Paper 234-2012

SG Techniques: Telling the Story Even Better!

Chuck Kincaid and Jack Fuller, Experis, Kalamazoo, MI

ABSTRACT

The SAS® Statistical Graphics (SG) procedures: SGPLOT, SGPANEL, SGSCATTER, and SGRENDER are exciting additions first put into production in SAS 9.2 that give easy access to some of the power of the Graphics Template Language (GTL). As good as these procedures are in helping you tell your analytical story, and they are good, there are techniques that can add extra value to your graphs.

This paper will discuss four concepts introduced by William S. Cleveland in his book Visualizing Data – level ordering, slicing, banking, and stacking. For these techniques, we will explain the concepts, provide examples and outline the basic algorithms in this paper. The audience for this presentation is the statistician or business analyst who wants to tell their story even better.

INTRODUCTION

By now, many have seen and, we hope, used the SAS ODS Graphics® procedures, formerly known as SAS/GRAPH Statistical Graphics procedures. These procedures are easy to use for straightforward, professional quality graphics and, at the same time, can be used to build multi-dimensional graphics that convey complex information. This paper discusses four techniques that are not yet part of the SG capabilities. If you would like to see them incorporated, let the SAS Developers know.

Graphics are created to tell a story visually according to the analysis undertaken. These four techniques – ordering, slicing, banking, and stacking – help to tell that story better. Until they are incorporated into the SG procedures directly, each one requires preparation of the data creating new, likely temporary, datasets. These datasets are then used with the respective SG procedure. Each of these algorithms has been turned into a macro that is available upon request.

All four concepts can be found in Bill Cleveland's book, *Visualizing Data*. This seminal book was developed while he was at Bell Labs and published in 1993. He has many other exciting ideas in there and it is well worth a read.

ORDERING LEVELS AND CELLS

Cleveland introduced the coplot in his book, *Visualizing Data*, and it has become widely used as a way to tell a story. The examples first used were based on trivariate data, that is, measurements of three quantitative variables. These displays had a natural ordering on each axis and for the cells. The "given levels" were created using a technique called *slicing* that we'll discuss below. The move to multiway data, that is, one quantitative response variable and two categorical variables does not necessarily provide a natural ordering for the axes and for the cells. However, imposing an ordering on the categorical variables based on a quantitative variable of interest can help in eliciting even more information from presentation. Let's look at the multiway example that Cleveland begins with.

Paper #234-2012, continued

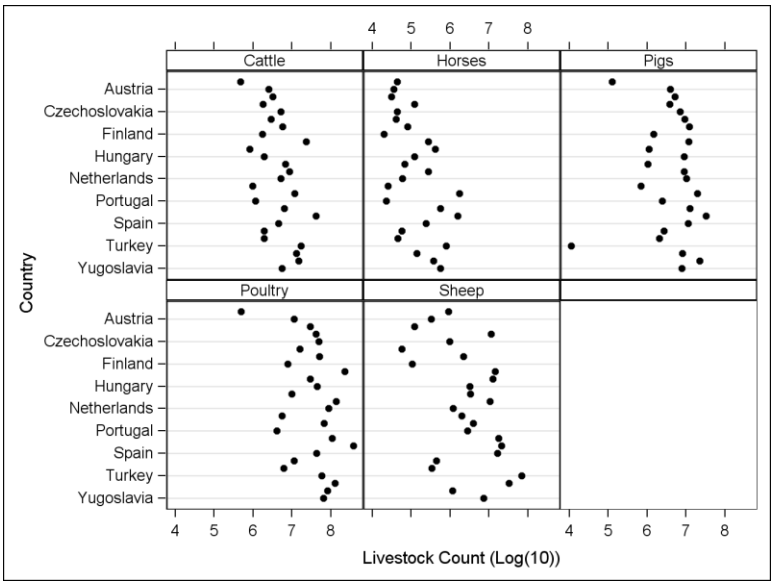

Figure 1

Figure 1 above "graphs the logarithms of livestock counts from a 1987 census of farm animals in 26 countries. … Russia *et al*. is the European part of Russia and the European countries that were formerly part of the Soviet Union." The data is graphed with a dot plot and the quantitative variable is displayed as log count because of the high variability in the counts. The default ordering of the countries on the vertical axis is alphabetical, and the order of the cells from upper left to lower right is alphabetical by Livestock Type, as well. Alphabetical ordering provides no information towards our understanding of the data. (Note that not all countries show on the axis because of the size of the graph. An option in SAS 9.3 gives the programmer control over this display.)

The data in this presentation appear very unassuming. The number of pigs in Turkey seems unusual, but otherwise there is not a lot that jumps out at us. However, if we order the levels of the class variable in the dot plots (country), then more information can be found.

The following table shows COUNTRY ordered by its median livestock count across all livestock type.

Country	Median	Country	Median
Albania	478,000	Czechoslovakia	5,131,000
Norway	971,000	Netherlands	5,241,000
Greece	1,107,000	East Germany	5,690,000
Finland	1,475,000	Yugoslavia	7,384,000
Sweden	1,902,000	Italy	8,928,000
Switzerland	1,954,000	Spain	11,263,000
Portugal	2,448,000	Poland	11,912,000
Austria	2,536,000	Romania	12,464,000
Denmark	2,873,000	United Kingdom	13,155,000
Hungary	3,183,000	France	14,346,000
Belgium	3,246,000	West Germany	15,098,000
Ireland	3,323,000	Turkey	16,983,000
Bulgaria	3,808,000	Russia et al.	33,100,000

This effectively converts COUNTRY from a nominal variable to an ordinal variable and gives a rudimentary scatterplot.

Paper #234-2012, continued

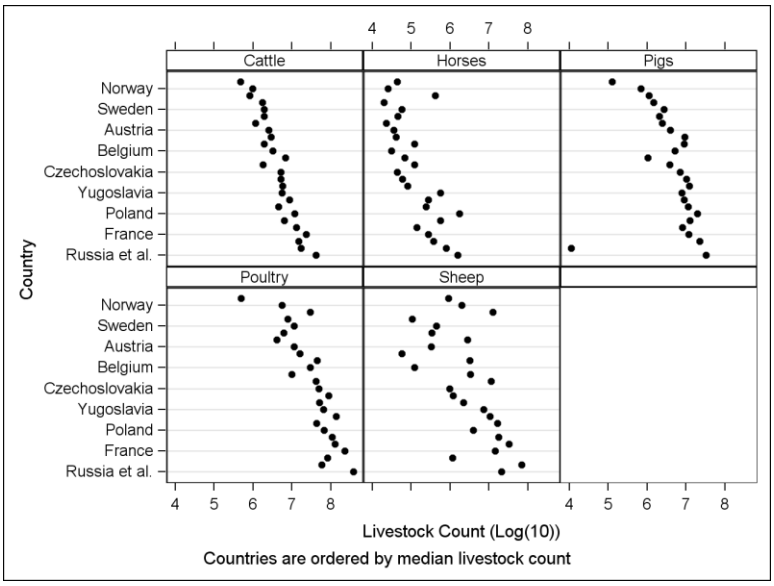

Figure 2

The new plot, Figure 2 above, tells the story better than the previous plot by giving us more information. We see that the number of Cattle and Pigs are strongly associated with the median number of livestock, and Sheep are weakly associated. The low value for Pigs in Turkey stands out even more than before, since it is now placed as the country with the second highest median number of livestock.

The algorithm for the *levelOrdering.sas* macro is as follows

1) Compute the class ordering
    a. Use The MEANS Procedure to calculate the statistic for the levels of the class variable.
    b. Sort the PROC MEANS output according to the calculated statistic. The observation number will be the value of the new class variable.
2) Build the format for the new class variable.
    a. This format will associate the text values of the original class variable with the numeric values of the new class variable.
    b. Create a CTLIN data set to use with PROC FORMAT.
3) Create the final data set.
    a. The new class variable can be used to sort the records according to their value of the statistic. It will display according to the values of the original class variable via the format above.

The SAS code that generated Figure 2 is

```
%levelOrdering (
  dsIn         = livestock,     /* Input data set */
  dsOut        = livestock2,    /* Output data set */

  oldClassVar = country,        /* Old class variable */
  newClassVar = countryNum,     /* New class variable */

  stat         = median,        /* Statistic */
  statVar      = count,         /* Statistic variable */
  fmtName    = cty_fmt          /* Format name */
  );

ods graphics / imagename="multiway ordering";

footnote1 "Countries are ordered by median livestock count";
proc sgpanel data=livestock2 description="multiway ordering";
    panelby livestock_type / novarname ;
    dot countryNum / response=count;
```

```
    colaxis type=log logbase=10 logstyle=logexponent alternate label="Livestock
Count (Log(10))";
    rowaxis fitpolicy=thin label="Country";
run;
```

This concept can also be used for ordering the cells in the panel and other places a class variable is used.

SLICING

As mentioned above, Cleveland started his examples with trivariate data, i.e. three quantitative variables. In order to do so, he needed a way to slice one of the variable's values into segments. Each segment defined the data points of the other two variables that would be included in the plot. In his book, Cleveland defines the *equal-count algorithm*, which sliced the data according to user provided criteria keeping two objectives in mind. First, to have, as much as possible, each of the intervals contain the same number of values. The second objective is to have "the fraction of values shared by each pair of successive intervals as nearly equal to the target fraction as possible."[1] The overlap of data from interval to interval is designed to smooth out the changing of the intervals. Without overlap the changes in the dependent panels from cell to cell would be choppier. We want to see how the graphic in the cells changes as we move along it in a smooth manner.

Consider watching the scenery while riding in a car. Suppose you were to look at the scenery on your stretch of the road, take a 15 minute nap, wake up and look at it again. If you're on a highway in a place like western Kansas or northern Michigan, then that may be enough, since the view does not change that much. However, as the scenery changes more rapidly or the route gets more complex, your naps will need to be shorter and shorter, until you have to pay attention all along the way. As you pay more attention, the overlap in what you see becomes greater and greater.

In Cleveland's algorithm, the user selects the number of intervals in which to slice the data and the target fraction of data points shared by successive intervals. There is an obvious tradeoff between the number of intervals, the number of points in each interval and the amount of overlap in the intervals. Our goal is to have enough data plotted in each cell to see the underlying relationship that we are exploring, but not so much so that we have not gained anything by slicing. We want enough intervals so that the changes in the relationships being explored (histograms, loess plots, etc.) can be seen in each plot without being masked by data from other surrounding plots. Accomplishing this goal is a trial and error process, like much of exploratory data analysis, and is left to the reader.

Let's look at the algorithm now, again translated from Cleveland's book.

Consider N univariate data points that we are making into buckets, sorted in ascending order as $(x_1, x_2, x_3, ..., x_N)$.

Each of the intervals we determine will have endpoints as values of x_i. Let

1) $k =$ number of conditioning intervals
2) $f =$ target fraction of data points shared by successive intervals
3) $l_j =$ lower endpoint of the j^{th} interval; Note that $l_1 = x_1$.
4) $u_j =$ upper endpoint of the j^{th} interval; Note that $u_N = x_N$.

Then, the number of values in each interval can be determined as…

5) $$r = \frac{N}{k(1-f)+f}$$

Note that r does not have to be an integer, which will be accounted for later. Note also that as k, the number of intervals, or slices, increases, then r, the number of values in each interval, decreases. In addition, as the target fraction, f, increases the number of values in each interval increases. Both of these relationships make intuitive sense. From this we calculate, that the index of the lower endpoint for the j^{th} interval is

6) $$1+(j-1)(1-f)r,$$

rounded to the nearest integer. Similarly the upper endpoint of the j^{th} interval is

7) $$r+(j-1)(1-f)r$$

rounded to the nearest integer.

Consider an example using Cleveland's rubber data. Following Cleveland, we have selected 6 intervals and a target fraction of 3/4.

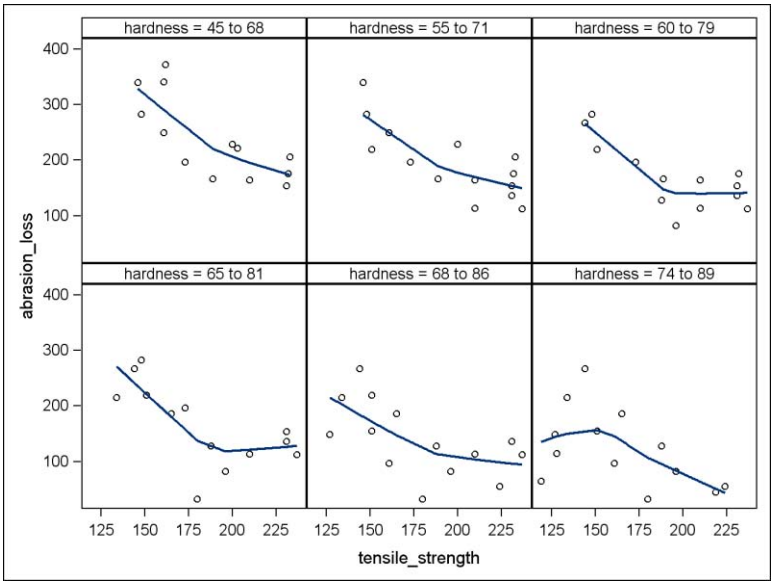

Figure 3

Figure 3 above was created with the following call to the macro and the SGPANEL procedure.

```
%slicing (
  dsIn          = rubber,             /* Input data set */
  dsOut         = rubber2,            /* Output data set */

  rankVar       = hardness,           /* Ranking variable */
  rankVar2      = tensile_strength,/* Secondary ranking variable (to handle ties) */
  groupVar      = slice,              /* New grouping variable */

  noIntervals   = 6,                  /* Number of intervals */
  overlapRatio  = .75                 /* Overlap (0<=overlapRatio<1) */
  );

proc sgpanel data=rubber2 noautolegend;
    panelby slice ;
    loess x=tensile_strength y=abrasion_loss;
run;
```

The macro, *slicing.sas*,

1) Calculates the indices according to the algorithm above
2) Creates an output data set with redundant records and a class variable, SLICE
3) Applies a format and a label to the class variable so that the output shows the values of the sliced variable, e.g. "hardness = 65 to 81"
4) The user can also provide a name and a library for the format.

Slicing is a great technique for creating panels according to the levels of a quantitative variable. The values for number of intervals and target fraction are often found by trial and error over a range for each.

BANKING

Banking is a technique "designed to maximize the discriminability of the orientations of the line segments in the chart."[2] Edward Tufte also advocates banking in his book *Beautiful Evidence* with the following…

Paper #234-2012, continued

"… as William Cleveland discovered, for judging slopes and velocities up and down hills in time-series, best is an aspect ratio that yields hill-slopes averaging 45°, over all the cycles in the time-series. That is, variations in slopes are best detected when the slopes are around 45°, uphill or downhill. ... the aspect ratio should be such that the time-series graphics tend toward a lumpy profile rather than a spiky profile .. or a flat profile."[3]

The aspect ratio of a graph in Cleveland's terminology is the ratio of the vertical scale (height) to the horizontal scale (width). For SAS procedures it is the opposite, that is, the ratio of the width to the height. However, in SAS we specify the height and width rather than directly specifying the aspect ratio. Therefore, we will use Cleveland's terminology in our discussions.

To better understand the importance of the aspect ratio, consider the plot in Figure 4 based on one from Cleveland's book.

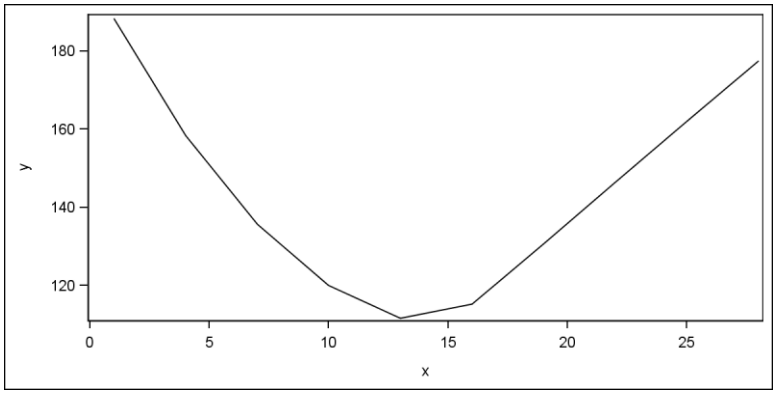

Figure 4

This aspect ratio clearly shows the curvature in the first half of the line. If we use the aspect ratios of 5 and 0.05, though, we get the following two plots, Figure 5 and Figure 6, respectively.

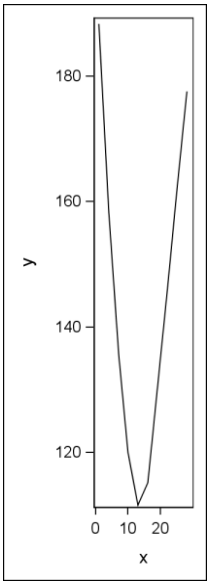

Figure 5

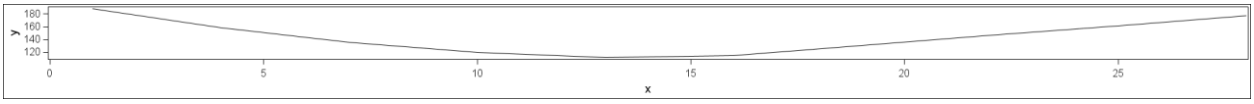

Figure 6

Paper #234-2012, continued

That curvature is more obscured in these figures. To be fair the example is somewhat contrived. If we consider the default settings for SGPLOT, we get the following figure, Figure 7, in which the curvature is also readily apparent.

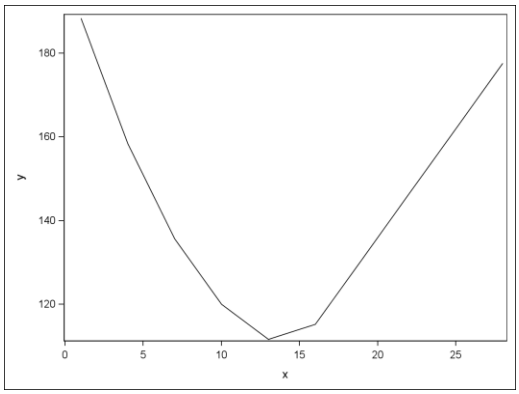

Figure 7

The aspect ratio can be controlled for ODS Graphics by the ODS GRAPHICS statement and the WIDTH= and HEIGHT= options. Starting in SAS 9.2 (TS2M3) there is an option in GTL called ASPECTRATIO that can be used with the LAYOUT OVERLAY to control the ratio of height to width of the plot wall area. To limit space, because it's not the primary point, and because most people will not be using GTL, we will discuss only the first. An example of using GTL can be found in a blog post by Prashant Hebbar. [4] It is also interesting to note that Hebbar uses a different method to determine the aspect ratio. In fact, there are many methods to calculate the aspect ratio. Heer and Agrawal [2] discuss 12 of them including ones they propose that take into account the frequency scales on which the data is most interesting. Their paper shows the variation in the aspect ratios derived from these methods.

The SAS default value for the aspect ratio is 0.75 (4/3, in SAS terminology) meaning that the height (480 pixels by default) is 75% of the width (640 pixels by default). If you specify only one of these two, then the other is automatically adjusted so that the default aspect ratio will be maintained. Most of the time this is the appropriate course of action, since, in many cases, the graph might have a specific layout. "For example, a plot that has multiple columns, or that has a statistics table on the side needs a wide aspect ratio. Changing the aspect ratio for this plot by specifying both width and height might produce unpredictable results."[5] However, for our purposes, this is exactly what we want to do.

Note that at this time, we will only consider changing the aspect ratio for the SGPLOT. The banking technique is very appropriate for multi-cell panels, where what we will see below for one cell is extended for all cells as if there were only one. All cells would then be set to the same aspect ratio. Because many factors make setting this aspect for multi-cell panels difficult, in this paper we will not consider the situation.

Another point to realize is that these options set the height and width of the entire plot and not just the data area. Whether the data area has the same aspect ratio depends on factors such as the size of the tick values, and whether and where a legend appears. Unfortunately, at this time, the SG procedures, nor GTL, provide more control. If this is of interest, let SAS know.

The idea behind banking is to consider the line segments defined by each pair of data points, their orientation and their length. First, we define some terms. Suppose we have N data points, (X_1,Y_1) … (X_N,Y_N). Then

8) h Horizontal length of the data in physical units (e.g. inches, cm)

9) v Vertical length of the data in physical units (e.g. inches, cm)

10) $h_i(h)$ Change in physical units when the horizontal length of the data rectangle is *h*

11) $v_i(v)$ Change in physical units when the horizontal length of the data rectangle is *v*

12) $a(h,v) = v/h$ Aspect ratio of the graph

13) $h_{span} = X_N - X_1$ Horizontal span of the data in data units (e.g. $, yrs, ft)

14) $v_{span} = Y_N - Y_1$ Vertical span of the data in data units (e.g. $, yrs, ft)

Paper #234-2012, continued

15) $\ddot{h}_i = |X_{i+1} - X_i|$ Absolute horizontal distance of the i[th] line segment in data units

16) $\ddot{v}_i = |Y_{i+1} - Y_i|$ Absolute vertical distance of the i[th] line segment in data units

17) $\bar{h}_i = \ddot{h}_i / h_{span}$ Relative width of the i[th] line segment

18) $\bar{v}_i = \ddot{v}_i / v_{span}$ Relative height of the i[th] line segment

19) $\theta_i(h,v) = \arctan(\frac{v_i(v)}{h_i(h)}) = \arctan(a(h,v)\bar{v}_i / \bar{h}_i)$ Orientation of the i[th] line segment

20) $l_i(h,v) = \sqrt{h_i^2(h) + v_i^2(v)} = h\sqrt{\bar{h}_i^2 + a^2(h,v)\bar{v}_i^2}$ Physical length of the i[th] segment

To illustrate these concepts, consider the 5 points below. The horizontal data span is h_{span} = 8 (= 10 – 8) and the vertical span is v_{span} = 10 (= 18 – 8).

Data Point (X_i,Y_i)	Absolute Horizontal Data Distance $(\ddot{h}_i)$	Absolute Vertical Data Distance $(\ddot{v}_i)$	Relative Width $(\bar{h}_i)$	Relative Height $(\bar{v}_i)$
(2,8)				
(6,12)	|6 – 2| = 4	|12 – 8| = 4	4/8	4/10
(8,9)	|8 – 6| = 2	|9 – 12| = 3	2/8	3/10
(9, 16)	|9 – 8| = 1	|16 – 9| = 7	1/8	7/10
(10, 18)	|10 – 9| = 1	|18 – 16| = 2	1/8	2/10

Our focus will be how the aspect ratio affects the mean of the absolute orientations weighted by the line segment lengths. The optimal aspect ratio is one for which that mean is 45°. (As mentioned above, there are other criteria for finding the optimal aspect ratio and we will only look at this one.) The mean is

$$\frac{\sum_{i=1}^{N}\theta_i(h,v)l_i(h,v)}{\sum_{i=1}^{N}l_i(h,v)} = \frac{\sum_{i=1}^{N}\arctan(a(h,v)\bar{v}_i/\bar{h}_i)\sqrt{\bar{h}_i^2 + a^2(h,v)\bar{v}_i^2}}{\sum_{i=1}^{N}\sqrt{\bar{h}_i^2 + a^2(h,v)\bar{v}_i^2}}$$

This mean depends on *v* and *h* only through $a(h,v)$, that is, the aspect ratio. As stated, then, the optimal aspect ratio is one that makes this mean equal to 45°. Fortunately, the weighted mean is a monotone function of $a(h,v)$, since there is no closed-form solution, and typically only a few iterations are needed to find a solution. In SAS this ratio can be found using the OPTMODEL procedure with an objective function of

```
min bank_eq =
  (
    (
      sum{k in indx}
           atan(ar*vibar[k] / hibar[k]) * sqrt(hibar[k]**2 + ar**2 * vibar[k]**2)
    ) / (
           sum{k in indx} sqrt(hibar[k]**2 + ar**2 * vibar[k]**2)
         ) - &orient
  )**2
     ;
```

Paper #234-2012, continued

Minimizing the square of the equation rather than the absolute value avoids certain issues in solving the equation.

The macro *banking.sas* would be called as

```
%banking (
  dsIn       = sunspot,        /* Input data set */
  dsOut      = sunspot2,       /* Output data set */

  widthVar    = year,          /* Width variable */
  width       = 8,             /* Physical width */
  mvarWidth   = myWidth,       /* Name of macro variable which will contain new
physical width */

  heightVar  = number,         /* Height variable */
  mvarHeight = myHeight,       /* Name of macro variable which will contain new
physical height */

  mvarAspect = myAspect        /* Name of macro variable which will contain the
aspect ratio */
  );
```

It works as follows:

1) Expects user input on (among other things),
    a. either the desired height or the desired width, but not both. In our example, we specified the desired width.
    b. the names of the macro variables in which the results will be returned
2) Calculates the terms indicated in the table above
3) Runs PROC OPTMODEL with the objective function above to determine the optimal aspect ratio
4) Calculates the new width or height, depending on which was entered, based on the optimal aspect ratio
5) Populates the specified macro variables with the width and height

Note that the macro variables in step 1b must be defined ahead of time, which in our example would be simply

```
%let myAspect = ;
%let myWidth = ;
%let myHeight = ;
```

The programmer can then use these macro variables in a statement like

```
ODS GRAPHICS / HEIGHT=&myVarHeight. WIDTH=&myVarWidth.;
```

This will change the aspect ratio of the (entire) graph area for all plots going forward. To reset the height and width to their defaults, the programmer would execute.

```
ODS GRAPHICS / RESET=all;
```

Because we do not have specific enough control, the height and width values may need to be modified to get the desired aspect ratio in the data area. In particular, one might need to account for the axes and legends.

The traditional example for banking, which works well because of the large difference between the optimal aspect ratio and the default aspect ratio, is the sunspot data. The default plot, Figure 8, provided by this code

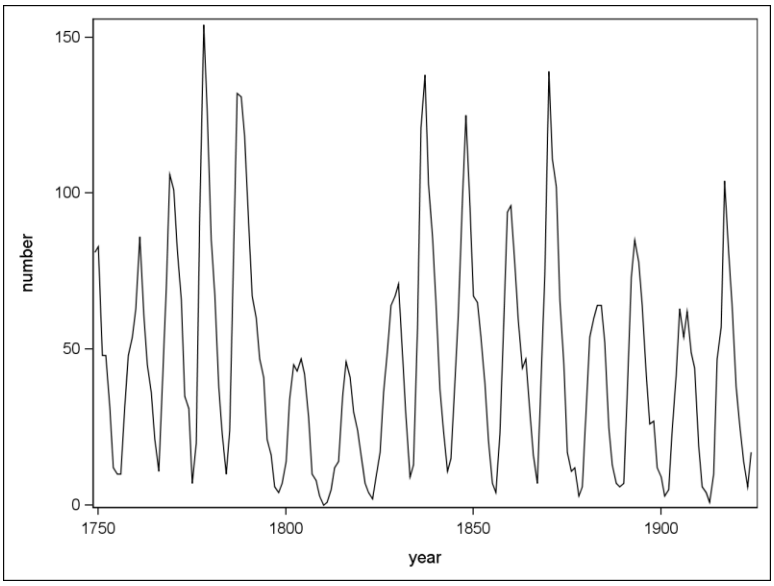

Figure 8

While this view is okay, let's look at it with the line segments banked to 45° as in Figure 9.

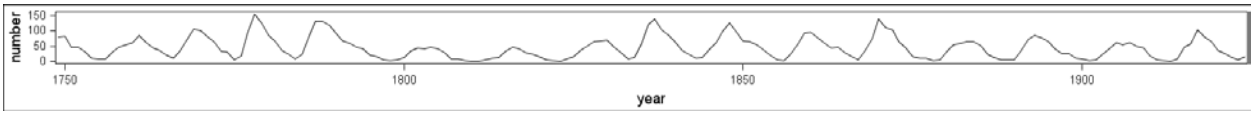

Figure 9

It is easier to compare the line segments to one another in this plot. We can now see that sunspot activity typically increases at a faster rate than it decreases.

Banking, as it is, is not as valuable as it could be. If there is enough interest and SAS adds it as an option to SGPLOT and SGPANEL (and the newer ASPECTRATIO option in GTL indicates they are working towards it), then it will be even more so. Note also, that there can be good reasons why the "optimal" aspect ratio is not the desired one. The programmer should use their best judgment.

STACKING

When plotting very long series of data, one must choose between very wide plots or very scrunched data. Or at times banking the data to 45° requires a very small aspect ratio which would require a very wide plot. In these situations, a standard plot may not be ideal. As an example, we will use seismograph data of the Kobe earthquake, recorded at Tasmania University, Hobart, Australia.[6] Here is a standard plot of the data using SGPLOT.

Paper #234-2012, continued

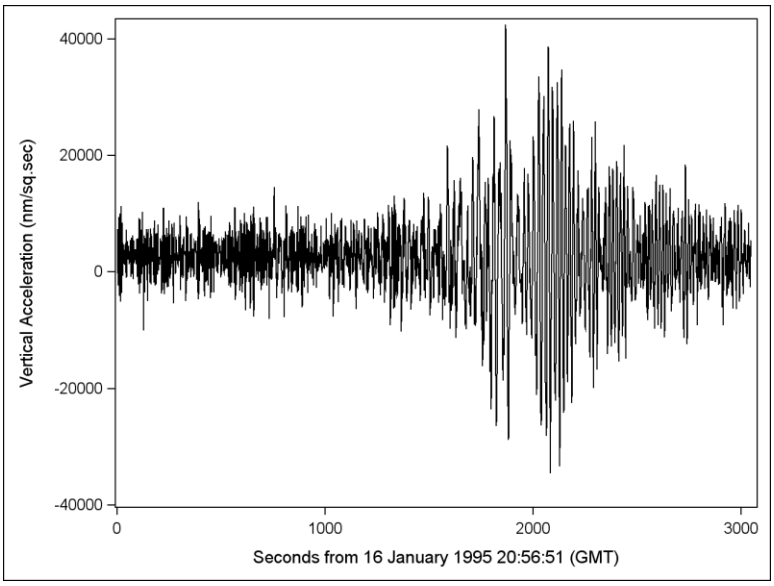

Figure 10

Suppose we want to see more detail in the data. The stacking technique was designed to help us do so. Stacking the plot cuts it into pieces and stacks them on top of one another, as in the figure below.

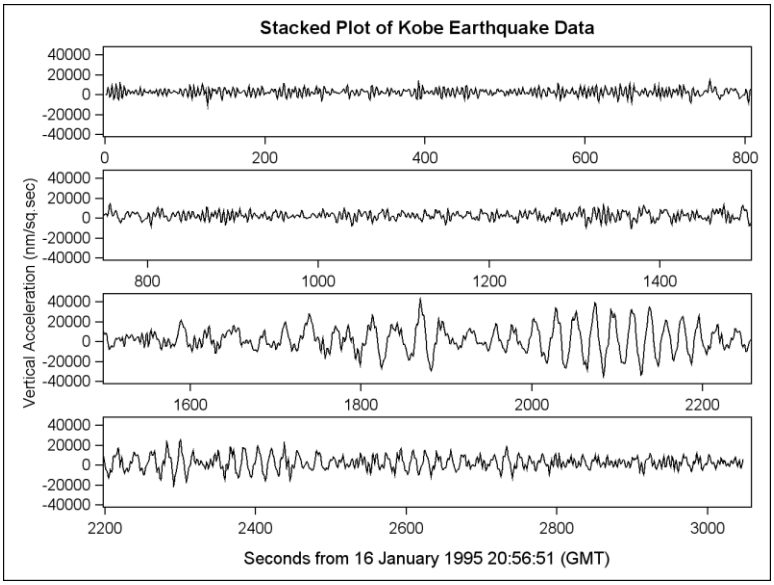

Figure 11

This plot gives more room for the horizontal axis and keeps the vertical axes uniform, so that now we can see more details in each section of the data. This type of plot cannot be drawn with the SGPLOT or SGPANEL procedures. One might be able to mimic this by categorizing the data on the horizontal axis, in this case SECONDS, into four groups and rescaling each to the same range. However, we prefer this method as it keeps the original scales.

Implementing stacking in ODS Graphics requires the use of GTL, since we are stacking four essentially different plots. The method we chose has two steps. First, the user runs SGPLOT for the kind of plot they want. This can include almost any kind of SGPLOT feature, such as multiple plots and reference lines. The TMPLOUT option is used to export the GTL code to a separate SAS file as shown below.

```
proc sgplot data=kobe tmplout="c:\mypath\kobe_ex.sas";
    series x=second y=vertacc ;
```

Paper #234-2012, continued

```
    refline 1000 / axis=x;
run;
```

The resulting GTL file, *kobe_ex.sas*, looks like the following.

```
proc template;
    define statgraph sgplot;
    begingraph /;
          layout overlay;
             SeriesPlot X=second Y=vertacc / primary=true LegendLabel="Vertical
                 Acceleration (nm/sq.sec)" NAME="SERIES";
             ReferenceLine x=1000 / clip=true;
          endlayout;
    endgraph;
    end;
run;
```

The parts we want out of this code are, in this case, the two plot statements.

```
SeriesPlot X=second Y=vertacc / primary=true LegendLabel="Vertical Acceleration
    (nm/sq.sec)" NAME="SERIES";
ReferenceLine x=1000 / clip=true;
```

For this example, the *stacking.sas* macro is called in the following manner.

```
%macro stacking (
    inDS     = kobe,                 /* Input data set */
    inTempl  = kobe_ex.sas,          /* Input template */
    outPath  = c:\mypath,            /* Output file */
    xVar     = second,               /* X variable */
    minXVal  = ,                     /* Min X value */
    maxXVal  = ,                     /* Max X value */
    nPlots   = 4                     /* Number of plots */
);
```

The macro creates a new GTL file with the following algorithm

1) Reads in the GTL file, *kobe_ex.sas*
2) Creates the template structure to handle multiple graphs, in our case NPLOTS=4
3) Duplicates the plot statements within that structure
4) Adds the required axes options as described below
5) Adds entries for titles, footnotes, Y axis label, X axis label, and legend, as appropriate

Customizations of the axes in the original SGPLOT will show up in the LAYOUT OVERLAY statement as options after a "/". However, to avoid complex parsing issues the current version of the macro uses a predefined LAYOUT OVERLAY statement so that we can add our own options. Any user customizations of the axes, such as a GRID option, will be lost. However, the user can edit the output GTL code to add these customizations back in.

The axis statements placed in the output file by the macro for the top stacked plot are

```
xaxisopts=(
    display=( ticks tickvalues line )
    type=linear
    linearopts=( viewmin=0 viewmax=750 Integer=true ) )
yaxisopts=(
    display=( ticks tickvalues line )  )
```

The DISPLAY option provides standard axes without labels. The TYPE option in these statements specifies whether the axis is plotting a LINEAR (standard numeric) or TIME variable. The user specifies which they are using via the XAXISTYPE macro parameter. LINEAROPTS, or TIMEOPTS as appropriate, specifies the minimum and maximum value of the data shown on the horizontal axis. ODS Graphics does not necessarily follow these limits exactly as Figure 11 shows, giving overlap between the plots, which seems beneficial to us.

The range of values for each horizontal axis is determined by the range of values for the entire plot divided by the number of plots being stacked. The default range of values for the entire plot is the minimum and maximum data value of the X variable. If desired the user can restrict the plot to a subset of the data's range with the MINXVAL and MAXXVAL macro parameters.

If a title were used, then it would show up in *kobe_ex.sas* in an ENTRYTITLE statement. This statement is used as is in the output code, for example,

```
entrytitle "Stacked Plot of Kobe Earthquake Data";
```

The vertical axes label is placed in a SIDEBAR statement. If we used the default label position for the vertical axis, then there would be, in this case, four labels, which are redundant and may not be entirely visible in the space available. The SIDEBAR centers the label across all plots much like SGPANEL would do. The horizontal axis label could be left on the lowest plot, but that takes away from the plot area so that the lowest plot is shorter than the others are. Our decision was to use the ENTRYFOOTNOTE statement to place the label below the entire plotting area, though a SIDEBAR statement could be used.

```
entryfootnote halign=center "Seconds from 16 January 1995 20:56:51 (GMT)" ;
```

Finally, if there is a legend, then the GTL code for the legend is used verbatim in another SIDEBAR, which the macro places across the top under the title so that it is out of the plot area. The user can modify the GTL code to change this location.

The macro then executes the PROC TEMPLATE code to define the template and calls PROC SGRENDER specifying the template and the data to render the output. If our data is KOBE and the template is called STACKED, then the SGRENDER call is simply

```
proc sgrender data=kobe template=stacked;
run;
```

The user will have to determine the best number of plots to stack in their situation, but going beyond four plots seems to reduce the space available for each plot too much.

CONCLUSION

The Statistical Graphics procedures are a wonderful addition to the programmer's toolkit. They expand the programmer's ability to understand the stories that are in the data and then create persuasive graphs to share their work with others. The techniques described here are helpful additions to that toolkit.

The macros described in this paper are available by contacting the author. They are provided as is and we ask that you retain the header documentation when you use them. Otherwise, please use them as much as you'd like and we hope that they help you to tell the story better.

REFERENCES

[1] Cleveland, William S., Visualizing Data, Hobart Press, 1993

[2] Heer, Jeffrey and Maneesh Agrawala, Multi-Scale Banking to 45°, IEEE Transactions on Visualization and Computer Graphics, Vol 12, No. 5, September/October 2006

[3] Tufte, Edward, Beautiful Evidence

[4] http://blogs.sas.com/content/graphicallyspeaking/2011/12/01/graphs-you-can-%E2%80%98bank%E2%80%99-on-with-aspect-ratio/

[5] SAS Help Documentation: Managing Your Graphics With ODS: Using the ODS GRAPHICS Statement.

[6] Hyndman, R.J. (n.d.) Time Series Data Library, http://robjhyndman.com/TSDL. Accessed on November 29, 2011.

ACKNOWLEDGMENTS

The first author would like to thank the second author, Jack Fuller, for his amazing programming expertise in creating the macros for publication. In addition, we acknowledge the R&D staff at SAS who developed these procedures and are very willing to talk about them at any time.

CONTACT INFORMATION

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

Chuck Kincaid

Paper #234-2012, continued

Experis
5220 Lovers Lane
Portage, MI 49002
(269) 553-5140
chuck.kincaid@experis.com
www.experis.us/analytics