VISUALIZATION OF SCIENTIFIC DATA:

SPECIALTY GRAPHS WITHOUT USING ANNOTATE

Arthur L. Carpenter

ABSTRACT

SAS/GRAPH™ is flexible enough to create many graphs and charts that would not normally be considered standard SAS/GRAPH output. Although the ANNOTATE Facility is often used to "dress up" a graph, there are a number of non-ANNOTATE techniques which can also be used to create non-standard graphics.

The examples illustrated in this presentation use the technique of data manipulation to "trick" SAS/GRAPH into producing graphs and charts that at first glance appear to be outside of its capabilities. This technique uses the incoming data to create additional data points. These data points are then manipulated and used in conjunction with the SYMBOL and PATTERN statement.

When created in the proper sequence these additional data points can create error bars on scatter plots, histograms, butterfly plots, box-whisker plots, histograms with broken scales, multiple axes, and special shapes and symbols.

Data manipulation is a fairly easy way to create specialty plots without resorting to the ANNOTATE facility. This paper is adapted from a section in the SAS/GRAPH book *Visualization of Scientific Data: Using SAS/GRAPH Without Annotate* by Art Carpenter and Charlie Shipp. Additional examples can be found in Carpenter, 1989.

Error bars

Error bars can be generated directly by using the interpolate option in the SYMBOL statement. There are some disadvantages to this approach, however. The statement `i=std2mjd` requests that the scatter plot be placed such that for each value of the horizontal variable a vertical line should be drawn two standard errors above and below the mean and that the means should be joined. Although the data will not be plotted it is used to determine the vertical axis. Figure 4.6.1a shows a plot generated using the INTERPOL option. The upper vertical axis is much too high (to accommodate a single large unplotted point), and the lower axis values do not allow the printing of all of the error bars (since there were no biomass values less than zero). Figure 4.6.1b corrects the scaling problems and allows symbols to be added to the plot at the mean and at the ends of the error bars.

```
proc gplot data=vol1.biomass;
plot bmtotl* date;
symbol1 v=none i=std2mj  c=black l=1;
title1 'Total Biomass';
title2 f=simplex '(using i=std2mj)';
footnote1 j=l h=1 'Example 4.6.1a';
run;
```

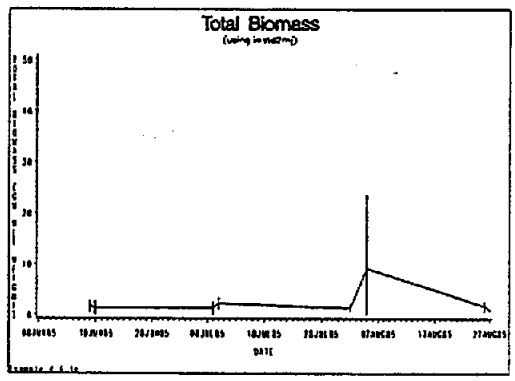

Figure 4.6.1a

We approach the task of creating Figure 4.6.1b from the standpoint of the result. The figure shows means and standard errors, and these can be easily calculated by using PROC SUMMARY (or any of several other univariate summary PROCedures). If the results are saved in a data set, there will be one observation containing the mean and standard error for each horizontal value (DATE). We will manipulate each of these observations to construct four observations which we will then plot. For each date, the code used in Example 4.6.1b outputs an observation for the vertical variable (mean biomass) then adds two (2) standard errors and outputs the second observation. The second and third (mean minus 2 standard errors) become the end points of the error bars. The fourth observation is a repeat of the first and returns the line to the mean so that successive means can be joined. A symbol statement is used to define the line and plot symbols.

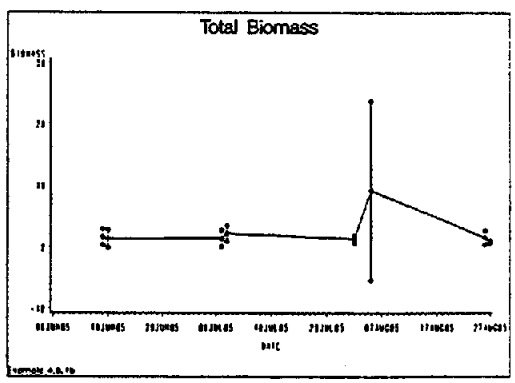

Figure 4.6.1b

```
* Display error bars using data manipulation;
proc sort data=vol1.biomass
          out=biosort;
by date;
run;

* determine the summary statistics;
proc summary data=biosort;
by date;
var bmtotl;
output out=stats mean=mean stderr=se;
run;

* merge the statistics with the primary data set;
data all(keep=date biomass);
merge biosort stats;
by date;
* for each observation create four;
biomass=mean;        output;
biomass=mean+ 2*se; output;
biomass=mean- 2*se; output;
biomass=mean;        output;
run;

proc gplot data=all;
plot biomass * date;
symbol1 v=dot i=join  c=black l=1;
title1 'Total Biomass';
footnote1 j=l h=1 f=simplex 'Example 4.6.1b';
run;
```

Histograms using GPLOT

PROC GCHART does not always provide the flexibility required when generating histograms. GPLOT, on the other hand, has a great deal of flexibility, but is not effective in the design of histograms. The INTERPOL=STEP SYMBOL statement option will create steps, but not a histogram. Through data manipulation GPLOT can be used to create histograms that:

- control the fill pattern according to some criteria
- vary the width of the bars
- break the horizontal axis
- vary the grouping criteria according to a format.
- allow uneven spacing between the bars

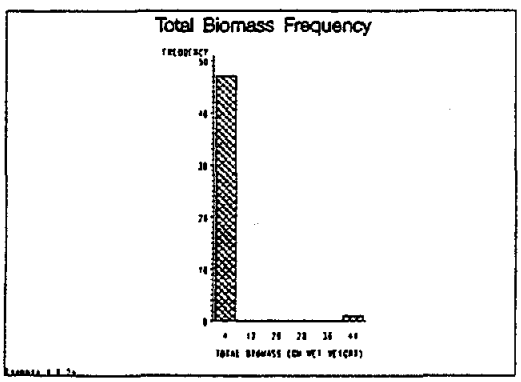

Figure 4.6.2a

Figures 4.6.2a and 4.6.2b were created using GCHART on the BIOMASS data. The grouping variable (BMTOTL) takes on values less than 6 except for one value of 44. When the full range is charted, all the meaningful information is lost (Figure 4.6.2a). The distribution of sizes can be shown, but

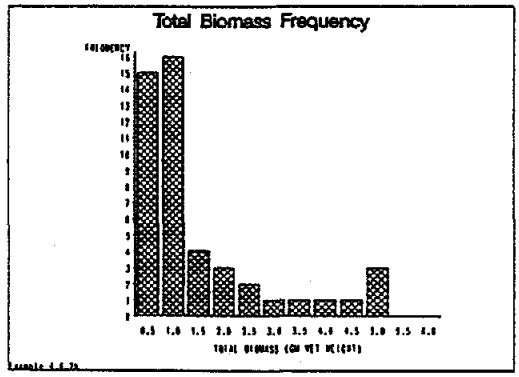

Figure 4.6.2b

only by eliminating the largest value (Figure 4.6.2b). It is possible to produce a meaningful graphic with all the information by using GPLOT and data manipulation (Figure 4.6.2c).

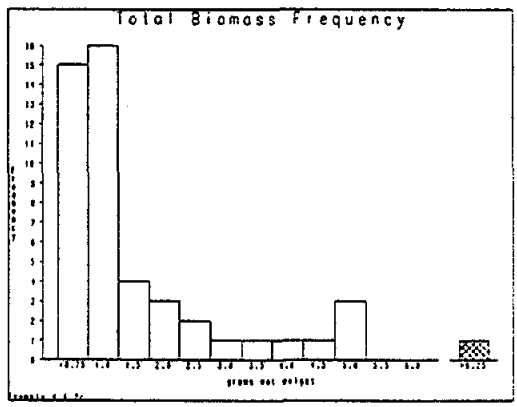

Figure 4.6.2c

The histogram in Figure 4.6.2c was created by PROC GPLOT rather than GCHART. The bars and the split horizontal axis were created using data manipulation techniques and the shading is accomplished with the AREAS= option. The data are grouped into categories and summarized. Then each observation is used to define the four corners of the vertical bar. The split horizontal axis was also constructed by specifying the end points of the two segments. Similar techniques can be used to create bars with variable spacing.

For continuous variables the first step is to define a midpoint variable. The ROUND function is used in this example to create groups spaced at .5 units. Logic is then added to combine any observations that fall on the extremes creating two groups with unequal width.

```
* Group the observations into buckets;
data group (keep=group bmtotl);
set vol1.biomass (keep=bmtotl);
group = round(bmtotl,.5);
* Combine 0-.25 with .25-.75;
if group=0 then group=.5;
* combine all groups > than 6.25 into one bucket;
group = min(7, group);
run;
```

PROC SUMMARY or some other summarization technique is used to summarize the data. This step and the previous step would have been done automatically by GCHART. GROUP will be used as the horizontal plot variable and FREQ as the vertical plot variable.

```
* determine the observation count for each bucket;
proc summary data=group;
by group;
var group;
output out=count n=freq;
run;
```

The default axis must be suppressed and an artificial one built, if a split horizontal axis is desired. The new axis is simply created by drawing two horizontal line segments with appropriately selected endpoints. Temporary variables are used to hold the original values of GROUP and FREQ. When AREAS=n is used in the PLOT statement, this line cannot take on the minimum value and still show so it has a small positive offset.

```
tgroup = group;
tfreq = freq;

*Create a broken horizontal axis;
if _n_=1 then do;
   pltvar=1;
   freq=0.01;
          group=0;   output;
          group=6.4; output;
   pltvar=2;
          group=6.6; output;
          group=7.4; output;
end;
```

The variable PLTVAR is used to distinguish the SYMBOL and PATTERN selections. In this example the first two values are assigned to the horizontal axis. Since the AREAS= option is used the lowest lines are plotted first and should have the lowest values of PLTVAR. The other two values of PLTVAR are used to distinguish the two types of histogram bars.

```
* separate the count in the largest group;
if tgroup le 6 then pltvar=3; else pltvar=4;
```

The summarized data usually has one observation for each bar on the histogram. This observation is then used to define each of the four corners of that bar. This example takes the midpoint (TGROUP) of each bar and establishes the corners at +/- .25 units. Since the width of each bar is now the same as the width of the group the bars will be next to each other. Bars with space between them can be created simply by adjusting the amount added and subtracted from TGROUP.

```
*lower left corner;
group = tgroup - .25; freq =     0; output;
*upper left corner;
                      freq = tfreq; output;
*upper right corner;
group = tgroup + .25;               output;
*lower right corner;
                      freq =     0; output;
```

The remainder of the job is a straight forward GPLOT. The AXIS statement is used to suppress the default horizontal axis and its tick marks, and a format created through PROC FORMAT is used to provide the horizontal axis labels.

```
proc gplot data=plot;
plot freq*group=pltvar / areas=4
                         haxis=axis1
                         vaxis=axis2
                         nolegend;
format group grp5.;
```

Multiple axes

Generally, a user places more than one plot on a page by using TEMPLATES. There are times, however, when due to alignment or other graphical considerations, multiple plots need to be generated without using TEMPLATEs. Example 4.6.2 created a horizontal axis by defining the endpoints of horizontal line segments. Line segments can also be used to create a series of complete axes.

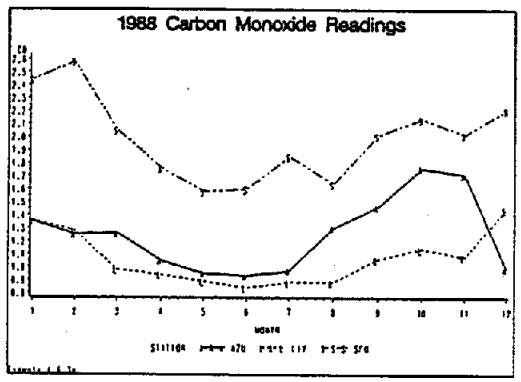

Figure 4.6.3a

The graphic shown in Figure 4.6.3a contains a single plot of three stations. Although not really necessary for this graphic (and perhaps not desirable), the plots of these stations can be separated as shown in Figure 4.6.3b.

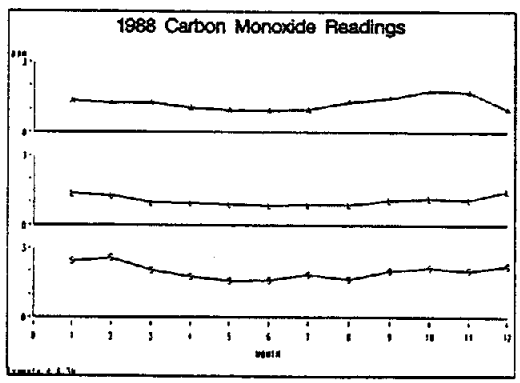

Figure 4.6.3b

The vertical and horizontal axes in Figure 4.6.3b have been generated through the use of line segments and the data have been separated by using a vertical offset. Taken together the three plots appear to have been separated.

The dummy axes are created first, and as was done in Example 4.6.2, the variable PLTVAR is used to identify segments and SYMBOL statements. In this example the

offset is known from Figure 4.6.3a, in actual practice the positioning of the axes will usually be dynamic. For each axis four points are defined. The first three define the endpoints and the origin, and the fourth has a missing value, which when coupled with the SKIPMISS option, separates the axes into distinct segments.

```
* Create dummy axes for each station;
* Allow three vertical units for each axis with one unit
between
* each plot.;
if _n_=1 then do;
   pltvar=1;
   * axis for SFO;
   yvar=0; month=12; output;
           month=0;  output;
   yvar=3;           output;
   yvar=.;           output;
   * axis for LIV;
   yvar=4; month=12; output;
           month=0;  output;
   yvar=7;           output;
   yvar=.;           output;
   * axis for AZU;
   yvar=8; month=12; output;
           month=0;  output;
   yvar=11;          output;
   yvar=.;           output;
   month=tmon;
end;
```

The data values for two of the stations (AZU and LIV) are offset from the true origin by some fixed amount. This amount, which is specified in the OFFSET variable, causes the data for these two stations to be higher on the vertical scale. A distinct PLTVAR value is also used for each station so that different SYMBOL definitions will be applied.

```
if first.station then do;
   * The data for each station is offset vertically to
fit
   * with the dummy axes.;
   if station = 'AZU' then do;
      offset=8;
      pltvar=2;
   end;
   else if station = 'LIV' then do;
      offset=4;
      pltvar=3;
   end;
   else do;
      offset = 0;
      pltvar=4;
   end;
end;
yvar = co + offset;
output;
```

Primary axis and symbol control is maintained through the use of the AXIS and SYMBOL statements. The line for both axes is turned off (STYLE=0) and the first letter of the station name is designated as the plot symbol.

```
* Control the vertical axis;
axis1 order = 0 to 11 by 1
      label = (h=1.1 'ppm')
      minor=none
      style=0;
* Control the horizontal axis;
axis2 minor=none
      style=0;

* Define the symbols for each subplot;
* SYMBOL1 controls the axes;
symbol1 v=none c=black l=1 i=join;
symbol2 v='A'  c=black l=1 i=join f=simplex;
symbol3 v='L'  c=black l=1 i=join f=simplex;
symbol4 v='S'  c=black l=1 i=join f=simplex;
```

The default tick mark values for the vertical axis are replaced with meaningful numbers is supplied through the use of a user generated format (VERT.).

```
* Define a format for the vertical axis;
proc format;
value vert 0,4,8  = '0'
           3,7,11 = '3'
           other  = ' ';
run;
```

The actual PROC GPLOT is fairly straightforward with PLOT statement options to turn off the legend, select the axis statements, and invoke the SKIPMISS option.

```
* Plot the data;
proc gplot data=air1;
plot yvar*month=pltvar / nolegend
                         skipmiss
                         vaxis=axis1
                         haxis=axis2;
format yvar vert.;
title1 '1988 Carbon Monoxide Readings';
footnote1 j=l h=1 'Example 4.6.3b';
run;
```

Figure 4.6.3b has the disadvantage of compressing the vertical axis and thus, in this case, removing much of the 'character' of the graphic. None-the-less this technique can be used to advantage in some situations.

A second vertical axis can also be generated by using the PLOT2 statement. Example 4.6.3c utilizes the PLOT2 statement to generate a second vertical axis and to provide additional control over various aspects of the plot.

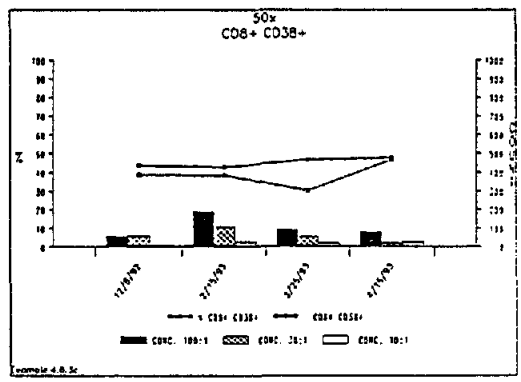

Figure 4.6.3c

Figure 4.6.3c has a number of interesting aspects and displays a lot of information on a single graphic. The data used in this example are from a study on Lymphocytes in the blood of a patient (50x) with a compromised immune system. Both the absolute counts and relative percentages of certain types of immune cells are displayed along with measurements from another assay at three concentrations.

The histogram portion of the graphic has been scaled in a data step to appear below the plots. Since it is the relative height at different concentrations that conveys the important

information, the values of the histograms have been lost. However, for plotting purposes the histogram portion is tied to the left axis. Both axes do represent the values of the plot portion of the graphic, the percent values on the left and the absolute counts on the right. Using two vertical axes (the PLOT2 statement is used for the right axis) allows us to generate and use two different legend statements in the same plot. The histograms were generated by using the methods covered in Example 4.6.2, and they were filled by using the AREAS= option on the PLOT statement.

Unlike the previous example this example program has been generalized to accommodate data sets with different dependent variables and a different number of horizontal data points. This was done by implementing it using a macro and macro calls.

The variable SYMNUM is used to control the SYMBOL statement usage, and in this example one observation is used to generate all three vertical bars for each date. The center of the three bars (HOLDPT) is used as the reference point for the generation of all three, hence the changing displacements. POINT is the horizontal plot variable and CONC is the vertical. The variable PTYPE is used to distinguish between the two types of plots.

```
* create pseudo points for the histograms;
* histograms plotted using PLOT statement;
conc2 = .; symnum2=5; ptype='1';
holdpt = point;
* three vertical bars: conc100, conc30, and conc10;
* create the four corners of each of the three
* bars for each POINT;
* conc100;
symnum = 1;
point=holdpt-.375; conc=0;       ptorder+1; output conc;
                   conc=conc100; ptorder+1; output conc;
point=holdpt-.125;               ptorder+1; output conc;
                   conc=0;       ptorder+1; output conc;
* conc30;
symnum = 2;
point=holdpt-.125; conc=0;       ptorder+1; output conc;
                   conc=conc30;  ptorder+1; output conc;
point=holdpt+.125;               ptorder+1; output conc;
                   conc=0;       ptorder+1; output conc;
* conc10;
symnum = 3;
point=holdpt+.125; conc=0;       ptorder+1; output conc;
                   conc=conc10;  ptorder+1; output conc;
point=holdpt+.375;               ptorder+1; output conc;
                   conc=0;       ptorder+1; output conc;
point=holdpt;
```

The PROC GPLOT call contains both a PLOT and a PLOT2 statement. This is required for a couple of reasons. First there are two different types of legends to display and the LEGEND statement can only do one type of legend at a time. Secondly the AREAS= option used with the PLOT statement complicates the graph. The real trick is coordinating the SYMBOL statement calls and definitions, which is done by using two separate variables (SYMNUM1 and SYMNUM2) to hold the definition request number.

The BY statement is also included, but it is used only to invoke the #BYVAL title option. Obviously the order that the points are placed onto the plot is very important and PTORDER insures that they stay in proper sequence.

```
goption hby=0;
proc gplot data=conc;
by patient;
plot conc * point = symnum / areas=3
                             skipmiss
                             legend=legend1
                             vaxis=axis2
                             haxis=axis1;

plot2 conc2*point = symnum2/ vaxis=axis3
                             legend=legend2
                             skipmiss;

symbol1 v=none c=black i=join l=1;
symbol2 v=none c=black i=join l=1;
symbol3 v=none c=black i=join l=1;
symbol4 v='U'  c=black i=join l=1 f=marker  h=.5;
symbol5 v='U'  c=black i=join l=1 f=markere h=.5;

pattern1 v=solid c=black;
pattern2 v=m2x45 c=black;
pattern3 v=empty c=black;

label conc2 = 'COUNT'
      conc  = '%';
title1  f=duplex h=1.5 '#byval(patient)';
title2  f=simplex h=1.5 "&varstrg";
footnote h=1 j=l f=simplex 'Example 4.6.3c';
run;
```

Drawing shapes (Butterfly Plots)

Butterfly plots can be used as a variation of a scatter plot when there is only one y value for each x value. Figure 4.6.4 contains the same information as was presented in Figure 4.6.3b, however, Figure 4.6.4 is much more expressive than Figure 4.6.3b and retains more of the plot's character. The height of the butterfly figure is the measure of magnitude of the dependent variable.

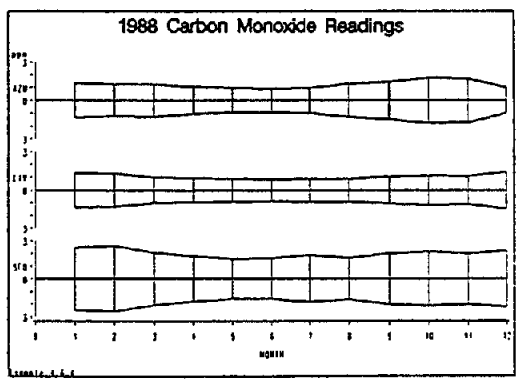

Figure 4.6.4

There are a number of similarities between the code used to generate Figures 4.6.3b and 4.6.4. Both create dummy axes, but the shape generated in a butterfly plot is centered on the horizontal axis, and as a consequence four statements are needed to define each axis.

```
* Create dummy axes for each station;
* Allow six vertical units for each axis with one unit
between
* each plot.;
if _n_=1 then do;
   pltvar=0;
   * axis for SFO;
```

```
   yvar=0; month=0;  output;
   yvar=6;           output;
   yvar=3;           output;
          month=12; output;
   yvar=.;           output;
   * axis for LIV;
   yvar=7; month=0;  output;
   yvar=13;          output;
   yvar=10;          output;
          month=12; output;
   yvar=.;           output;
   * axis for AZU;
   yvar=14;month=0;  output;
   yvar=20;          output;
   yvar=17;          output;
          month=12; output;
   yvar=.;           output;
   month=tmon;
end;
```

The OFFSET for each station is assigned, as is a variable (PLT) which counts the stations. Because PLT is inside the FIRST.STATION DO loop, it will be incremented once for each new station.

```
if first.station then do;
   plt+1;
   * The data for each station is offset vertically to
fit
   * with the dummy axes.;
   if station = 'AZU' then offset=17;
   else if station = 'LIV' then offset=10;
   else offset = 3;
end;
```

The butterfly for each station is made up of three lines; the upper and lower bounds and a vertical line joining the bounds at each horizontal value. Each of these lines has its own unique value of PLTVAR for each station. The current value of PLTVAR is a function of the station number (PLT), the number of stations (3), and the type of line.
Some butterfly plots do not include the vertical line. In this example the vertical lines are connected along the dummy horizontal axis. This in effect abrogates the need for drawing the dummy horizontal axis for each station. The vertical line for each month consists of three segments (four points). These are added starting and ending at the center line so that they can be connected from month to month. The missing value technique could have been used to provide a break between months if desired.

```
* Each butterfly plot is made up of three sets of lines
* with a common SYMBOL statement;
* Upper line;
pltvar = plt*3-2; yvar = offset + co; output;
* Lower line;
pltvar = plt*3-1; yvar = offset - co; output;
* Hilo line;
pltvar = plt*3  ; yvar = offset     ; output;
                  yvar = offset - co; output;
                  yvar = offset + co; output;
                  yvar = offset     ; output;
```

The axis statements and the PROC GPLOT step are very similar to Example 4.6.3b, although the PROC FORMAT has been somewhat altered.

```
* Define a format for the vertical axis;
proc format;
value vert 0,7,14  = '3'
           6,13,20 = '3'
           3,10,17 = '0'
           4       = 'SFO'
           11      = 'LIV'
           18      = 'AZU'
           other   = ' ';
run;

* Plot the data;
proc gplot data=air1;
plot yvar*month=pltvar / nolegend
                         skipmiss
                         vaxis=axis1
                         haxis=axis2;
format yvar vert.;
title1 '1988 Carbon Monoxide Readings';
footnote1 j=l h=1 'Example 4.6.4';
run;
quit;
```

Box-whisker

With release 6.06, the SYMBOL statement option INTERPOL can be set to BOX. This option will generate box-whisker plots and has several variations. When the available options are not sufficient, box plots can be generated using the same techniques as shown in Example 4.6.4. Selected statistics such as the median and various percentiles can be calculated using PROC UNIVARIATE and added onto the graph. Extensions of the box-whisker plots shown here are easily created using these same techniques.

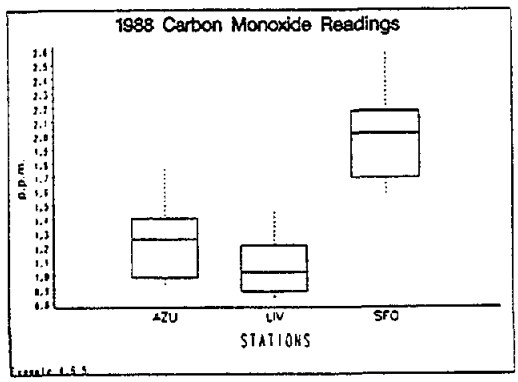

Figure 4.6.5

The box plot shown in Figure 4.6.5 is a fairly simple representation of one of a variety of box plots that can be generated using these methods. The median, quartile range, first quartile, and third quartile are calculated and saved through PROC UNIVARIATE.

```
* determine the median, and quartile
* statistics for each station;
proc univariate data=caair noprint;
by station;
var co;
output out=stats median=median
                q1=q1 q3=q3 qrange=qrange;
run;
```

The length and end points of the whiskers are determined by the quartile range. Some box plots will use a length of 1.5 times the quartile range (QRANGE). In this example the length is extended to the most extreme point that falls within this distance. This requires an additional pass of the data which saves the high and low end points (HIGHPT & LOWPT). These two variables are added to the summary

statistics generated by UNIVARIATE.

```
* Determine the whisker endpoints;
* Whisker endpoints are the most extreme
* data values that are
* within 1.5*qrange of the quartiles.;
data stats2;
merge stats caair;
by station;
retain lowpt highpt;
drop co;
if first.station then do;
  lowpt=.; highpt=.;
end;

* does this point determine the whisker end point?;
if q1-1.5*qrange <= co <= q3+1.5*qrange then do;
   * look for the smallest value that is between
   * q1-1.5*qrange and q1;
   if lowpt=. then lowpt = co;
   * look for the largest value that is between
   * q3+1.5*qrange and q3;
   if highpt=. then highpt = co;
   else highpt=max(highpt,co);
end;

if last.station then output;
run;
```

The data set STATS2 has all the information to build the boxes, however one of the useful advantages of the box plot is to show any outliers. To do this the original data are merged with the statistics in order to determine which if any (there aren't any in this example) observations extend past the ends of the whiskers. Outliers (points between 1.5 and 3.0 times the quartile range from the quartile) are noted by receiving a PLTVAR code of 3. Extreme outliers (greater than 3 times the quartile range from the quartile) receive a different value of PLTVAR and will be plotted with a different symbol.

The box and whiskers are built using the point addition method used in the previous example. The whiskers are assigned a PLTVAR of 1 which will be matched to a dotted line in a SYMBOL statement. The solid lines of the box are assigned through the second SYMBOL statement accessed with PLTVAR=2. Notice the missing values to prevent upper and lower whiskers and adjacent boxes to from being joined. STACNT adjusts the positions of the boxes horizontally by counting the stations, therefore this step will automatically accommodate any number of stations.

```
* combine the stats with the data and retain extreme
points;
data both;
merge stats2 caair;
by station;
if first.station then do;
   * Build the box and whiskers from the summary stats;
   stacnt + 1;

   * Whiskers are dotted lines;
   pltvar=1;
   * start at the top whisker;
   xvar=stacnt   ; yvar=highpt      ; output;
                   yvar=q3          ; output;
   xvar=.        ;                  output;
   xvar=stacnt   ; yvar=lowpt       ; output;
                   yvar=q1          ; output;
   xvar=.        ;                  output;

   * The box is a solid line;
   pltvar=2;
   xvar=stacnt+.3; yvar=q3          ; output;
                   yvar=median      ; output;
   xvar=stacnt-.3;                    output;
   xvar=stacnt+.3;                    output;
                   yvar=q1          ; output;
   xvar=stacnt-.3;                    output;
                   yvar=q3          ; output;
   xvar=stacnt   ;                    output;
   xvar=stacnt+.3;                    output;
   xvar=.        ;                    output;

end;

* plot outliers;
xvar=stacnt;
yvar=co;

* Determine where this point falls;
* Extreme outliers;
if co < q1-3*qrange or co > q3+3*qrange then do;
   pltvar=4;
   output;
end;
else if co < q1-1.5*qrange or co > q3+1.5*qrange then
do;
   pltvar=3;
   output;
end;
run;
```

The GPLOT is once again the straight-forward part of the graphic. Axis statements are used to dress up the plot.

```
* Control the vertical axis;
axis1 label=(h=1.5 f=simplex a=90 'p.p.m.');

* Control the horizontal axis;
axis2 label=(h=1.8 'STATIONS')
      order=(0 to 4 by 1)
      major=none
      minor=none
      value= (h=1.5 f=simplex
              t=1 ' ' t=2 'AZU' t=3 'LIV' t=4 'SFO' t=5
' ');

* Define the symbols;
symbol1 v=none c=black l=2 i=join;
symbol2 v=none c=black l=1 i=join;
symbol3 v=circle c=black;
symbol4 v=diamond c=black;

* Plot the data;
proc gplot data=both;
plot yvar*xvar=pltvar / nolegend
                        skipmiss
                        vaxis=axis1
                        haxis=axis2;
title1 h=2 '1988 Carbon Monoxide Readings';
footnote1 j=l h=1.2 'Example 4.6.5';
run;
```

Variations of the box plot include caps on the whiskers, notches in the boxes, and adjustable box width to indicate the number of observations.

Cluster Scatter Plot

The graphical presentation of cluster analysis results can be extremely important if not critical in determining the cohesion and isolation of clusters. Usually the determination of the number of natural clusters, the variables that define those clusters, and cluster membership criteria is a process that may include several trials. Scatter plots of the proposed

clusters can be used as an aid to decide on the final clustering criteria. Although plots are not generated directly by the cluster PROCedures, FASTCLUS allows the cluster information to be saved for later plotting.

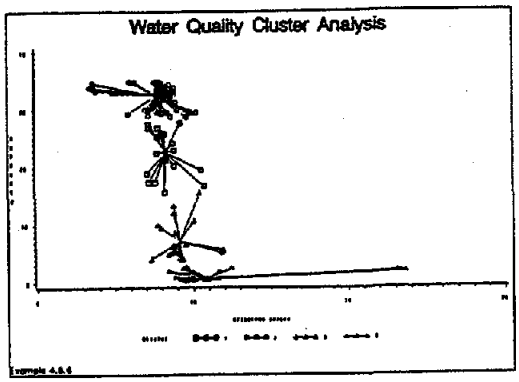

Figure 4.6.6

In Figure 4.6.6 PROC GPLOT is used to plot the water quality data into four possible clusters. The two selected plot variables are SALINITY and DO (dissolved oxygen). Each line segment is drawn from the data point to the center of the cluster by creating a second observation for each data point.

First the cluster analysis is performed using FASTCLUS and the two output data sets are saved. MEANCLUS contains the cluster summary information and OUTCLUS contains each observations cluster membership.

```
proc fastclus data=vol1.h2oqual
     mean=meanclus
     out=outclus
     maxclusters=4
     noprint
     ;
var depth temp ph do cond salinity;
run;

* sort the cluster mean data;
proc sort data=meanclus;
by cluster;
run;

* sort the raw data with the cluster assignments;
proc sort data=outclus;
by cluster;
run;
```

The two data sets are brought together and the cluster summary information is used to define the center of each cluster. This has been done inside a macro which will allow the user to plot any pair of variables easily.

```
%macro pltclus(xvar, yvar);
* create a macro to plot any two data variables;

* Combine the cluster means with the rawdata;
* Select two plotting variables;
data both (keep=cluster &xvar &yvar);
set meanclus (keep= cluster &xvar &yvar)
    outclus  (keep= cluster &xvar &yvar);
by cluster;
retain xmean ymean;
* the first value for each cluster contains the summary
info;
if first.cluster then do;
   xmean= &xvar;
   ymean= &yvar;
   output;
end;
else if &xvar>. and &yvar>. then do;
   * for each point output two observations that form
the line
   * segment from the centroid to the data point;
   output;
   &xvar = xmean;
   &yvar = ymean;
   output;
end;
```

The GPLOT is straight forward and uses the SYMBOL statements to separate the clusters.

```
* graph the clusters using GPLOT;
proc gplot data=both;
plot &yvar * &xvar = cluster;

symbol1 line=1 i=join c=black v=square;
symbol2 line=1 i=join c=black v=circle;
symbol3 line=1 i=join c=black v=triangle;
symbol4 line=1 i=join c=black v=star;
symbol5 line=1 i=join c=black v=diamond;
title1 'Water Quality Cluster Analysis';
footnote h=1 f=simplex j=l 'Example 4.6.6';
%mend pltclus;

%pltclus(do  , salinity)  run;
```

Analysis of this figure indicates that while SALINITY does a good job of separating clusters, dissolved oxygen (DO) by itself does not.

SUMMARY

SAS/GRAPH is highly flexible and can be used to generate many types of graphs that at first seem to be outside of the range of even its capabilities. The points addition method allows us to trick SAS/GRAPH and especially GPLOT into making graphs such as histograms, butterfly charts, and box plots.

TRADEMARK INFORMATION

SAS/GRAPH is a registered trademark of SAS Institute, Inc. in the USA and other countries. ™ indicates USA registration.

REFERENCES

Carpenter, A. and C. Shipp, in press, *Visualization of Scientific Data: Using SAS/GRAPH Without Annotate*. SAS Institute, Inc. Cary, NC.

Carpenter, A. 1989. "The GPLOT Procedure: Quick Tricks for the New User". *Proceedings of the Fourteenth Annual SUGI Conference*. SAS Institute, Inc. Cary, NC. pp. 591-596.

ABOUT THE AUTHOR

Arthur L. Carpenter has over seventeen years of experience as a statistician and data analyst and has served as a senior consultant with California Occidental Consultants, CALOXY,

since 1983. His publications list includes a number of papers and posters presented at SUGI and he has developed and presented several courses and seminars on statistics and SAS programming. Art has served as a steering committee member and president of the Southern California SAS User's Group, a Section Chair at the Western Users of SAS Software regional conference, WUSS, and in various positions at SUGI. He has developed and presented several courses and seminars on statistics and SAS programming and has taught for Colorado School of Mines, University of Redlands, and University of California at San Diego.

CALOXY offers SAS contract programming and in-house SAS training nationwide, including a three day course on SAS/GRAPH. This presentation is based on a section in his latest book, *Visualization of Scientific Data: Using SAS/GRAPH Without Annotate*, which was co-authored with Charlie Shipp.

AUTHOR

Arthur L. Carpenter
California Occidental Consultants
4239 Serena Avenue
Oceanside, CA 92056-5018

(619) 724-8579