PhUSE 2006

Paper CC09

CALL EXECUTE and graphs

Shafi Chowdhury, Shafi Consultancy Limited, London, U.K
Daniel Boisvert, Genzyme, Cambridge, U.S.A

ABSTRACT

"Please change this so treatment groups have the same symbol in the different plots" is a common phrase from statisticians reviewing plots. We then go away, look at what treatment groups there are, which ones are missing from a particular plot, and proceed to change all our SYMBOL statements for the various plots. We repeat this process for each new study. This paper will show how we can solve this problem by creating SYMBOL statements dynamically using CALL EXECUTE. It will show how the programs can always assign the same symbols for each treatment groups regardless of whether all treatment groups are present or not in the plot.

INTRODUCTION

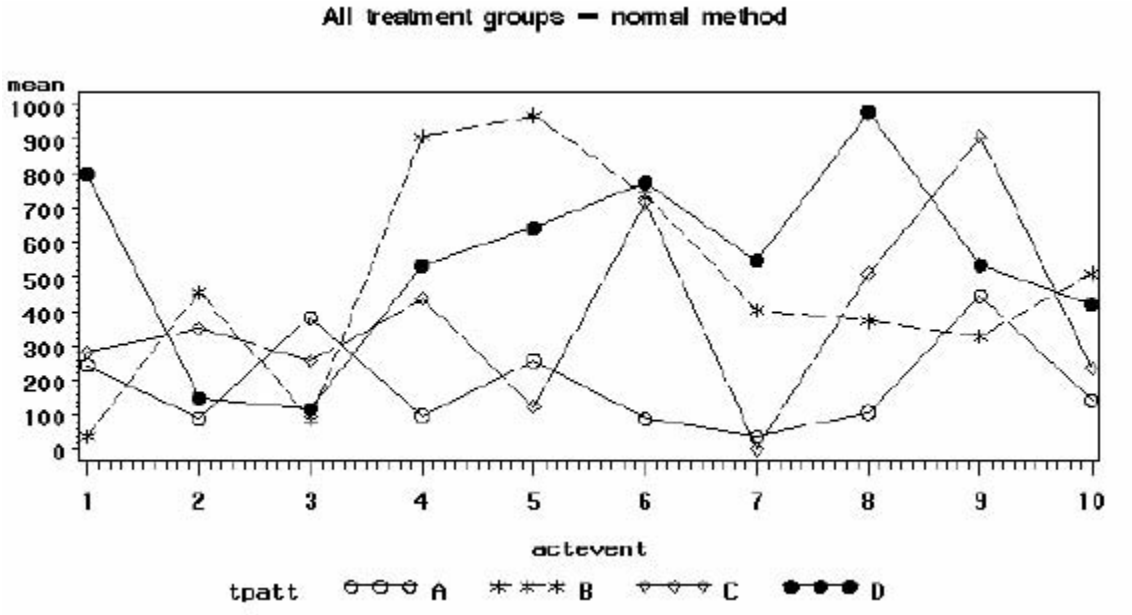
One issue which comes up often in plots is that the symbols used are not always consistent for treatment groups between different plots. This arises when a treatment group is missing from a particular plot, usually because there were no eligible patients from that treatment. SAS assigns symbols sequentially, and so if there are four treatment groups and one plot had no data for the third treatment group, the fourth treatment will be assigned the third SYMBOL statement, creating inconsistency with plots containing all four treatment groups. It is much better for those reading the trial report if each treatment always had the same symbol, thereby reducing the risk of misinterpretation.

NORMAL METHOD

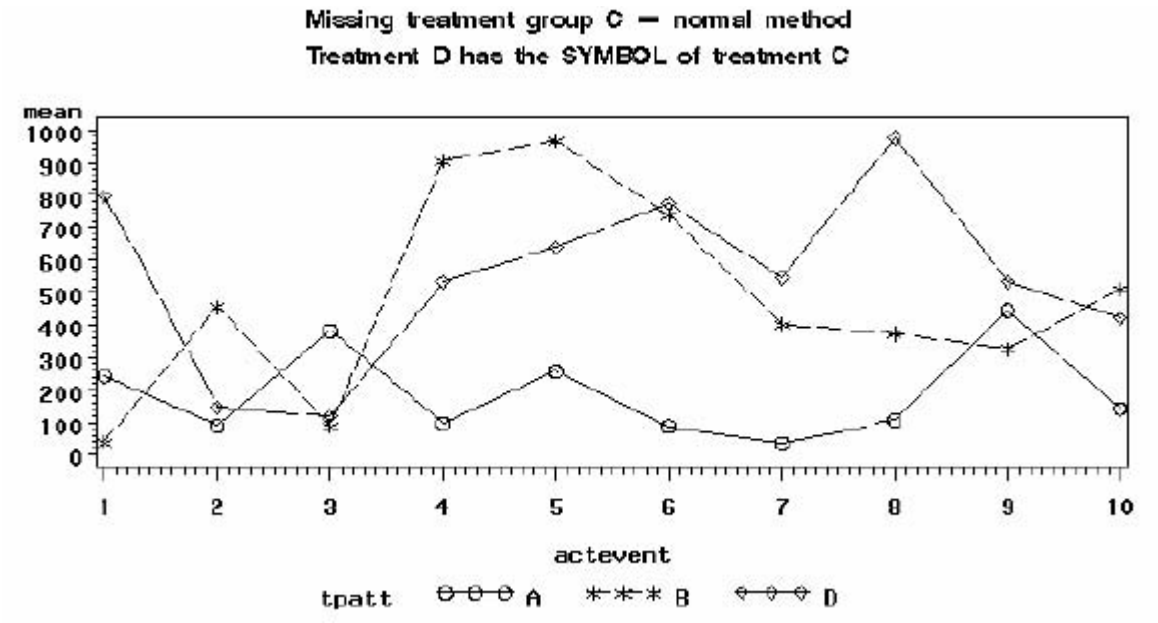
Assume we have to produce a series of graphs of mean over time by treatment (mean*actevent=tpatt). This sounds very straight forward and it is likely that we will write a piece of code which looks something like this.

```
SYMBOL1 I=J V=CIRCLE L=7 C=BLACK;
SYMBOL2 I=J V=STAR L=3 C=BLACK;
SYMBOL3 I=J V=DIAMOND L=5 C=BLACK;
SYMBOL4 I=J V=DOT L=1 C=BLACK;

TITLE H=1 F=SWISSB 'All treatment groups - normal method';
PROC GPLOT DATA=indata;
  PLOT mean*actevent=tpatt;
RUN;
QUIT;
TITLE;
```



PROBLEM WHEN ONE TREATMENT GROUP IS MISSING:



In this example Treatment D has taken on the symbol of diamond whereas in the previous example Treatment D was a solid dot. This occurs because the symbol statement is assigned sequentially and is not based directly on the treatment group. Because D comes up third, in the data set, SYMBOL3 (diamond) is assigned to it. This can cause confusion to the reader.

SOLUTION

To remedy this situation we need to connect the symbol statement to the value of the treatment. Start by creating macro variables for the treatments. The value of the macro variable is the SYMBOL statement without the keyword SYMBOL. Here we need the name of the treatment (a,b,c,d) to be in the name of its macro variable.

```
%LET TRTa = I=J V=CIRCLE  L=7 C=BLACK;
%LET TRTb = I=J V=STAR    L=3 C=BLACK;
%LET TRTc = I=J V=DIAMOND L=5 C=BLACK;
%LET TRTd = I=J V=DOT     L=1 C=BLACK;
```

Then create a data set which contains all of the values of the treatment groups variable (TPATT) which are going to be present in the graph.

```
PROC SORT DATA=alltrt OUT=_trts(KEEP=tpatt) NODUPKEY
  BY tpatt;
RUN;
```

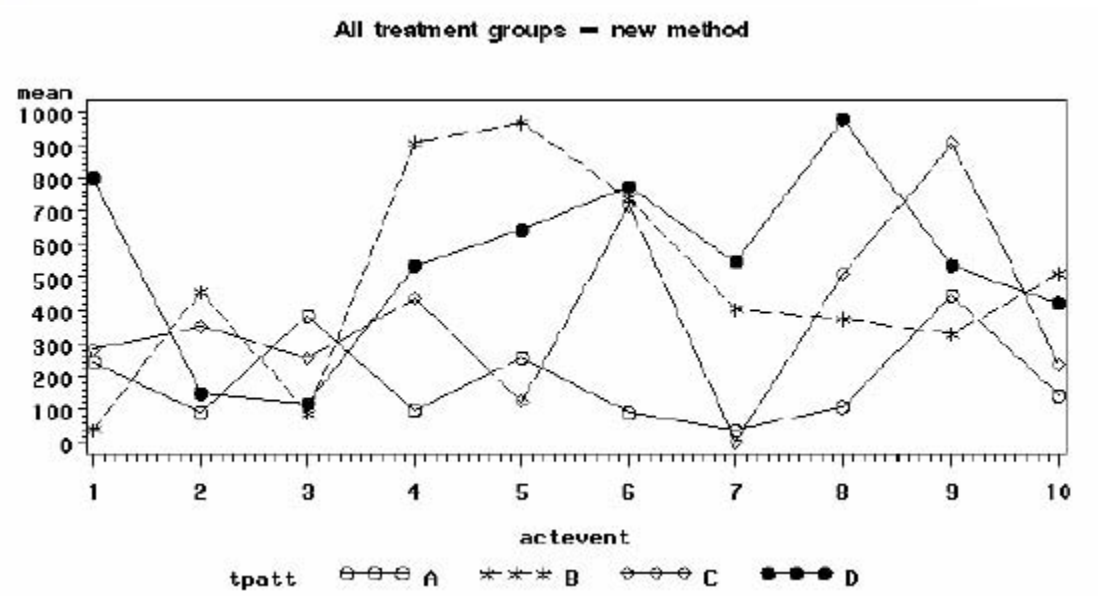
Using CALL EXECUTE we assign the symbols to their treatments.

```
DATA _NULL_;
  SET _trts;
  CALL EXECUTE ('SYMBOL'||LEFT(PUT(_N_,BEST8.))||' &trt'||LEFT(tpatt)||';');
RUN;
```

The following lines of code are generated.

```
NOTE: There were 4 observations read from the data set WORK._TRTS.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.01 seconds


NOTE: CALL EXECUTE generated line.
1       + SYMBOL1        I=J   V=CIRCLE    L=7   C=BLACK;
2       + SYMBOL2        I=J   V=STAR      L=3   C=BLACK;
3       + SYMBOL3        I=J   V=DIAMOND   L=5   C=BLACK;
4       + SYMBOL4        I=J   V=DOT       L=1   C=BLACK;
```



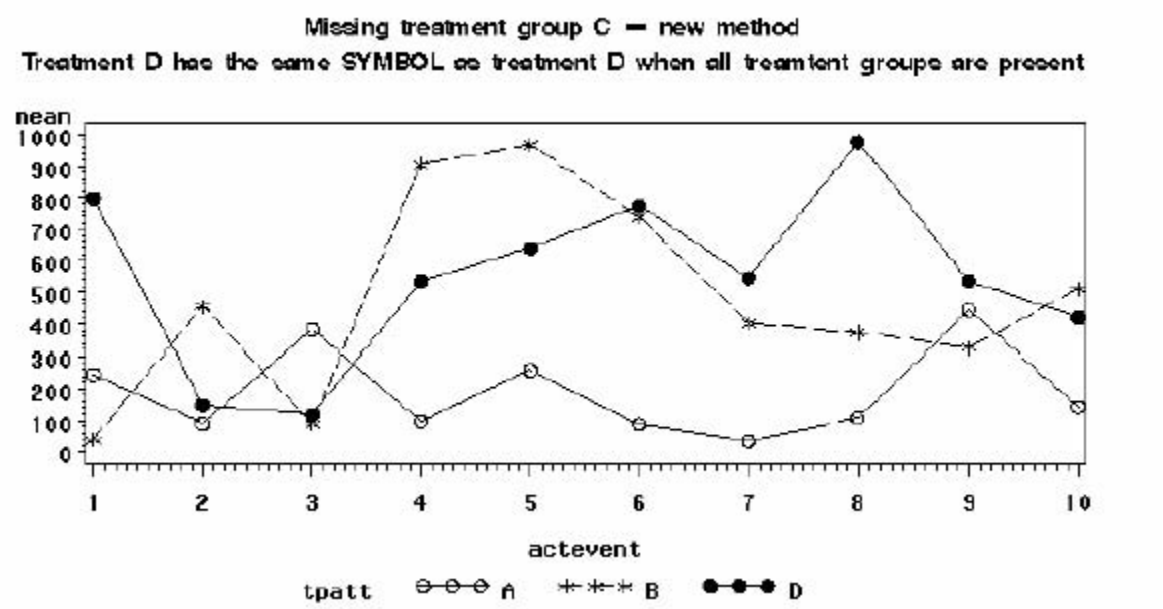
Now when one treatment groups is omitted and the same code is used, the following SYMBOL statements are produced.

```
20731   DATA _NULL_;
20732     SET _trts;
20733     CALL EXECUTE ('SYMBOL'||LEFT(PUT(_N_,BEST8.))||' &trt'||LEFT(tpatt)||'
20733 ! ;');
20734   RUN;

NOTE: There were 3 observations read from the data set WORK._TRTS.
NOTE: DATA statement used:
      real time           0.00 seconds
      cpu time            0.00 seconds


NOTE: CALL EXECUTE generated line.
1       + SYMBOL1        I=J   V=CIRCLE    L=7   C=BLACK;
2       + SYMBOL2        I=J   V=STAR      L=3   C=BLACK;
3       + SYMBOL3        I=J   V=DOT       L=1   C=BLACK;
```

Notice that SYMBOL3 is now V=DOT L=1, instead of V=DIAMOND L=5 as it was in the previous example. This code then creates plots with consistent symbols, i.e. Treatment D always has the same symbol.



CONCLUSION

CALL EXECUTE can be used in many different situations to write SAS code dynamically. It uses datasets to automatically change values which would otherwise have to be changed manually. This is just one example of how it can be used to remove the risk of inconsistent symbols for treatment groups.

CONTACT INFORMATION

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

Shafi Chowdhury
Shafi Consultancy Limited

Email: shafi@shaficonsultancy.com
www.shaficonsultancy.com

Daniel Boisvert
Email: DJBoisvert@gmail.com