Is the Legend in your SAS/GRAPH® Output Telling the Right Story?

Justina M. Flavin, Pfizer Global Research & Development, La Jolla Laboratories, San Diego, CA
Arthur L. Carpenter, California Occidental Consultants, Oceanside, CA

ABSTRACT

When generating plots using the GPLOT procedure, a third variable, such as a treatment or dose group, is often used as a grouping variable. GPLOT automatically uses the values of this variable in the LEGEND, but very often the designers of the plots will want particular symbols, line types, and/or colors to be specified for the display of specific values of this variable. Of course consistency is important, and if the program is subsequently run on a subset of the original data, some values of this discrete variable may be absent in the subset. This can result in an inconsistent representation (symbol, line type, color) of a particular value of the variable in a series of plots.

This paper will present macro and non-macro solutions for producing plots with consistent attributes for each distinct value of the grouping variable, showing how to maintain this consistency regardless of the subset used. The paper will also explain how to generate a legend which will either display all values that exist in the data or only those that appear in the subset.

KEYWORDS

GPLOT, LEGEND, symbols

INTRODUCTION

In the analysis of clinical trials data, output that includes only a subset of the entire study population is often required. Many times the subset does not contain all treatment or dose groups. For most non-graphical SAS output, a program can successfully be rerun without modification on a subset, and the absence of one of more of the treatment or dose groups does not affect the display of the data.

One notable exception to this occurs in using the GPLOT procedure to create plots of three variables and assigning symbol definitions to the values to the third variable. Since SAS assigns the symbol statements in sequential order, in rerunning a program on a subset of the data, the same treatment or dose group may be displayed using a different symbol, line type, and/or color in some or all of the plots. This lack of consistency across a series of plots can be a source of confusion and make treatment or dose group comparisons more difficult to observe.

THE PROBLEM

To illustrate the problem, consider creating a plot of some (mean) results over time, stratified by dose group. For the examples shown, the following data will be used.

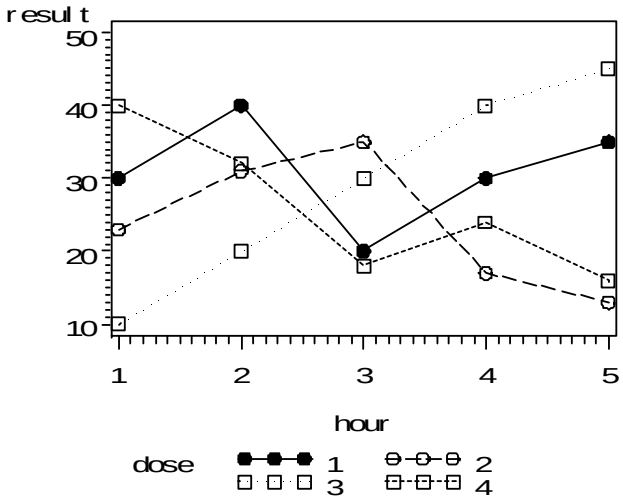
```
data drug;
  input dose hour result @@;
 datalines;
  3 1 10 3 2 20 2 1 23 3 3 30 3 4 40 3 5 45
  2 2 31 2 3 35 2 4 17 4 1 40 4 2 32 4 3 18
  4 4 24 4 5 16 1 1 30 1 2 40 1 3 20 1 4 30
  1 5 35 2 5 13
run;
```

In these data, there are four dose groups taking on the values of 1 to 4.

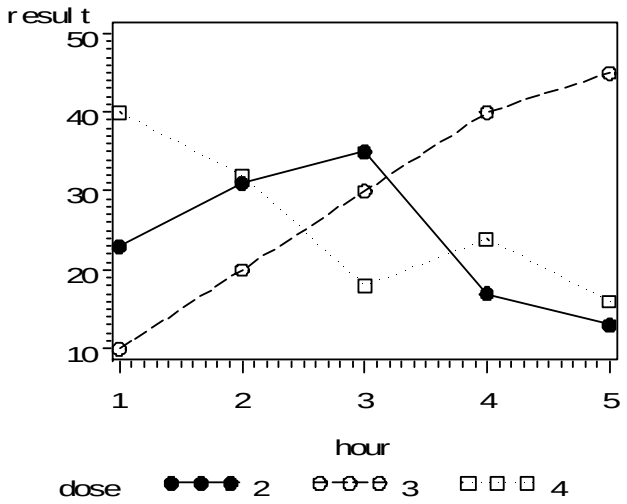
The symbol statements can be defined as follows:

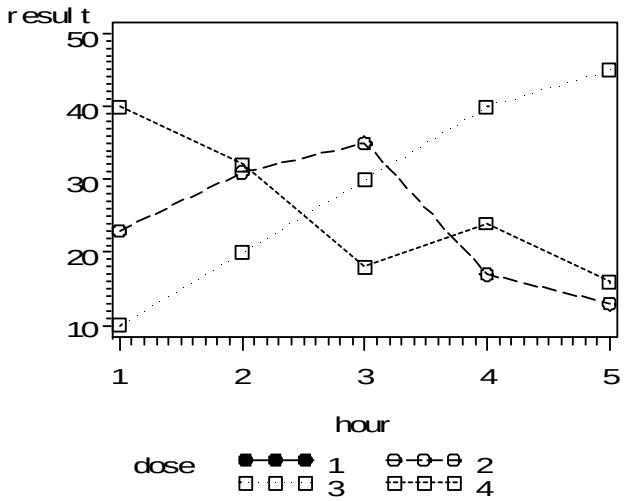
```
symbol1 v=dot i=j c=black l=1;
symbol2 v=circle i=j c=black l=3;
symbol3 v=square i=j c=black l=33;
symbol4 v=square i=j c=black l=2;

proc gplot data=drug;
  plot result*hour=dose;
run;
```

Resulting in the following output:



Now suppose the data are subset and the first dose group is absent from the subset. Rerunning the code produces the following graph:





In this graph, Dose Group 2 is represented by a solid line and a solid dot, which is the same symbol and line type that was assigned to Dose Group 1 in the previous graph.

There are several techniques that can be used to ensure that a given dose level will always be associated with a given SYMBOL statement, regardless of the data subset used.

SOLUTION 1

This solution creates a dummy data set with one unplottable observation per dose. When appended to the original data set that is to be plotted, the correct use of the symbol statements is guaranteed even though each SYMBOL statement will not have plotted points. Regardless of the subset used, the legend will be constant for all plots, displaying all dose groups that exist in the data, not just those that appear on the graph.

```
data dummy;
  do i=1 to 4;
    dose=i; result=.; hour=.; drop i; output;
  end;
run;

data drug2;
  set drug(where=(dose in (2,3,4))) dummy;
run;

symbol1 v=dot i=j c=black l=1;
symbol2 v=circle i=j c=black l=3;
symbol3 v=square i=j c=black l=33;
symbol4 v=square i=j c=black l=2;

proc gplot data=drug2;
  plot result*hour=dose;
run;
```

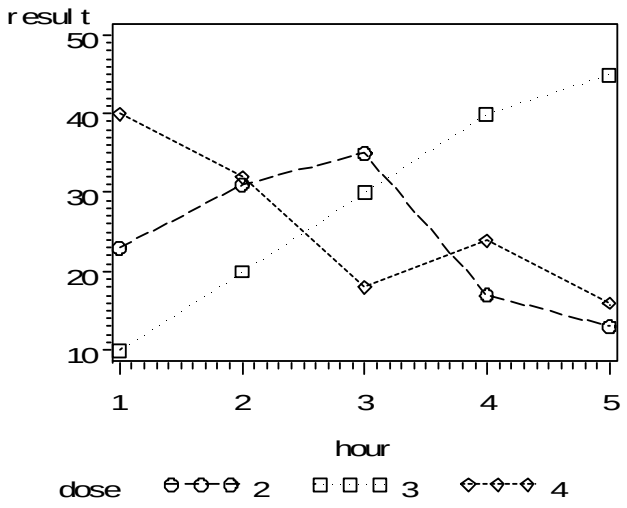
SOLUTION 2

This solution assigns the total number of dose groups that exist in the subset to a macro variable (&numdose). A list of dose values that exist in the subset is assigned to another macro variable (&doselist). The macro variables containing a list of the desired symbols (&symbols) and line types (&linetype) are created. Through repeated use of the %scan function, the symbol statements are generated dynamically within a macro.

With this solution, the legend will change from graph to graph, only displaying the dose groups that exist in the subset. However, the association between the dose group and its SYMBOL statement will be consistent.

```
data drug2;
  set drug;
  where dose in (2,3,4);
run;

proc sql noprint;
  select count (distinct dose) into: numdose
  from drug2;
  select distinct dose
         into: doselist separated by ' '
  from drug2;
quit;

%let symbols=(dot circle square diamond);
%let linetype=(1 3 33 2);

%macro makeit;
%do j=1 %to &numdose;
    symbol&j
      v=%scan(&symbols, %scan(&doselist, &j))
      i=j c=black
      l=%scan(&linetype, %scan(&doselist, &j));
%end;
%mend makeit;

%makeit;
```

```
proc gplot data=drug2;
  plot result*hour=dose;
run;
```



PLOTTING A THIRD VARIABLE THAT IS CHARACTER

In the previous example, the DOSE variable was numeric and ordered sequentially starting at 1. If the third variable is either a non-sequential numeric variable or a character variable, the aforementioned solutions can be modified to achieve the same results.

For these examples, assume that DOSE has the following four values: 1, b1, b2, c.

SOLUTION 1 REVISITED

The code to create the dummy data set in Solution 1 can be changed as follows:

```
data dummy;
  dose='1'; result=.; hour=.; output;
  dose='b1'; result=.; hour=.; output;
  dose='b2'; result=.; hour=.; output;
  dose='c'; result=.; hour=.; output;
run;
```

Or, to automate building the dummy data set, use the PROC SQL code that was shown in Solution 2 to generate a macro variable (&numdose) that holds the total number of dose groups in the population and the macro variable (&doselist) that contains a list of all possible values of DOSE. These macro variables are then subsequently used in the %makedummy macro which creates the dummy data set.

```
proc sql noprint;
  select count (distinct dose) into: numdose
  from drug;
  select distinct dose
         into: doselist separated by ' '
  from drug;
quit;
```

```
%macro makedummy;

  data dummy;
    length dose $%length(&doselist);

    %do j=1 %to &numdose;
      dose="%scan(&doselist, &j)";
      result=.; hour=.; output;
    %end;
  run;

%mend makedummy;

%makedummy;

data drug2;
  set drug(where=(dose in ('b1' 'b2' 'c')))
      dummy;
run;
```

Note that this solution requires that the dummy data set be constructed from a data set that contains all values of DOSE, not just those values that occur in the subset.

SOLUTION 2 REVISITED

This solution requires the use of the macro %REVSCAN, which returns the word number given the word itself. The macro %REVSCAN is the reverse of the %SCAN function, and is described in detail in the 2nd edition of *Carpenter's Complete Guide to the SAS Macro Language*. The code for %REVSCAN is included at the end of this paper. An additional macro variable (&alldoses) that contains the list of all dose groups is also required.

```
data drug2;
 set drug(where=(dose in ('b1' 'b2' 'c')));
run;

proc sql noprint;
  select count (distinct dose) into: numdose
  from drug2;
  select distinct dose
         into: doselist separated by ' '
  from drug2;
  select distinct dose
         into: alldoses separated by ' '
  from drug;
quit;

%let symbols=(dot circle square diamond);
%let linetype=(1 3 33 2);

%macro makeit;

%do j=1 %to &numdose;

%let
dose=%revscan(&alldoses,%scan(&doselist, &j));
```

```
symbol&j v=%scan(&symbols, &dose)
         i=j c=black
         l=%scan(&linetype, &dose);

%end;

%mend makeit;

%makeit;

proc gplot data=drug2;
  plot result*hour=dose;
run;
```

CONCLUSION

This paper has illustrated some ways of generating consistent graphical output when the same program is run on different subsets of the data. Which solution to use depends on the type of legend desired and whether the grouping variable is character or numeric. Unless a program is being written for one time use, adding the extra code to ensure consistency in the output is well worth the effort.

REFERENCES

Carpenter, Arthur L., *Carpenter's Complete Guide to the SAS Macro Language*, Cary, NC: SAS Institute Inc., 1998, 242 pp.

Carpenter, Arthur L., *Carpenter's Complete Guide to the SAS Macro Language, 2nd Edition*, Cary, NC: SAS Institute Inc., 2004.

TRADEMARK INFORMATION

APPENDIX – SUPPORT CODE

%REVSCAN

```
%macro revscan(list, word);

%local wcnt wnum;
%let wcnt=0;
%let wnum=0;

%* Determine the word number in a list of words;

%do %while
(%scan(&list,%eval(&wcnt+1),%str( )) ne %str());

%let wcnt = %eval(&wcnt+1);

%if
  %upcase(%scan(&list,&wcnt,%str()))
  =%upcase(&word)
%then %let wnum=&wcnt;
%end;

&wnum

%mend revscan;
```

CONTACT INFORMATION

Justina M. Flavin
Pfizer Global Research & Development, La Jolla Laboratories
11085 Torreyana Road
San Diego, CA 92121
(858) 622-7376
justina.flavin@pfizer.com

Justina Flavin is a Senior Clinical Programmer/Analyst at Pfizer Global Research & Development in San Diego. She has been developing software in the pharmaceutical industry for over eleven years. Justina served as the conference chair of Pharma*SUG* '99, was the facilities and seminar course coordinator for WUSS 2000, and serves on the EC of SANDS, the San Diego SAS User Group. She has a B.A. in Applied Mathematics from the University of California, San Diego.

Art Carpenter
California Occidental Consultants
PO Box 6199
Oceanside, CA 92058-6199
(760) 945-0613
art@caloxy.com
www.caloxy.com

Art Carpenter's publications list includes two chapters in *Reporting from the Field*, three books and over four dozen papers and posters presented at SUGI, NESUG, WUSS, and Pharma*SUG*. Art has been using SAS since 1976 and has served in leadership roles in various local, regional, and national user groups. Art is a SAS Certified Professional™. Through California Occidental Consultants he teaches SAS courses and provides contract SAS programming support nationwide.