DES="Remove all data from a month and year";
  DATA &cumDS;
    SET &cumDS (WHERE=(month^="&month" OR year NE &year));
  RUN;
%MEND;

/* See listing of stored macros */
PROC CATALOG CATALOG=fiscMac.SASMACR;
  CONTENTS;
RUN;
```

b. Usage

```
LIBNAME data "&wd/fiscalData";
LIBNAME fiscMac "&wd/fiscalData";
OPTIONS MSTORED SASMSTORE=fiscMac;

%newCum(data.expenses)
%drop(cumDS=data.expenses, month=Jan, year=2017)
%drop(cumDS=data.expenses, month=Feb, year=2017)

%addMonthData(month=Jan, year=2017, cumDS=data.expenses,
              fileLoc=&wd)
%addMonthData(month=Feb, year=2017, cumDS=data.expenses,
              fileLoc=&wd)

%showAll(year=2017, cumDS=data.expenses)
```

3) **Aside: ODS OUTPUT**

c. For any PROC, search the SAS help for "ODS Table Names" along with the PROC name to get a list of all possible SAS datasets that can be produced containing various parts of the PROC output.

d. You can add a statement like
`ODS OUTPUT OFFICIAL_TABLE_NAME = myDatasetName;`
to the PROC to send a portion of the output to the dataset whose name you specify. You can include as many NAME=name pairs as you like, and/or you can use multiple ODS OUTPUT statements. Usually you will want/need to try out an example to find out what specifically ends up in the dataset, and what the exact variable names are.

e. E.g., for PROC UNIVARIATE, there are 17 possible tables, and we note that one called "TestsForNormality" is associated with the NORMALTEST option.
Let's try this code:

```
FILENAME drd4 "&wd/data/DRD4.dat";
DATA drd4;
  INFILE drd4 FIRSTOBS=2;
  INPUT DRD4 sensPar EBS;
RUN;

TITLE "Demonstrate ODS OUTPUT";
PROC UNIVARIATE DATA=drd4 NORMALTEST;
  VAR EBS;
  ODS OUTPUT TESTSFORNORMALITY=myNormTests;
RUN;
```

The result is a dataset called myNormTests. This can be examined directly through the Explorer Window, or by using PROC PRINT, or most systematically with:

```
PROC CONTENTS DATA=myNormTests;
RUN;
```

The result of CONTENTS is:

```
Data Set Name   WORK.MYNORMTESTS       Observations  4
Member Type     DATA                  Variables     7

       Alphabetic List of Variables and Attributes

#    Variable    Type    Len    Format       Label
4    Stat        Num       8    BEST8.       Value of Goodness-of-Fit Statistic
2    Test        Char     18                 Goodness-of-Fit Test
3    TestLab     Char      4                 Label for Goodness-of-Fit Statistic
1    VarName     Char      3
6    pSign       Char      1                 Sign of p-value
5    pType       Char      9                 p-value Label
7    pValue      Num       8    PVALUE6.4    p-value
```

The result of PROC PRINT is:

Obs	Var Name	Test	Test Lab	Stat	pType	p Sign	pValue
1	EBS	Shapiro-Wilk	W	0.9609	Pr < W		0.043
2	EBS	Kolmogorov-Smirnov	D	0.1232	Pr > D		0.018
3	EBS	Cramer-von Mises	W-Sq	0.1389	Pr > W-Sq		0.034
4	EBS	Anderson-Darling	A-Sq	0.8474	Pr > A-Sq		0.028

f. Once you have the PROC output in a dataset, the usual next step is to run a DATA _NULL_; step, and use something like:

```
IF Test="Shapiro-Wilk" THEN CALL SYMPUTX("NormPVal", pValue);
```

This stores the p-value in a macro variable for later use. Alternatively, sometimes all you want to do create one or more new columns in the dataset.

g. You might think that it is a good idea to use a NOPRINT option to suppress output to the Output Window, because often you are only interested in the dataset that is created, but unfortunately, this inactivates the ODS OUTPUT statement. One thing you are allowed to do is add a statement like ODS SELECT TESTSFORNORMALITY; which limits the information in the Output Window to the same information as what goes into the dataset.

4) Full macro example using ODS OUTPUT

```
/* A macro to add LCI, UCI confidence limits endpoints to
   GLM ESTIMATE results (same as CLPARM option).
   Use:  ODS OUTPUT ESTIMATES=myEstDS OVERALLANOVA=myAnovaDS;
   to store the GLM output that is the input for this macro.
*/
%MACRO contrastCI(estimates, overallANOVA, alpha=0.95);
  %LOCAL errDF SEmult L U;
  DATA _NULL_;
    SET &overallANOVA;
    IF Source = "Error" THEN DO;
      CALL SYMPUTX('errDF', df);
      CALL SYMPUTX('SEmult', TINV(1-(1-&alpha)/2, df));
      END;
  RUN;
  %LET L = CIL%SYSEVALF(&alpha * 100);
  %LET U = CIU%SYSEVALF(&alpha * 100);
  %PUT Calculating %SYSEVALF(&alpha*100)% CI using df=&errDF,
       multiplier=&SEmult..;
  DATA &estimates;
    SET &estimates;
      &L = estimate - &SEmult * StdErr;
      &U = estimate + &SEmult * StdErr;
  RUN;
%MEND contrastCI;
```

Example:

```
TITLE "Creating contrasts dataset from ESTIMATE's";
PROC GLM DATA=drd4;
  CLASS sensPar DRD4;
  MODEL EBS = sensPar|DRD4;
  ESTIMATE "Un-sensitive: DRD4 absent-present"
    DRD4 1 -0.5 -0.5
    sensPar*DRD4 1 -0.5 -0.5  0 0 0   0 0 0;
  ESTIMATE "Medium Sensitive: DRD4 absent-present"
    DRD4 1 -0.5 -0.5
    sensPar*DRD4 0 0 0  1 -0.5 -0.5   0 0 0;
  ODS OUTPUT estimates=contrasts overallANOVA=OA;
RUN;
QUIT;

%contrastCI(contrasts, OA)
%contrastCI(contrasts, OA, alpha=0.99)
PROC PRINT DATA=contrasts;
  FORMAT Estimate StdErr CIL95 CIU95 CIL99 CIU99 6.2;
RUN;
```

Obs	Dependent	Parameter	Estimate
1	EBS	Un-sensitive: DRD4 absent-present	-5.49
2	EBS	Medium Sensitive: DRD4 absent-present	4.56

Obs	StdErr	tValue	Probt	CIL95	CIU95	CIL99	CIU99
1	1.08	-5.08	<.0001	-7.65	-3.32	-8.37	-2.60
2	1.12	4.09	0.0001	2.33	6.80	1.58	7.54