

Macros, Annotate and SAS Graph the Easy Way

Michael Keohan, E-Z Solutions, Inc., E. Hanover, NJ

Abstract

Analyzing multi-center clinical trials data can require combining a variety of statistics onto one graph. For example, a plot of the mean percent change from baseline by treatment is more meaningful if the standard error of the mean (SEM) and N are included on the plot. This paper demonstrates how to design code to make this process easy. Using the annotate facility, passing GLM statistics to SAS graph and writing generic code to be used across several analysis variables are some of the topics discussed.

Assumptions

You have a series of SAS data sets which contain clinical trials information. For illustration purposes, let us assume the data sets which you are going to process have the variables: visit, center, trt and avar (analysis variable measured in % change from baseline).

The data sets can contain analysis variables ranging from lab tests to a primary efficacy variable, which are measured as percent change from baseline. Also, there are two studies which will require similar graphics output; the only difference is one study has 3 treatment groups and the second study has four treatment groups. The objective is to generate a plot of mean percent change from baseline by treatment.

Program 1

We will use proc univariate OUT= to generate the N and MEAN statistics to UNIOUT;

```
proc univariate data=effdat noprint;
  by visit trt;
  var uvar;
  output out=uniout n=n mean=mn;
```

```
proc print data=uniout noobs;
  title "data from uniout";
```

Print #1

data from uniout

OBS	VISIT	TRT	N	MN
1	2.0	1	68	0.07177
2	2.0	2	68	0.88175
3	2.0	3	62	-0.40878
4	3.0	4	72	0.08341
5	3.0	1	60	-0.10165
6	3.0	2	50	2.09189

At this point we are ready to generate plot #1. First we need to specify the gsfname. The gsfname reference name tells SAS to send all graphics output to the specified file name.

```
filename gsfname "plot1 lj600ps a";
```

Next the goptions are coded to specify horizontal size, horizontal origin position and text fonts. The unit of measurement (GUNIT) is set to PCT which means that all height measurements will (from axis, title, footnote, legend, symbol statements) be a percentage of the graphics area.

The append gsfmode will append one graph after the next to the gsfname reference. The orientation is set to portrait, as well as, the automatic paper feed for a HP laserjet printer.

```
goptions hsize=6.75 in horigin=1.25 in ftext=simplex gunit=pct
gsfmode=append rotate=portrait autofeed colors=(black red
green) device=lj600ps;
```

Title statements are set similar to non-graphics PROCs except the height (h=) and justification (j=) can be assigned. Please notice for a text string the justification which directly precedes will be applied (and the DEFAULT is CENTER).

```
title1;
title2 h=1.5 j=left 'Study Name' j=right '(&sysdate)';
title3 h=1.5 'Mean of Percent Change from Baseline';
```

Now, let us focus on generating a plot of the mean of avar by treatment group for each visit.

Axis statements are used to define where the different variable information will be captured. For our purposes, the axis will be the vertical axis and the visits (or months) will go across the bottom of the page.

```
axis1 c=black value=(h=1.5) minor=none label=(h=1.4 a=90
r=0 "MEAN OF % CHANGE") length=5.8 in;
axis2 c=black minor=none major=(n=4) order=(2,3,4,5,5.5)
value=(h=1.5 '6' '12' '18' '24' 'ENDPOINT 24' label=(h=1.5
"MONTH") length=5.2 in;
```

We have now labeled the axis and defined the length (in inches), color, and value ranges. We still need to define the particular symbols to be used and how the data points will be displayed. This is accomplished by using the symbol statement as follows:

```
symbol1 h=1.5 c=blue i=join value=triangle;
symbol2 h=1.5 c=green i=join value=square;
symbol3 h=1.5 c=red i=join value=dot;
symbol4 h=1.5 c=black i=join value=star;
```

Notice we have set the height, color, interpolation option and the actual symbol value. The interpolation option (i=join) tells SAS to draw a straight line from one data point to the next.

By using a legend statement, we can specify how we want our legend to appear. This statement would look as follows:

```
legend1 across=4 position=(top outside center) frame
mode=reserve label=non shape=symbol(5,2) value=(h=1.5
j=1) offset=(2,0) cborder=black;
```

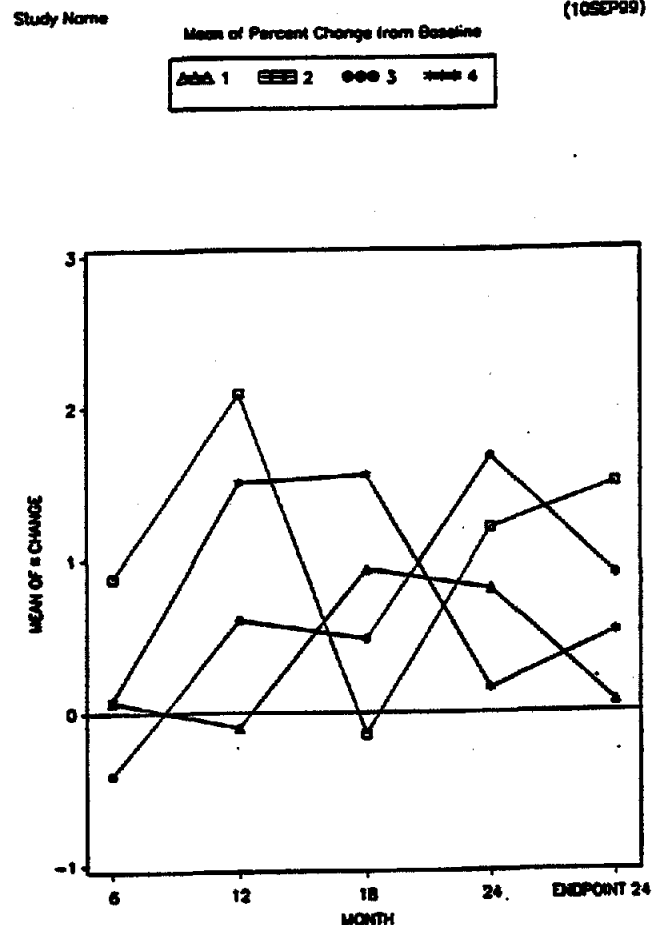
This is specifying to put the legend outside the plot area, up top in the center. The four symbols will go across, the legend text will be surrounded by a frame, without a label. The SHAPE=(width,height) will determine the characteristics of particular symbol segments which appear in the legend. Similarly, the VALUE=(H=, J=) defines the appearance of the text of the legend entry. The MODE= RESERVE tells SAS (in conjunction with the position) to reserve a certain procedure output area specifically for the legend only. In this case, this reserved area will be above the plotting area.

Here is the statement which generates the plot:

```
proc gplot;
plot n*visit=trt / frame vaxis=axis1 haxis=axis2
legend=legend1;
label visit='MONTH' trt='TREATMENT GROUP';
```

A frame has been requested. And a legend, vertical and horizontal axis have been assigned. These statements yield

Plot #1



Program II

The second program example is additional lines of code intended to be added to the Program I example (GPLOT, AXIS statements are being replaced).

We would now like to add the additional statistics SEM and N. These statistics can be derived from the GLM procedure. Proc glm will be run using the OUTSTAT= to capture the SS (Type III Sum of Squares) and DF (Degrees of Freedom) into GLMOUT. Note: the SS and DF statistics will change if the GLM model statement is changed.

```
proc glm data=effdat outstat=glmout order=data noprint;
by visit;
class trt center;
model uvar=trt center trt*center / ss3;
```

```
proc print data=glimout;
  title 'raw data from glm';
```

Print #2

raw data from glm

OBS	VISIT	_NAME_	_SOURCE_	_TYPE_	DF	SS	F	PROB
1	2.0	PCTCHNG	ERROR	ERROR	213	3403.99		
2	2.0	PCTCHNG	TRT	SS3	3	7.09	0.14794	0.93093
3	2.0	PCTCHNG	CENTER	SS3	15	213.20	0.88937	0.57685
4	2.0	PCTCHNG	TRT*CENTER	SS3	38	558.80	0.92017	0.60717
5	3.0	PCTCHNG	ERROR	ERROR	172	2958.75		
6	3.0	PCTCHNG	TRT	SS3	3	93.50	1.81180	0.14683
7	3.0	PCTCHNG	CENTER	SS3	14	380.16	1.57857	0.08940
8	3.0	PCTCHNG	TRT*CENTER	SS3	34	602.42	1.03000	0.43220
9	4.0	PCTCHNG	ERROR	ERROR	117	2541.71		
10	4.0	PCTCHNG	TRT	SS3	3	40.06	0.61465	0.60684
11	4.0	PCTCHNG	CENTER	SS3	13	401.30	1.42098	0.15981
12	4.0	PCTCHNG	TRT*CENTER	SS3	30	399.54	0.61305	0.93877
13	5.0	PCTCHNG	ERROR	ERROR	50	884.59		
14	5.0	PCTCHNG	TRT	SS3	3	12.03	0.22662	0.87744
15	5.0	PCTCHNG	CENTER	SS3	9	219.82	1.38052	0.22213
16	5.0	PCTCHNG	TRT*CENTER	SS3	13	278.04	1.20890	0.30110
17	5.5	PCTCHNG	ERROR	ERROR	239	5856.41		
18	5.5	PCTCHNG	TRT	SS3	3	14.70	0.19995	0.89635
19	5.5	PCTCHNG	CENTER	SS3	15	337.06	0.91704	0.54578
20	5.5	PCTCHNG	TRT*CENTER	SS3	40	1067.08	1.08869	0.34020

The error observations will be selected out and merged with the univariate statistics. The SEM is calculated, as well as, a shift being added to the visit variable thus creating an offset on the graph.

```
data glimout;
  set glimout;
  by visit;
  keep visit mse ss df;
  if _source_='ERROR';
  mse = ss/df;
```

```
data plotdat;
  merge uniout glimout;
  by visit;
```

```
sem = sqrt(mse/n);
mnupper = mn + sem;
mnlower = mn - sem;
```

```
if visit ne 5.5 then do;
  if trt=2 then visit=visit+.2;
  if trt=3 then visit=visit+.4;
  if trt=4 then visit=visit+.6;
end;
else if visit ne 5.5 then do;
  visit = 6;
  if trt=2 then visit=visit-.6;
  if trt=3 then visit=visit-.4;
  if trt=4 then visit=visit-.2;
end;
format trt trtment.;
```

The plotdat data set can now be used to generate an annotate data set which will contain the SEM lines and the corresponding N text values. First, we will create a SEMLINES data

set which contain the commands to annotate the Standard Error of the Mean lines.

```
proc sort data=plotdat;by trt visit;
```

```
data semlines;
  set plotdat;
  length function color $ 8;
  retain color 'BLACK' when 'A' xsys ysys '2';
```

```
if df=0 then do;
  y = mn;
  mnupper = mn;
  mnlower = mn;
end;
```

```
function = 'MOVE'; x = visit-.10; y = mnupper; output;
function = 'DRAW'; x = visit+.10; y = mnupper; line = 1;
output;
function = 'MOVE'; x = visit; y = mnupper; output;
function = 'DRAW'; x = visit; y = mnlower; line = 1; output;
function = 'MOVE'; x = visit-.10; y = mnlower; output;
function = 'DRAW'; x = visit+.10; y = mnlower; line = 1; output;
```

Notice that each observation in the PLOTDAT data set will generate six observations in the SEMLINES annotate data set (see Print #3). Additionally, the graphics "pointer" is first moved to a defined location where the line will start from. Subsequently, the next "DRAW" command (function='DRAW') specifies the type of line to draw (line=1), and to what location the line will be drawn to.

Next, the NTEXT dsu will be created to display

on the graph the number of observations which contributed to the SEM calculation.

```
data ntext;
  set plotdat;
  length function color $ 8;

  if df=0 then do;
    y      = mn;
    mnupper = mn;
    mnlower = mn;
  end;

  when      = 'A';
  xsys      = '2';
  ysys      = '2';
  size      = .6;
  position  = '5';
  color     = 'BLACK';
  y         = mnlower - .10;
  function  = 'LABEL';
  x         = visit;
  text      = put(n,2.);
  output;

  * CONCATENATE TEXT AND LINES DSN'S *;
data annotate;
  set ntext semlines;
  if n gt 0;

proc print data=annotate;
  var visit trt mnupper mnlower x y function color text;
  where trt in(2);
  title'annotate';
```

We are changing our axis statements by adding the order specification to our vertical axis. This is needed to allow the SEM lines to appropriately fit on the graph. SAS will scale for the data from PLOTDAT but cut off any data points which are outside the relative plotting area.

```
axis1 c=black order=(-1.5 to 3.0 by .5) value=(h=1.5) minor=none label=(h=1.4 a=90 r=0 "MEAN OF % CHANGE") length=5.8 in;
axis2 c=black minor=none major=(n=4) order=(2,3,4,5,6) value=(h=1.5 '6' '12' '18' '24' 'ENDPOINT 24' label=(h=1.5 "MONTH")
length=5.2 in;
```

If you compare the AXIS2 statements the ORDER= portion have a fifth value of 5.5 and 6. After reviewing Plots #1 and #2, the hash marks on the horizontal axis are spaced equally. The spacing does not depend on the values, when the above notation is used.

Here is the corresponding titles and GPLOT statements:

```
title1;
title2 h=1.5 j=left 'Study Name' j=right "(&sysdate)";
title3 h=1.5 "N and Mean (" h=1.5 font=math 'G' h=1.5 font=simplex 'SEM) of Percent Change from Baseline';

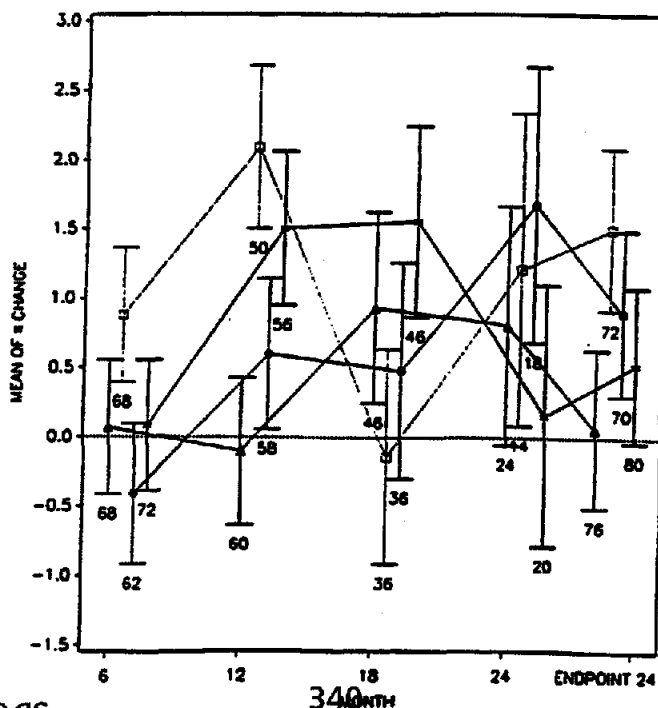
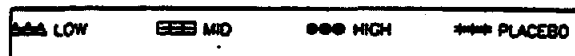
proc gplot data=plotdat;
  plot mn*visit=trt / lvref=2 vref=0 annotate=annotate frame vaxis=axis1
        haxis=axis2 legend=legend1;
  label
    visit = 'MONTH'
    trt   = 'TREATMENT GROUP'
  ;
run;
```

Print #3

annotate									
OBS	VISIT	TRT	MNUPPER	MNLOWER	X	Y	FUNCTION	COLOR	TEXT
6	2.1	MID	1.36653	0.39696	2.1	0.27696	LABEL	BLACK	68
7	3.1	MID	2.67844	1.50534	3.1	1.38534	LABEL	BLACK	50
8	4.1	MID	0.63418	-0.91945	4.1	-1.03945	LABEL	BLACK	36
9	5.1	MID	2.33438	0.08609	5.1	-0.03391	LABEL	BLACK	14
10	5.8	MID	2.08306	0.91630	5.8	0.79630	LABEL	BLACK	72
51	2.1	MID	1.36653	0.39696	2.0	1.36653	MOVE	green	
52	2.1	MID	1.36653	0.39696	2.2	1.36653	DRAW	green	
53	2.1	MID	1.36653	0.39696	2.1	1.36653	MOVE	green	
54	2.1	MID	1.36653	0.39696	2.1	0.39696	DRAW	green	
55	2.1	MID	1.36653	0.39696	2.0	0.39696	MOVE	green	
56	2.1	MID	1.36653	0.39696	2.2	0.39696	DRAW	green	
57	3.1	MID	2.67844	1.50534	3.0	2.67844	MOVE	green	
58	3.1	MID	2.67844	1.50534	3.2	2.67844	DRAW	green	
59	3.1	MID	2.67844	1.50534	3.1	2.67844	MOVE	green	
60	3.1	MID	2.67844	1.50534	3.1	1.50534	DRAW	green	
61	3.1	MID	2.67844	1.50534	3.0	1.50534	MOVE	green	
62	3.1	MID	2.67844	1.50534	3.2	1.50534	DRAW	green	
63	4.1	MID	0.63418	-0.91945	4.0	0.63418	MOVE	green	
64	4.1	MID	0.63418	-0.91945	4.2	0.63418	DRAW	green	
65	4.1	MID	0.63418	-0.91945	4.1	0.63418	MOVE	green	
66	4.1	MID	0.63418	-0.91945	4.1	-0.91945	DRAW	green	
67	4.1	MID	0.63418	-0.91945	4.0	-0.91945	MOVE	green	
68	4.1	MID	0.63418	-0.91945	4.2	-0.91945	DRAW	green	
69	5.1	MID	2.33438	0.08609	5.0	2.33438	MOVE	green	
70	5.1	MID	2.33438	0.08609	5.2	2.33438	DRAW	green	
71	5.1	MID	2.33438	0.08609	5.1	2.33438	MOVE	green	
72	5.1	MID	2.33438	0.08609	5.1	0.08609	DRAW	green	
73	5.1	MID	2.33438	0.08609	5.0	0.08609	MOVE	green	
74	5.1	MID	2.33438	0.08609	5.2	0.08609	DRAW	green	
75	5.8	MID	2.08306	0.91630	5.7	2.08306	MOVE	green	
76	5.8	MID	2.08306	0.91630	5.9	2.08306	DRAW	green	
77	5.8	MID	2.08306	0.91630	5.8	2.08306	MOVE	green	
78	5.8	MID	2.08306	0.91630	5.8	0.91630	DRAW	green	
79	5.8	MID	2.08306	0.91630	5.7	0.91630	MOVE	green	
80	5.8	MID	2.08306	0.91630	5.9	0.91630	DRAW	green	

Study Name (10SEP99)
N and Mean ($\pm$ SEM) of Percent Change from Baseline

Plot #2



Another difference to observe between Plot 1 and Plot 2 are how the formatted values appear in the second legend. This is because the SAS Graph (like most SAS output procedures), will apply formats to its output. If you review Program II you will find a format trtment attached to the TRT variable. This format looks like:

```
proc format;
  value trtment
    1 = 'LOW'
    2 = 'MID'
    3 = 'HIGH'
    4 = 'PLACEBO';
```

Macro Design

The SAS MACRO facility enables several lines of code to be surrounded by macro and mendl statements (%macro example, %mend example) and subsequently be executed by just referencing the macro by name (%example). Additionally, certain predefined parameters can be varied at invocation thus eliminating the need to repeat the same lines of code over and over. When designing macros, a programmer must decide what might possibly change from one execution to the next, and then define these changes as parameters (or macro variables). These types of parameters can be setup as macro variables within the macro (either as a local or global definition) or outside the macro definition (with a %let or call symput). A local macro variable can only be referenced at or below the "level" (or macro) it is defined in.

Program III

This program is similar to program II except it will be designed as a macro (SEMPLOT) to allow the code to be re-executed for different analysis variables.

First, the program name, printer device and number of treatments are defined using the %let approach.

```
%let pgm=ezplot;
%let pdev=lj600ps;    * Laser Jet printer *;
%let ntrt=4;
```

Next, we define the standard options and graphics options:

```
filename gsafile "&pgm &pdev A";
```

* GRAPHICS OPTIONS;

```
goptions hsize=6.75 in horigin=1.25 ftext=simplex
gunit=pct gsfmode=append rotate=portrait
autofeed colors=(black red green blue);
```

* SYSTEM OPTIONS;

```
options nofmterr nodate number ls=132 ps=60
mprint ffmtsearch=(stored.fmts library);
```

The input data set (datin), analysis variable (avar), title (title), plot number (num) and visit range (visrange) are defined as keyword parameters to the SEMPLOT macro.

Clinical trials data can be collected at various visits and it is sometime necessary to run only specific visits. Therefore, the visit range selection is implemented. I have defined the default to include all visits by setting VISRANGE=%str(2,3,4,5,6) in the macro definition.

If you review the program you will notice the macro variables references are preceded with an ampersand (&). This will symbolically substitute the values for these variable which are assigned when the macro is called. The keyword parameter will default the values assigned in the macro statement, if they are not referenced in the macro call. Furthermore, the order the keyword parameters appear (in the macro invocation) does not matter.

```
%macro semplot(datin=,avar=,title=,
visrange=%str(2,3,4,5,6),num=);
```

* CALCULATE N AND MEAN;

```
proc univariate data=&datin noprint;
  where visit in(&visrange);
  by visit trt;
  var &avar;
  output out=uniout n=n mean=mn;
```

* CALCULATE MSE BY GLM;

```
proc glm data=&datin outstat=glmout order=data noprint;
  where visit in(&visrange);
  by visit;
  class trt center;
  model &avar = trt center trt*center / ss3;
```

```
data glmout;
  set glmout;
  by visit;
  keep visit mse ss df;
```

```

if _source_ = 'ERROR';
mse = ss/df;

* CREATE DSN TO PLOT;
data plotdat;
merge unout glmout;
by visit;

* CALCULATE SEM AND UPPER AND LOWER ERROR
VALUES;
sem = sqrt(mse/n);
mnupper = mn+sem;
mnlower = mn-sem;

* CREATE OFFSET BY VISIT FOR TREATMENT GROUPS;
if visit ne 5.5 then do;
if trt = 2 then visit = visit + .1;
if trt = 3 then visit = visit + .2;
if trt = 4 then visit = visit + .3;
end;
else if visit = 5.5 then do;
visit=6;
if trt = 1 then visit = visit - .3;
if trt = 2 then visit = visit - .2;
if trt = 3 then visit = visit - .1;
end;
format trt trtment.;

proc sort data=plotdat;by trt visit;

* CREATE THE SEM LINES;
data semlines;
set plotdat;
length function color $ 8;
retain color 'BLACK' when 'A' xsys ysys '2';

function = 'MOVE'; x = visit-.10; y = mnupper; output;
function = 'DRAW'; x = visit+.10; y = mnupper; line = 1;
output;
function = 'MOVE'; x = visit; y = mnupper; output;
function = 'DRAW'; x = visit; y = mnlower; line = 1; output;
function = 'MOVE'; x = visit-.10; y = mnlower; line = 1;
output;
function = 'DRAW'; x = visit+.10; y = mnlower; line = 1;
output;

data semlines;
set semlines end=eof;
retain min 9999 max -9999;
if trt=1 then color='blue';
if trt=2 then color='green';
if trt=3 then color='red';
if trt=4 then color='black';

* IF TREATMENT GROUP HAS ONLY ONE OBS *;
if df=0 then do;
y = mn;
mnupper = mn;
mnlower = mn;
end;

* SEARCH FOR MIN AND MAX VALUES FOR SCALING *;
max = max(max,mnupper);
min = min(min,mnlower);

* CALCULATE RANGE / INCREMENT;
if eof then do;
incr=round((max-min)/10,.01);
if 0 le incr le .25 then incm=.25;else
if .25 lt incr lt .50 then incm=.5;else
if .50 lt incr lt 10 then incm=round(incr,1.);else
if 10 lt incr lt 100 then incm=round(incr,10.);else
if 100 lt incr lt 1000 then incm=round(incr,100.);

call symput('incr',compress(incr));
call symput('textdif',round((max-min)/30,.01));
call symput('vmax',round(incr+max,incr));
call symput('vmin',round(-incr+min,incr));
end;
run;

* CREATE TEXT FOR N'S;
data ntext;
set plotdat;
length function color $ 8;

* IF TREATMENT GROUP HAS ONLY ONE OBS *;
if df=0 then do;
y = mn;
mnupper = mn;
mnlower = mn;
end;

when = 'A';
xsys = '2';
ysys = '2';
size = .6;
position = '5';
color = 'BLACK';
y = mnlower - &textdif;
function = 'LABEL'; x = visit; text = put(n,2.);
output;

* CONCATENATE TEXT AND LINES DSN'S;
data annotate;
set ntext semlines;
if n gt 0;

title1;
title2 h=1.5 j=left 'Study Name'
justify = right '(&sysdate)';
title3 h=1.5 j=left 'Study Protocol #';
title4 h=1.5 'Figure &num (&pgm)';
title5 h=1.5 '&title';
title6 h=1.5 'N and Mean (' h=1.5 font=math 'G'
h=1.5 font=simplex ' SEM) of &avar';

axis1 c=black order=(&vmin to &vmax by &incr) value=(h=1.5)
minor=none
label=(h=1.4 a=90 r=0 'MEAN OF %upcase(&avar)')
length=5.6 in;
axis2 c=black minor=none major=(n=4) offset=(4)
order=(2,3,4,5,6) value=(h=1.5 '6' '12' '18' '24' 'ENDPOINT
24')
label=(h=1.5 j=center 'MONTH') length=5.2 in;

symbol1 h=1.5 c=blue i=join value=triangle f=;
symbol%eval(&ntrt-2) h=1.5 c=green i=join value=square f=;
symbol%eval(&ntrt-1) h=1.5 c=red i=join value=dot f=;
symbol&ntrt h=1.5 c=black i=join value=star f=;

```

```

legend1 across=&ntrt position=(top outside center) frame mode=reserve
      label=none shape=symbol(5,2) value=(h=1.5 j=1) offset=(2,0)
      cborder=black;

proc gplot data=plotdat;
  plot mn*visit=trt / lvref=2 vref=0 annotate=annotate frame
      vaxis=axis1 haxis=axis2 legend=legend1;
  label
      visit = 'MONTH'
      trt   = 'TREATMENT GROUP';
run;
title;
%mend semplot;

%semplot(datin=datin,avar=avar,num=1, title=Macro Example of SEM lines);

```

Program III is identical to program II (except for the macro variable references) until the second SEMLINES data step. In this step, there are "data-driven" parameters which are calculated. These parameters (incr, textdif, vmax, vmin) assign the appropriate y axis scaling (maximum and minimum values) and the corresponding increment values. These are needed to fully automate the creation of the Mean +/-SEM plot. This algorithm seems to work for most cases.

Conclusion

As we have seen, passing statistics to SAS GRAPH is straight forward. The annotate facility can be utilized to make statistics more meaningful. Finally, SAS MACROS can reduce programming time substantially, if they are designed properly.

Acknowledgments

Dr. Kim Andriano contributed significantly to the statistical methodologies utilized in this paper.

For Further Information

The author can be reached by mail at

Michael Keohan
 E-Z Solutions, Inc.
 Information Systems Consulting
 1770 Massachusetts Avenue
 Suite 286
 Cambridge, MA 02140

or by phone at

(201) 503-8102

or through the internet at:

keohanm@delphi.com or Keohan.Michael@EHCCGATE.SANDOZ.COM