

Lumping and Splitting: Using Graphics Building Blocks with SAS ODS Statistical Graphics

Linda Collins, PharmaStat LLC
PhUSE Single Day Event
Sept 25, 2012

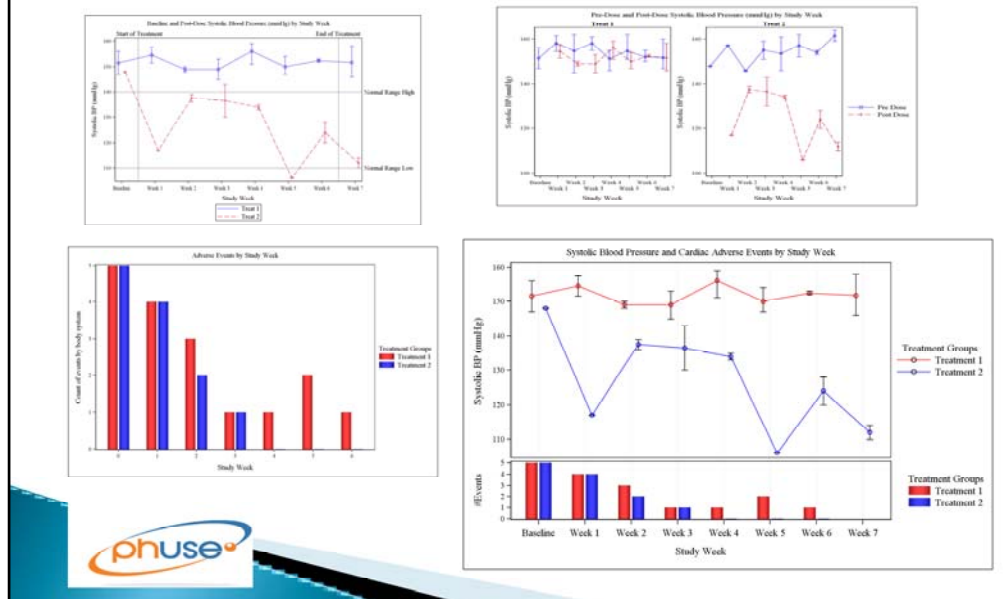




It's been said that there are two kinds of people in the world: those who divide everything into two groups, and those who don't. To taxonomists, these folks are commonly known as 'lumpers' and 'splitters'. Regardless of which type one is, anyone who tries to make sense of data has to figure out how to organize it into coherent patterns. When we design graphical displays, the trick is figuring out meaningful groupings. Is there a way to present one 'big picture', or are the data better understood by arranging smaller summary groups?

In this talk, we will show some ways to use SAS ODS Statistical Graphics to juxtapose or to separate representations of data.

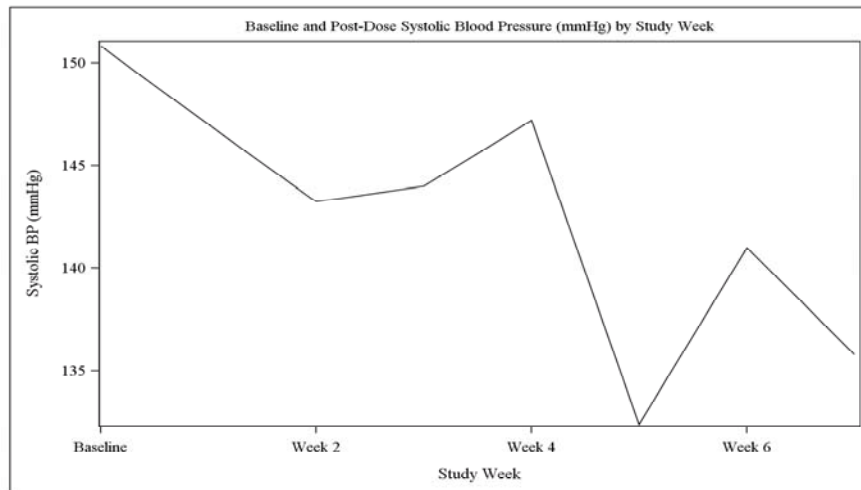
Sample Graphs using ODS SG



In SAS, the ODS statistical graphics software allows users to combine graphical features in a variety of ways. One can produce a wide assortment of graphs including line graphs, bar charts, histograms, box plots, etc. Once the basic elements of a graph are set up, one can:

- Superimpose multiple types of graphical elements on a graph. For instance, scatter plots and line plots can be combined. (upper left graph)
- Use the basic structure of a graph to create a 'lattice' or grid of related graphs (upper right graph)
- Combine multiple graphs into an aligned presentation (lower right graph is a combination of the two on the left)

Simple graph using ODS SG



Let's start with an example of a simple line graph produced with SAS ODS statistical graphics. This is based on vital signs data (systolic blood pressure) over the duration of a short course of study treatment. The code is shown in the next slides.

Processing for simple graph

- ▶ Select the input data: summarize
- ▶ Create a graphics template using ODS Statistical Graphics
- ▶ Redirect the output destination to an RTF file
- ▶ Generate the graph by rendering the summarized statistics through the template using PROC SGRENDER



These are the steps taken in the program that produces the simple graph.

SAS Code: Select and Summarize

```
data advs ;  
  set outdata.advs ;  
  where paramcd = 'SYSBP' and (avisit=0 or atptnum = 2) ;  
run;  
  
proc sort data = advs ;  
  by avisit ;  
run ;  
  
proc univariate data = advs noprint ;  
  by avisit ;  
  var aval ;  
  output out = vsdesc N = N MEAN = MEAN ;  
run ;
```



First, the program accesses the input data and applies any subsetting needed. Then we run a statistical procedure to generate the statistics to be presented.

SAS Code: Create Graphics Template

```
%let title1 = %str(Baseline and Post-Dose Systolic Blood Pressure (mmHg) by Study Week ) ;

proc template;
  define statgraph myplot;
    mvar TITLE1 ;

    begingraph ;
      if (EXISTS(title1))
        entrytitle TITLE1 / textattrs=GRAPHVALUETEXT (size=10pt );
      endif;

      layout overlay / yaxisopts=( type=linear label="Systolic BP (mmHg)" )
        xaxisopts=( type=linear label="Study Week" ) ;

      SeriesPlot X=avisit Y=mean ;

    endlayout;
  endgraph;
end;
run;
```



This section of code sets up a graphics template in the SAS ‘template store’.

Some points to note:

- The DEFINE STATGRAPH starts the graphics template definition. We are naming this template ‘myplot’ so that we can refer to it when we render the graph.

IT is ended with an END statement (This constitutes a “define block”).

- We can refer to macro variables (in this case &title1) by naming them in the MVAR section.

In this case the macro variable is used in an ENTRYTITLE statement to put a title on the graph.

Statements like ENTRYTITLE can be made conditional with IF/ENDIF syntax.

- The BEINGRAPH statement starts the graph block; the overall ‘container’ for the graph.

It is ended with the ENDGRAPH statement (This constitutes a “graph block”).

- The LAYOUT OVERLAY starts the definition of the specific plot layout. This can encompass definitions of axes, as well as one or more plotting statements.

It is ended with the ENDLAYOUT statement (This constitutes a “layout block”).

- The SERIESPLOT statement defines the plot, using variables that will be found in the dataset that is passed to the template.

SAS Code: Create Graphics Output

```
ods rtf file = "..\results\g_sample1_lineplota.rtf"  
    style=styles.rtf ;  
  
ods graphics / height= 5in width=8in ;  
  
proc sgrender data = vsdesc template=myplot ;  
run;  
  
ods rtf close ;
```



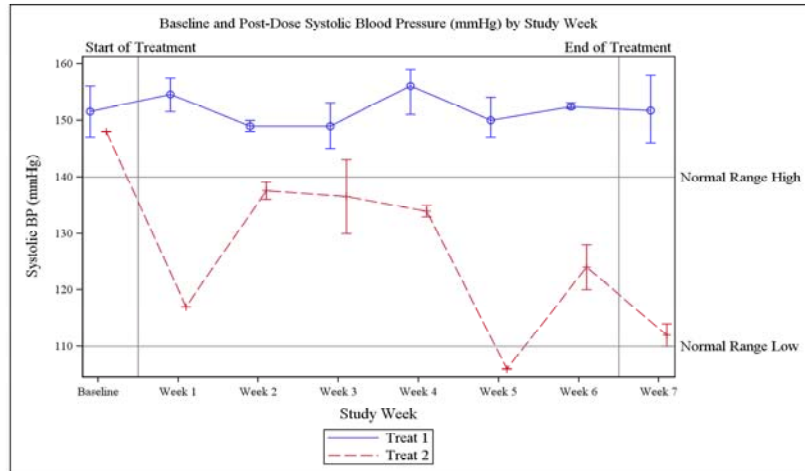
The ODS RTF statement redirects the output to an RTF destination. Optionally, you can designate a predefined SAS style sheet or one set up by the user.

The ODS GRAPHICS statement enables the SG graphics. The options here can be used to define the dimensions of the graph image in the output file.

The PROC SGRENDER procedure links the dataset containing the summarized statistics to the template code. This step actually produces the graphics image.

The ODS RTF CLOSE statement closes the output file.

Splitting and Lumping #1: Adding to the Simple Graph



Now that we have the basics of graph code anatomy, let's start splitting (separating data into meaningful categories) and lumping (adding more useful information).

Let's suppose that our fictitious disease involves abnormally high blood pressure, and we are hoping that the treated subjects are more likely to stabilize into a normal range of blood pressure while on the drug.

For starters, the graph would be much more useful if we separated by treatment groups. The treatment group symbols are associated with a legend.

In addition, we can present more information using overlays. In this example, we have added symbols and error bars to display the quartiles around each mean data point.

We can also provide very useful context by using reference lines to show the extent of the study drug treatment period and the normal range of the measurement.

With these additions, we see that the Treatment 2 subjects do have the average measurements go into the normal range after treatment with study drug, while the placebo group (Treatment 1) remains abnormally high.

Code for Additions to Graph: Data Preparation

```
data advs ;  
  set outdata.advs ;  
  where paramcd = 'SYSBP' and (avisit=0 or atptnum = 2) ;  
  format atrtn trt. avisit avisit. ;  
  if atrtn = 1 then avisit = avisit - 0.1 ;    ** Create an offset in the x-axis value ;  
  if atrtn = 2 then avisit = avisit + 0.1 ;  
run ;  
proc sort data = advs ;  
  by paramcd avisit atrtn ;  
run ;  
proc univariate data = advs noprint ;  
  by paramcd avisit atrtn ;    * break statistics out by treatment ;  
  var aval ;  
  output out = vsdesc N    = N MEAN = MEAN  
                    Q1 = Q1 Q3 = Q3 ;    * generate quartiles ;  
run ;
```



In the modified program, we prepare for the separated lines by generating our statistics separately by treatment group. We make a slight adjustment to the AVISIT variable based on treatment group so that the lines will be slightly offset from each other. We also generate quartile statistics in the dataset that is used for plotting.

Code for Additions to Graph: Template Statements

```
layout overlay / yaxisopts=(type=linear label="Systolic BP (mmHg)" )
                  xaxisopts=(type=linear label="Study Week");

SeriesPlot X=avisit Y=mean / group = atrtn name = "linegrp";
ScatterPlot X=avisit Y=mean / group = atrtn yerrorlower=q1 yerrorupper=q3 ;

DiscreteLegend "linegrp" / location = outside;

referenceline x=0.5 / curvelabel="Start of Treatment"
                  curvelabellocation=outside ;
referenceline x=6.5 / curvelabel="End of Treatment"
                  curvelabellocation=outside ;
referenceline y=110 / curvelabel="Normal Range Low"
                  curvelabellocation=outside ;
referenceline y=140 / curvelabel="Normal Range High"
                  curvelabellocation=outside ;

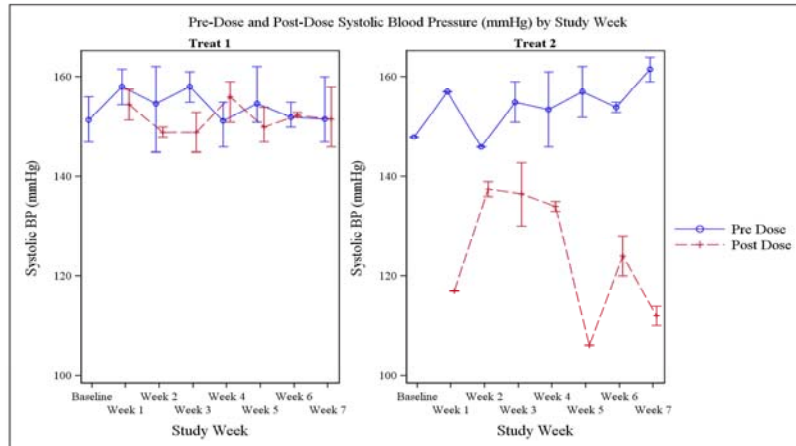
endlayout;
```



We create the additional features by making modifications within the LAYOUT OVERLAY block. Because these statements are all within the same LAYOUT OVERLAY, they are superimposed on a single graph.

- In the SERIESPLOT statement, we add a GROUP= specification that creates separate lines for each unique value of the treatment variable ATRTN.
- The GROUP variable is linked with a LEGEND definition by means of the name we have assigned, "linegrp".
- The DISCRETELEGEND statement allows us to assign characteristics to the legend, such as the location (inside or outside of the graph area). Many other characteristics may be defined.
- The SCATTERPLOT statement will assign symbols and vertical bars representing the interquartile range. It is drawn on top of the SERIESPLOT in this example.
- Each REFERENCELINE statement will draw either a vertical line at the location x= or a horizontal line at the y= location. The reference lines can be labeled with user defined text, either inside or outside the graph location. Note that although these reference lines have hardcoded locations, one could also base a reference line on values supplied in data variables.

Splitting #2: Aligned Graphs in a Grid



Let's suppose we want to dice our data a little bit more finely yet.

Let's say that we suspect that the drug has a short-lived effect that we see soon after the drug administration, and that the treated subjects return to the high values of blood pressure after the drug wears off. We now want to present two different dimensions, a comparison between treatment groups and a comparison between pre-dose and post-dose values. One way to do this is to set up a grid or 'lattice' of graphs, one graph for each treatment group. Within each treatment group, we can use the same technique we saw before to generate separate lines, this time per time point rather than per treatment.

When we do this, we want to create a common legend for the time points. We also want the y-axis to be coordinated so that the two treatment groups are presenting the values on the same scale.

Code for Aligned Graphs: Data Preparation

```
proc univariate data = advs noprint ;  
  by paramcd avisit atptnum atrtn ;  
  var aval ;  
  output out = vsdesc N = N MEAN = MEAN Q1 = Q1 Q3 = Q3 ;  
run ;  
  
data vsdescstran ;  
  set vsdesc ;  
  if atrtn = 1 then do ;  
    N1 = N ; MEAN1 = MEAN ; Q11 = Q1 ; Q31 = Q3 ;  
  end ;  
  if atrtn = 2 then do ;  
    N2 = N ; MEAN2 = MEAN ; Q12 = Q1 ; Q32 = Q3 ;  
  end ;  
  
run;
```



In order to prepare for the aligned graphs, we will include all of the data for all of the timepoints (not shown). We then slice our statistics by timepoint as well as treatment group.

Code for Grid of Graphs: Template Statements

```

layout lattice / columns = 2 rows = 1
                rowDataRange=unionall ;

sidebar / align = right ;
    discretelegend "linegrp" / location = OUTSIDE {other options};
endsidebar ;
%do l = 1 %to 2 ;
    cell ;
        cellheader ; entry "Treat &i" ; endcellheader ;
        layout overlay / {options} ;
            SeriesPlot X=avisit Y=mean&i / group = atptnum
                                name = "linegrp" ;
            ScatterPlot X=avisit Y=mean&i / group = atptnum
                                yerrorlower=q1&i yerrorupper=q3&i ;
        endlayout;
    endcell;
%end ;
endlayout; /* end of lattice */

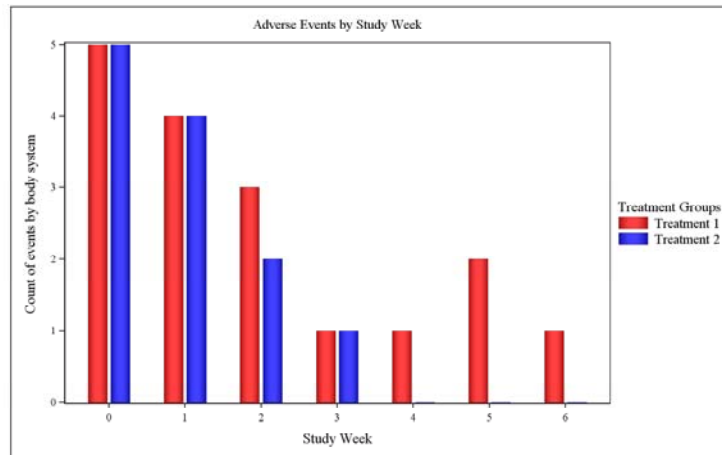
```



To do the 'lattice' or aligned grid of graphs, we do the following:

- Make the major layout block a 'LAYOUT LATTICE'.
- in the layout statement, we define the number of rows and columns of graphs to be generated. In this case we will have two graphs across the page (columns) and one graph per row.
- in the layout statement, we specify rowdataRange=unionall, to say that the graphs within this block should share a common y axis scale.
- Since we want the legend for the lines within this set of graphs to have a common legend, we put the legend statements into a SIDEBAR. This allows us to specify that we want this legend placed in the right margin of the plot area, outside of the individual graphs.
- We then do a macro loop to produce a 'cell' for each graph to be produced. Each cell can have its own header to label the individual graph, in this case with the name of the treatment group.
- Within the cell, we have a LAYOUT OVERLAY as we saw previously. In this case, we are using a separate set of variables for the plotted values for each treatment group.

Lumping #2: Adding a Separate Graph Panel



We've now seen that there is an effect of our drug on blood pressure. Suppose we wanted to investigate whether there is a difference between groups over time in adverse events that could be associated with blood pressure. In this graph, we do a count of the number of reported adverse events that started in each of the study weeks, broken out by treatment group.

First let's look at the bar chart and its code alone. Then we will incorporate it into a combined graph.

(Note: counting events in this way is a fairly crude measure and not likely to be used in an actual analysis. This example is meant to illustrate the graphics code only.)

Code for Bar Chart: Data Preparation

```
data adae ;  
  set outdata.adae ;  
  where aebodsys="Cardiac disorders";  
  if atrtn = 1 then tally1 = 1 ;  
  if atrtn = 2 then tally2 = 1 ;  
run ;  
  
proc sort data = adae ;  
  by avisit aebodsys ;  
run ;  
  
proc univariate data = adae noprint ;  
  by avisit ;  
  var tally1 tally2 ;  
  output out = aecount N = Naes1 Naes2 ;  
run ;
```



(Note: counting events in this way is a fairly crude measure and not likely to be used in an actual analysis. This example is meant to illustrate the graphics code only.)

Here, we select the adverse events of interest and prepare a summary count of events by treatment group by week.

Code for Bar Chart: Template Statements

```
layout overlay / (options)
;
BarChartParm X=avisit Y=Naes1 / name = "stack1" discreteoffset=-0.15
                                legendlabel ="Treatment 1" ;
BarChartParm X=avisit Y=Naes2 / name = "stack2" discreteoffset= 0.15
                                legendlabel ="Treatment 2" ;

DiscreteLegend "stack1" "stack2" /
                                location=OUTSIDE
                                halign=RIGHT
                                border=FALSE
                                title="Treatment Groups"
                                across  = 1;

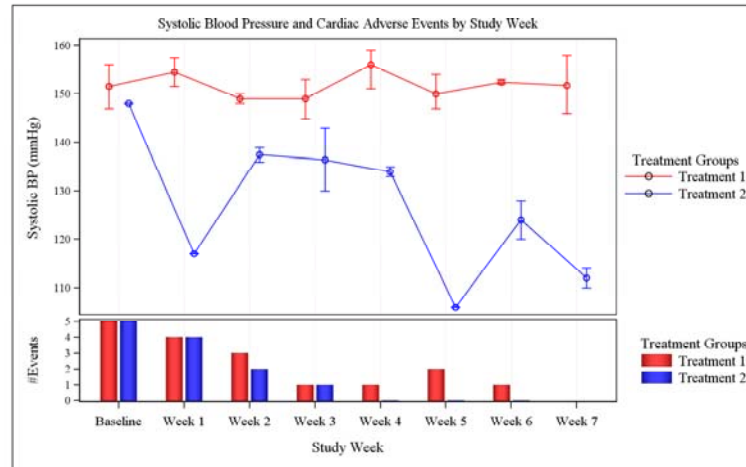
endlayout;
```



Here, we have a LAYOUT OVERLAY similar to the one we saw earlier for the line graph. This layout contains statements for a BARCHARTPARM rather than the SERIESPLOT or SCATTERPLOT.

The DISCRETEOFFSET is an option that is available in SAS 9.3 mod 3 and higher. This is available with DISCRETE axes only and shifts the position of the bars in relation to the axis tick mark, so that the bars are displayed next to each other.

Lumping #2: Fitting Line and Bar Graphs Together



In this graph, we combine the code for the line graph and the bar chart that we saw separately. This allows one to look at the relationship of blood pressure measurements and the number of adverse events together overtime.

Code for 2-Panel Graph: Template Statements

```

layout lattice / columndatarange=unionall rows=2 rowweights=(0.75 0.25);
  columnaxes ;
    columnaxis / griddisplay = on label="Study Week" ;
  endcolumnaxes ;
  layout overlay / xaxisopts=( type=discrete label=" Study Week" (other options) ;
    SeriesPlot X=avisit Y=mean1 / name = "linegrp1" legendlabel ="Treatment 1" ;
    SeriesPlot X=avisit Y=mean2 / name = "linegrp2" legendlabel ="Treatment 2" ;
    ScatterPlot (as shown previously) ;
    DiscreteLegend "linegrp1" "linegrp2" / location=OUTSIDE title="Treatment Groups"
  endlayout; /* end of overlay */
  layout overlay / (options);
    BarChartParm X=avisit Y=Naes1 / name = "stack1" legendlabel ="Treatment 1" ;
    BarChartParm X=avisit Y=Naes2 / name = "stack2" legendlabel ="Treatment 2" ;
    DiscreteLegend "stack1" "stack2" / location=OUTSIDE title="Treatment Groups" ;
  endlayout; /* end of overlay */
endlayout; /* end of lattice */


```

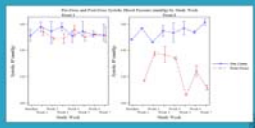
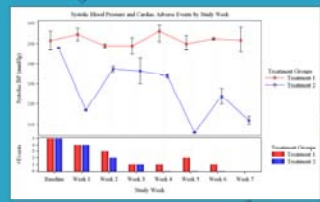



Here we do a lattice of graphs that are aligned on the x-axis rather than the y-axis. To do the 'lattice' or aligned grid of graphs, we do the following:

- Make the major layout block a 'LAYOUT LATTICE'.
- in the layout statement, we define the number of rows and columns of graphs to be generated. In this case we will have one graphs across the page (columns) and two graphs stacked vertically (per 'row').
- in the layout statement, we specify columndatarange=unionall, to say that the graphs within this block should share a common x axis scale.
- The ROWWEIGHTS option allocates 75% of the vertical (row) space to the first graph (the line plot) and 25% to the second graph (the bar chart).
- Note that since one of the graphs is a bar chart, which needs a DISCRETE axis, we will also use a DISCRETE axis for the line graph as well.
- We want the legend for each graph to have a its own legend. Therefore we put the legend statements within the LAYOUT for each individual graph.

Many Things? One Big Thing? – Either Way!





Questions? Contact:
lcollins@pharmastat.com

So, whether you want to produce many things or one big thing, you can split the difference without taking your lumps.
(Groan).