MORE COMPLEX DATA STRUCTURES IN SAS MACRO: MODELING ARRAYS OF RECORDS
Irene Mendelson
Pharma Research Division, Hoffmann-La Roche Inc.

ABSTRACT

SAS macro is similar to more traditional programming languages but has significant differences. At first glance, it has no real capability for complex data structures beyond arrays of macro variables, e.g. &VAR& as a shorthand for &VAR1 to &VARn where I ranges from 1 to n. But, with a little discipline, arrays of records can be constructed by making use of structured dataset variables and the ability to pass information from SAS datasets to macro variables. This technique will be discussed within the context of a program designed to create a composite graph called a control chart by superimposing control charts for each individual lot number. The user specifies the lots desired on the composite chart in one SAS dataset. Using a series of macros, the program then builds the datasets required from the master dataset, creates each chart, superimposes them, and even cleans up intermediary datasets no longer required.

This paper will state the programming problem and illustrate how various SAS macro techniques help to solve it. The specific techniques discussed include using single macro variables, using single macro variables whose values are assigned from SAS dataset values, and using macro variable arrays and records whose values are also assigned from SAS dataset values.

THE PROBLEM

A client would like a composite control chart to illustrate how all of the lots of interest conform to overall mean, upper control limits, and lower control limits on the variable weight. See figure 1 for the desired chart. The programming details must take into account that the program may be re-run with a different number of lots and that each lot may not have data for all time periods, including time periods at the endpoints of the chart.

SAS/QC offers the Shewhart control chart, but it is designed only for displaying information about one lot [1]. If the control charts for each lot could be produced and properly dimensioned, then they could be superimposed one on top of the other to produce the composite control chart desired.

THE OVERALL SOLUTION

In solving programming problems, a good first step is to frame it in terms of basic English sentences more formally called pseudocode, written below in *italics*. For the problem above, this can be translated as:

**Step 1**
Get user input for product code, final graph parameters, etc.

**Step 2**
Create lot master file with all time points for all lots to produce other chart parameters, including the sample mean or MU, three times the sample standard deviation or SIGMA, and time points for the horizontal axis.

**Step 3**
Create the individual lot files with time points found in all lots.

**Step 4**

![Figure 1](image-url)
Create the individual control charts for each lot.

Step 5
Combine all charts into one composite chart.

This solution suggests the use of loops to create each individual lot file and each individual control chart. The use of SAS macro is also suggested for two reasons. One, the same code with a different dataset name can create the control chart for each lot. And, two, the SAS dataset values for sample mean and for three times the sample standard deviation will need to be passed to the SAS procedure code to create the control charts.

TECHNIQUE I: SINGLE VARIABLES

The simplest use of SAS macro variables is through the assignment statement, the %LET statement. Applying this to our programming problem above, this type of statement can be a good way to request input from the user on product code and the three parameters for the control chart that cannot be easily calculated from the data. In this way, all user-supplied single values are collected at the top of the program and can be easily changed using a system editor.

The corresponding code fragment from the SAS program:

\[
\begin{align*}
\text{%LET PCODE} & = \text{xxx} ; \\
\text{%LET XMAX} & = 340 ; \\
\text{%LET XMIN} & = 320 ; \\
\text{%LET XINC} & = 5 ;
\end{align*}
\]

Note that &PCODE will select the product code while the other variables define the range and tick mark interval on the x-axis or vertical axis for the composite control chart.

TECHNIQUE II: SINGLE VALUES FROM DATA

Often, SAS datasets contain values that should be passed into SAS macro variables. This is done through the CALL SYMPUT function. In our programming problem above, the overall sample mean and overall sample standard deviation are produced from the MEANS procedure in an output dataset called MEANW. This output dataset has only one observation. For the purposes of producing the control charts, we would like to have the overall mean in the macro variable &XBAR and three times the overall sample standard deviation in the macro variable &SBAR.

The corresponding code fragment from the SAS program to produce these two macro variables then looks like:

\[
\begin{align*}
\text{DATA} \ _\text{NULL} ; \\
& \text{SET MEANW} ; \\
& \text{SBAR} = \text{SBAR} * 3 ; \\
& \text{CALL SYMPUT('XBAR',XBAR)} ; \\
& \text{CALL SYMPUT('SBAR',SBAR)} ; \\
& \text{RUN} ;
\end{align*}
\]

Note that the _NULL_ dataset is used since we are interested in producing macro variables and not SAS datasets. Note also that to avoid confusion we are using the same names for the macro variables as for the original SAS dataset variables. We could have changed the SAS macro variable names by changing the names within the quotes in the CALL SYMPUT lines.

TECHNIQUE III: ARRAYS FROM DATA

The motivation for constructing arrays of macro variables from data comes from refining step 3 of our original pseudocode, namely the part of the program to create the individual lot files. A small wrinkle is added by the requirement that the name of the individual lot file contain the lot number. This pseudocode can be refined as follows:

Step 3, pass 1
For each lot,
\[
\begin{align*}
& \text{Look at the master lot file.} \\
& \text{If the master lot number matches the individual lot number,} \\
& \text{Then output to the individual lot file.}
\end{align*}
\]

Step 3, pass 2
\[
\begin{align*}
\text{DATA} \ \text{lot[1]} \ \text{lot[2]} \ldots \ \text{lot[n]} ; \\
& \text{SET SORTED} ;
\end{align*}
\]
If we could construct the lot number and lot arrays, then we could use two different macro DO loops running from 1 to the number of lots. (An efficient way to create a macro variable &NUMLOTS to store the number of lots is shown in the SAS code appendix.) The SAS code with a few pieces missing would then look like the following:

```
DATA
	%DO I=1 %TO &NUMLOTS ;
	  (something goes here)
	%END ;
	%DO I=1 %TO &NUMLOTS ;
	  IF LOT = &??
	    THEN OUTPUT &?? ;
	%END ;
```

We can also gain some insight into how to construct the lot number and lot arrays by looking at the ultimate SAS code for our three lot example after all macro references have been resolved:

```
DATA LOT111 LOT222 LOT333 ;
  SET SORTED ;
  IF LOT = '111' THEN OUTPUT LOT111 ;
  IF LOT = '222' THEN OUTPUT LOT222 ;
  IF LOT = '333' THEN OUTPUT LOT333 ;
RUN ;
```

Looking into our bag of tricks, people have used macro variables imbedded in other macro variables to simulate an array index. For example, &NUMI for I = 1,2,3 resolves to &NUM1, &NUM2, and &NUM3. The trick is somehow to assign 111 to &NUM1, 222 to &NUM2, and 333 to &NUM3. The key is to use a dataset called LOTS that contains a variable LOT with those values in it and the CALL SYMPUT function previously discussed. The SAS code to set up this dataset and its associated macro variables is shown below:

```
DATA LOTS ;
  LENGTH LOTNUM $ 4 LOTNAME $ 7 ;
  INPUT LOTNUM $ 1-4 ;
  LOTNAME = 'LOT' || LOTNUM ;
  LOTINDEX = _N_ ;
CARDS ;
  111
  222
  333
RUN ;
DATA _NULL_ ;
  SET LOTS ;
  NUM = 'NUM' || LEFT(LOTINDEX) ;
  NAME = 'LOT' || LEFT(LOTINDEX) ;
  CALL SYMPUT(NUM,LOTNUM) ;
  CALL SYMPUT(NAME,LOTNAME) ;
RUN ;
```

Here the SAS dataset variable NUM changes its value each time the CALL SYMPUT function is evaluated to create the macro variable array &NUMI. Similarly, the SAS dataset variable NAME changes its value to create the macro variable array &LOTI. In this way, we have created a SAS macro equivalent to the record data structure, namely each lot is described by related information elements, specifically lot number, associated dataset name, and other elements shown in the SAS code appendix.

Now the SAS code fragment for Step 3 of our original program can be expanded below:

```
DATA
	%DO I=1 %TO &NUMLOTS ;
	  %LET DATAFILE = LOT&I ;
	  &&DATAFILE
	%END ;
; 
SET SORTED ;
%DO I=1 %TO &NUMLOTS ;
  %LET LOTNUM = NUM&I ;
  %LET LOTNAME = LOT&I ;
  IF LOT = &LOTNUM
    THEN OUTPUT &&LOTNAME;
%END ;
```
Another example of using this macro variable array \&\&NUM\&I is to create the title line for the graph that lists all of the lots displayed on the composite control chart. Looping over the \%LET assignment statement for all lots will do the trick as follows:

\%LET TITLE3 = LOTS ;
\%DO I=1 %TO \&NUMLOTS ;
  \%LET TITLE3 = TITLE3 &\&NUM\&I ;
\%END ;

The first time the loop is executed, for I=1, \&TITLE3 evaluates to LOTS 111. This means that the \%LET statement inside the loop gives TITLE3 = LOTS 111. The second time the loop is executed, for I=2, \&TITLE3 evaluates to LOTS 111 and \&\&NUM\&I evaluates to \&NUM2 or 222. This means that the \%LET statement inside the loop this time gives TITLE3 = LOTS 111 222. The third and final time the loop is executed, for I=3, \&TITLE3 evaluates to LOTS 111 222 and \&\&NUM\&I evaluates to \&NUM3 or 333. The end result is TITLE3 = LOTS 111 222 333, which is exactly what we want and what we see in figure 1.

CONCLUSION

Using the SAS DATA step, the data structures in SAS macro can be extended beyond single macro variables into arrays and records of macro variables. The SAS DATA step function CALL SYMPUT is the key, not only for passing values from a SAS dataset into a SAS macro variable but also for passing values from a SAS dataset into arrays of SAS macro variables. These arrays of SAS macro variables can hold related information or records. All of these data structures have been successfully used in this paper to solve a specific programming problem, namely to construct a composite control chart to show how weight varies for different lots of a pharmaceutical compound.

REFERENCES


ACKNOWLEDGEMENTS

I would like to acknowledge Scott Stoller for introducing me to this way of looking at SAS macro variable arrays and records and Beverly DeNobriga for her collaboration in producing the composite control chart program.

SAS CODE APPENDIX

```
options nodate nonumber ls=18
     ps=8;
...
%let pcode = xxx;
%let xmax = 340;
%let xmin = 320;
%let xinc = 5;
...
%let pcode = xxx;
%let xmax = 340;
%let xmin = 320;
%let xinc = 5;
...
```

```
proc sort data=sasuser.&pcode out=sorted ;
by lot time ;
run ;
proc sort data=sorted out=sorted2 ;
by time ;
run ;
proc means data=sorted2 noprint ;
by time ;
var weight;
output out=lotmast mean=meanw ;
run ;
```
length lotnum $ 4 lotname $ 7 lotname2 $ 8
lreplay $ 5 ;
input lotnum $ 1-4 ;
lotname = 'lot' || lotnum ;
lotname2 = lotname || 'm' ;
lotindex = _n _ ;
lreplay = '1' || left( lotindex*2 - 1 ) ;
cards ;
1111
2222
3333
;
run ;

* Create numlots, lot1, lot2, etc. macro variables *

* Create title3 macro variable with list of lots graphed in it *
%macro t3mac ;
%global title3 ;
%local i ;
%let title3 = LOTS ;
%do i=1 %to &numlots ;
%let title3 = &title3 &&NUM&i ;
%end ;
%mend t3mac ;

%t3mac ;
run ;

* Create macro to spin out individual lots *
%macro datamac ;
%local i lotnum lotname ;
data
do i=1 %to &numlots ;
%let datafile = LOT &i ;
%let datafile = LOTM&i ;
&datafile
%end ;
%mend datamac ;
%datamac ;
run ;

* Define macro to merge lot master file *;
* with each lot file *;
%macro alltimes( datai, datao ) ;
data &datao ;
merge lotmast &datai ;
by time ;
drop meanw ;
run ;
%mend alltimes ;

%macro mergemac ;
%local i datai datao ;
do i=1 %to &numlots ;
%let datai = LOT &i ;
%let datao = LOTM&i ;
%alltimes( &datai, &datao ) ;
%end ;
%mend mergemac ;

%mergemac ;

proc means mean range data=sorted noprint ;
by lot time ;
var weight ;
output out=vall mean=weight range=range ;
run ;

* Clean up lot master and other unneeded datasets *
%macro cleanmac( datacat ) ;
%local i ;
proc datasets library=work ;
delete lotmast sorted ;
run ;
%mend cleanmac ;
%let datacat = LOT ;
%cleanmac( &datacat ) ;

* Set up macro variables for control chart *;
title 'overall mean max min of sample means';
proc means mean max min std data=vali;
var weight;
output out=meanw mean=xbar max=xmax min=xmin
   std=sbar;
run;
data _null_;
set meanw;
sbar = sbar * 3;
call symput('xbar' "mar) ;
call symput('sbar' ,&bar) ;
run;

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Step 4: Create the individual control charts for each lot *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
%let device = cgmwp;
filename gsasfile 'nesug2.gsr;
goptions device = &device nodisplay noprompt;
G/acro cchart ( datafile ) ;
proc shewhart data = &datafile graphics
gout=tempcat;  
xchart weight*time = "*" /
   noconnect separate bilevel
   vaxis = &xmin to &xmax by &xinc
   mu = &xbar sigma = &sbar;
run;
%macro cchart ( datafile );
  proc shewhart data = &datafile graphics
gout=tempcat;
xchart weight*time = "*" /
   noconnect separate bilevel
   vaxis = &xmin to &xmax by &xinc
   mu = &xbar sigma = &sbar;
run;
%mend cchart ;
%macro chartmac ( datacat ) ;
  %local i;
  %do i=2 %to &numlots ;
    %let datafile = &datacat&i ;
    %cchart ( &datafile ) ;
  %end;
%mend chartmac ;
TITLE1 F=SWISS 'X BAR CHART';
TITLE2 h=1 pct "FILLING PROCESS FOR PRODUCT CODE &pcode";
TITLE3 h=1 pct "&title3";
TITLE4 H=1.0 IN A=-90 ;
%let datafile = &lotm1 ;
%cchart ( &datafile ) ;

%let datafile = &lotm1 ;
%cchart ( &datafile ) ;

%let datacat = LOTM ;
%chartmac ( &datacat ) ;

* Clean up more temporary work datasets ;
proc datasets library=work;
delete sorted2;
run;

%let datacat = LOTM ;
%cleanmac ( &datacat ) ;

* Combine all graphs and replay into graphics stream file *
%goptions cback=white colors=(black) display
   device=&device nocharacters nocell npolygonfill
   ifactor=4 gacess=gsasfile vpos=43 hpos=78 ;
%macro graphall( device ) ;
%local i screen;
proc grepplay igout=tempcat nofs;
device &device;
templ temp;
tdef leftrgt
   1/llx=0 lly=0
   ulx=0 uly=100
   urx=100 ury=100
   lrx=100 lry=0;
template leftrgt;
treplay
%do i=1 %to &numlots ;
    %let screen = Replay&i ;
    %&screen
%end;
%mend graphall ;
%graphall ( &device ) ;

AUTHOR CONTACT INFORMATION
Irene Mendelson
Hoffmann-La Roche Inc.
340 Kingsland Street
Nutley, NJ 07110
(201) 235-4334
mendelsi@nisd0.dnet.roche.com

NESUG '92 Proceedings