CMU MSP 36-601: SAS Macros 4

Howard Seltman, Apr 10, 2019

1) Debugging SAS Macros

a. SYMBOLGEN, MLOGIC, and MPRINT

```
%MACRO contrastCI(estimates, overallANOVA, level=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-&level)/2, df));
    END;
  RUN;
  %LET L = CIL%SYSEVALF(&level * 100);
  %LET U = CIU%SYSEVALF(&level * 100);
  %PUT Calculating %SYSEVALF(&level*100)% CI using df=&errDF,
       multiplier=&SEmult..;
  DATA &estimates;
    SET &estimates;
    &L = estimate - &SEmult * StdErr;
    &U = estimate + &SEmult * StdErr;
  RUN;
%MEND contrastCI;

DATA drd4;
  INFILE "data/drd4.dat" FIRSTOBS=2;
  INPUT drd4 sensPar ebs;
RUN;
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;

OPTIONS SYMBOLGEN;
%contrastCI(contrasts,oa)
SYMBOLGEN: Macro variable OVERALLANOVA resolves to oa
SYMBOLGEN:  Macro variable LEVEL resolves to 0.95
NOTE: There were 3 observations read from the data set WORK.OA.
SYMBOLGEN:  Macro variable LEVEL resolves to 0.95
SYMBOLGEN:  Macro variable LEVEL resolves to 0.95
SYMBOLGEN:  Macro variable LEVEL resolves to 0.95
SYMBOLGEN:  Macro variable ERRDF resolves to 54
SYMBOLGEN:  Macro variable SEMULT resolves to 2.00488
Calculating 95% CI using df=54, multiplier=2.00488.
SYMBOLGEN:  Macro variable ESTIMATES resolves to contrasts
SYMBOLGEN:  Macro variable ESTIMATES resolves to contrasts
SYMBOLGEN:  Macro variable L resolves to CIL95
SYMBOLGEN:  Macro variable SEMULT resolves to 2.00488
SYMBOLGEN:  Macro variable U resolves to CIU95
SYMBOLGEN:  Macro variable SEMULT resolves to 2.00488
NOTE: There were 2 observations read from the data set WORK.CONTRASTS.
NOTE: The data set WORK.CONTRASTS has 2 observations and 8 variables.
OPTIONS NOSYMBOLGEN MLOGIC;
```

```
%contrastCI(contrasts, oa)
MLOGIC(CONTRASTCI):  Beginning execution.
MLOGIC(CONTRASTCI):  Parameter ESTIMATES has value contrasts
MLOGIC(CONTRASTCI):  Parameter OVERALLANOVA has value oa
MLOGIC(CONTRASTCI):  Parameter LEVEL has value 0.95
MLOGIC(CONTRASTCI):  %LOCAL  ERRDF SEMULT L U

NOTE: There were 3 observations read from the data set WORK.OA.

MLOGIC(CONTRASTCI):  %LET (variable name is L)
MLOGIC(CONTRASTCI):  %LET (variable name is U)
MLOGIC(CONTRASTCI):  %PUT Calculating %SYSEVALF(&level*100)% CI using
                          df=&errDF, multiplier=&SEmult..
Calculating 95% CI using df=54, multiplier=2.00488.

NOTE: There were 2 observations read from the data set WORK.CONTRASTS.
NOTE: The data set WORK.CONTRASTS has 2 observations and 8 variables.

MLOGIC(CONTRASTCI):  Ending execution.



OPTIONS NOMLOGIC MPRINT;
%contrastCI(contrasts, oa)
MPRINT(CONTRASTCI):   DATA _NULL_;
MPRINT(CONTRASTCI):   SET oa;
MPRINT(CONTRASTCI):   IF Source = "Error" THEN DO;
MPRINT(CONTRASTCI):   CALL SYMPUT('errDF', TRIM(LEFT(PUT(df,8.))));
MPRINT(CONTRASTCI):   CALL SYMPUT('SEmult',
                        TRIM(LEFT(PUT(TINV(1-(1-0.95)/2,df),8.5))));
MPRINT(CONTRASTCI):   END;
MPRINT(CONTRASTCI):   RUN;

NOTE: There were 3 observations read from the data set WORK.OA.

Calculating 95% CI using df=54, multiplier=2.00488.
MPRINT(CONTRASTCI):   DATA contrasts;
MPRINT(CONTRASTCI):   SET contrasts;
MPRINT(CONTRASTCI):   CIL95 = estimate - 2.00488 * StdErr;
MPRINT(CONTRASTCI):   CIU95 = estimate + 2.00488 * StdErr;
MPRINT(CONTRASTCI):   RUN;

NOTE: There were 2 observations read from the data set WORK.CONTRASTS.
NOTE: The data set WORK.CONTRASTS has 2 observations and 8 variables.
```

Turn all debugging off with:

```
OPTIONS NOSYMBOLGEN NOMLOGIC NONMPRINT;
```

b. The Magic String (Unbalanced quotes etc.)

From Frank Poppe: "Sometimes you have been typing away in the Editor window of a SAS session, and only after submitting your code it becomes clear that you must have unbalanced quotes somewhere. But where, and was it a single or double quote? It is often difficult to correct that. Since long there has been a 'magic string', and it is still useful. Here it is:

```
)))));*';*";*/;quit;*%mend;
```

The nice thing is that it does not upset things if everything is okay, so it does not hurt to use it regularly."

2) **Macro loops and conditionals** (These ***only*** work inside macros, ***not*** in "open code".)

a. %IF/%THEN/%ELSE

i. Syntax:

```
%IF someCondition %THEN someAction;
[%ELSE someOtherAction;]
```

ii. Example: (Important: note HALF instead of "HALF")

```
/* given "pre" or "post" for "pre_or_post", "quarter" or "half" for "time",
   and a period number, make a report */
%MACRO report(pre_or_post, time, num);
  %LOCAL period;
  %IF %UPCASE(&time) = HALF %THEN %LET period = H&num;
  %ELSE %LET period = Q&num;
  TITLE "Report for &pre_or_post in &period";
  PROC MEANS DATA=prePost;
    VAR &pre_or_post&period;
  RUN;
%MEND report;
%report(post, Half, 2)
%report(pre, q, 3)
```

b. %**DO/%END** allows multiline actions

i. **Syntax:**

```
%IF someCondition %THEN %DO;
   someActionLine1;
   ...
   someActionLineN;
%END;
```

ii. **%DO/%END Example:**

```
%MACRO reportE(pre_or_post, time, num);
  %LOCAL period;
  %IF %UPCASE(&time) = HALF %THEN %LET period = H&num;
  %ELSE %DO;
    %IF %UPCASE(&time) = QUARTER %THEN %LET period = Q&num;
    %ELSE %DO;
      %PUT ERROR: time must be HALF or QUARTER;
      %ABORT;
    %END;
  %END;
  TITLE "Report for &pre_or_post-tax in &period";
  PROC MEANS DATA=prePost;
    VAR &pre_or_post&period;
  RUN;
%MEND reportE;
%reportE(post, Half, 2)    /* postH2 */
%reportE(pre, q, 3)        /* ERROR! */
%reportE(pre, quARteR,   3)  /* preQ3 */
```

c. %DO loops repeat actions groups based on a count

i. **Syntax:**

```
%DO myVar = myStartValue %TO myEndValue;
  someActionLine1;  (may reference &myVar)
  ...
  someActionLineN;  (may reference &myVar)
%END;

or

%DO UNTIL(expression);     (>=1 iteration; checked at bottom)
   ...
%END;
or

%DO WHILE(expression); (>=0 iterations; checked at top)
   ...
%END;
```

ii. %DO loop example:

```
%LET wd = /folders/myfolders/36602;
LIBNAME data "&wd/data";
DATA data.FY2016;
  INPUT expenses$ @@;
  DATALINES;
  food entertain food mortgage entertain food taxes
RUN;

DATA data.FY2017;
  INPUT expenses$ @@;
  DATALINES;
  utilities food entertain utilities taxes food
  mortgage food taxes
RUN;

/* A convenient word counting macro */
%MACRO countw(text);
  %SYSFUNC(COUNTW(&text, ' '))
%MEND;

/* A multi-year report macro */
/* Input: space separated years corresponding to
   data sets FYyyyy
   Output: tabulations of expenses for each year
*/
%MACRO multiYearReport(years);
  %LOCAL i year;
  %DO i = 1 %TO %countw(&years);
    %LET year = %SCAN(&years, &i);
    TITLE "Report for &year (printed &sysdate9)";
    PROC FREQ DATA=data.FY&year;
      TABLES expenses;
    RUN;
  %END;
%MEND multiYearReport;
%multiYearReport(2016 2017)
```

3) Fixing the "TITLE restore problem"

a. A standard computing principle for functions or macros is "leave everything as you found it". E.g., in R, any function that uses `par(mfrow=newDim)`, `par(oma=newMar)`, etc. should use the paradigm: `oldpar=par(no.readonly=TRUE); on.exit(par(oldpar))` before changing the `par()`.

b. It is not easy to restore the TITLEs (and/or FOOTNOTEs) after setting new values in a macro. Here is a solution (taken from "BellevueBob" at http://stackoverflow.com/questions/18724118/store-current-sas-title-change-it-temporarily-and-restore-it-at-end-of-macro).
Note: CATT() removes trailing spaces, then concatenates.

```
TITLE "My title";
FOOTNOTE "Note this";
PROC PRINT DATA=SASHELP.VTITLE; RUN;
FOOTNOTE;
TITLE;

/* Macro to save all titles and footnotes (non-recursive) */
%MACRO saveTF(tempDSname = _old_titles);
  DATA &tempDSname;  /* hard-coded dataset name */
    SET SASHELP.VTITLE;  /* hard-coded dataset name */
  RUN;
%MEND saveTF;

/* Macro to restore all saved titles and footnotes */
%MACRO restoreTF(tempDSname = _old_titles);
  %IF NOT %SYSFUNC(EXIST(&tempDSname)) %THEN %DO;
    %PUT Attempt to restore TITLE without saving as &tempDSname;
    %ABORT;
  %END;
  DATA _NULL_;
    SET &tempDSname;
    IF type = "T" THEN DO;
      cmd = CATT("TITLE", number, ' "', TRIM(text), ' ";');
    END;
    ELSE DO;
      cmd = CATT("FOOTNOTE", number, ' "', TRIM(text), ' ";');
    END;
    CALL EXECUTE(cmd);  /* collect to execute after DATA step */
  RUN;
  /* Delete temporary dataset */
  PROC DATASETS LIBRARY=WORK NOLIST;
    DELETE &tempDSname;
  RUN;
  QUIT;
%MEND restoreTF;
```

```
/* A test of saving and restoring titles */
%MACRO htest;
  %saveTF()
  TITLE "Contents of drd4";
  PROC CONTENTS DATA=drd4; RUN;
  %restoreTF()
%MEND;
TITLE "Howard’s data"; TITLE2 "Prepost dataset";
%htest
PROC CONTENTS DATA=Prepost; RUN;
```

4) Extended example: Improving addMonthData (from Macros3.pdf)

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

/* addMonthData() adds a month's data to the cumulative
   data.  If action=new, existing data for the month causes
   an error.  If action=add, old data for the month is
   allowed, and data from the current .dat file is added in.
   If action=replace, any current data is erased before
   adding in the new data.
*/
%MACRO addMonthData(month, year, cumDS, dataLoc, action=new,
                    summarize=yes) /
                    STORE SOURCE DES="Add month and summarize";
  /* Check for a valid action; */
  %IF &action^=new AND &action^=add AND &action^=replace %THEN %DO;
    %PUT Error using macro addMonthData(): 'action' must be 'new',
'add', or 'replace'.;
      %ABORT;
  %END;

  /* Drop data from old month if replacing; */
  %IF &action=replace %THEN
    %drop(month=&month, year=&year, cumDS=&cumDS);

  /* If action=new and old data for this month are on file, abort; */
  %IF &action=new %THEN %DO;
    %LOCAL oldData;
    %LET oldData = none;
    DATA _NULL_;
      SET &cumDS (WHERE=(month="&month" AND year=&year));
      CALL SYMPUT("oldData", month);
    RUN;
    %IF &oldData^=none %THEN %DO;
      %PUT Error running macro addMonthlyData(): action=new, but old
data exists;
      %ABORT;
    %END;
  %END;
```

```
/* Load data for this month into a temporary data set; */
  %LOCAL inname;
  %LET inname = "&dataLoc/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;

  /* Merge current data into old data; */
  DATA &cumDS;
    SET &cumDS E&month&year;
  RUN;

  /* Delete temporary data set; */
  PROC DATASETS NOLIST LIBRARY=WORK;
    DELETE E&month&year;
  RUN;
  QUIT;

  /* Run all summary reports if summarize=yes; */
  %IF &summarize^=yes %THEN %RETURN;
  %showAll(year=&year, cumDS=&cumDS);
%MEND addMonthData;


LIBNAME data "&wd/data";
%addMonthData(month=Jan, year=2017, cumDS=data.expenses,
              dataLoc=&wd, action=replace)
%addMonthData(month=Jan, year=2017, cumDS=data.expenses,
              dataLoc=&wd, action=new)
%addMonthData(month=Jan, year=2017, cumDS=data.expenses,
              dataLoc=&wd, action=add)
%addMonthData(month=Jan, year=2017, cumDS=data.expenses,
              dataLoc=&wd, action=replace, summarize=no)
%showAll(year=2017, cumDS=data.expenses)

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


%COPY addMonthData / SOURCE;
```

5) **Tips from SAS: "**Developing Bug-free Macros"

- The names in the %MACRO and %MEND statements match, and there is a %MEND for each %MACRO.
- The number of %DO statements matches the number of %END statements.
- %TO values for iterative %DO statements exist and are appropriate.
- All statements end with semicolons.
- Comments begin and end correctly.
- Macro variable references begin with & and macro statements begin with % .
- Macro variables created by CALL SYMPUT are not referenced in the same DATA step in which they are created.
- Statements that execute immediately (such as %LET) are not part of conditional DATA step logic.
- Single quotation marks are not used around macro variable references (such as in TITLE or FILENAME statements). When used in quoted strings, macro variable references resolve only in strings marked with double quotation marks.
- Macro variables, %GOTO labels, and macro names do not conflict with reserved macro keywords.