

Paper 262-2012

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize

Perry Watts, Stakana Analytics, Elkins Park, PA
Nate Derby, Stakana Analytics, Seattle, WA

ABSTRACT

In many of the SAS Institute publications about the new ODS statistical graphics, there is an introductory statement that defines an “effective” graph as one that reveals “patterns, differences and uncertainty that are not readily apparent in tabular output” ([Kuhfeld, 2010](#); [Rodriguez, 2008](#); [Rodriguez and Cartier, 2009](#)). Developing a good graph becomes a challenge, however, when input data map to crowded displays with overlapping points or lines. Such is the case with the Framingham Heart Study of 5209 subjects captured in the `sashe1p.heart` data set and a series of 100 cumulative booking curves for the airline industry. In addition, interleaving time series plots can be difficult to interpret, and patterns can be missed when lattice plot panels are charted out-of-order. In the paper, transparency, layering, data point rounding, median calculation, and color coding are among the techniques that are evaluated for their effectiveness to add visual clarity to graphics output. The following Graph Template Language (GTL) statements are referenced in the paper: `ENTRY`, `HISTOGRAM`, `SCATTERPLOT`, `LINEPARM`, `REFERENCeline`, `BANDPLOT`, and `SERIESPLOT` plus layouts `OVERLAY`, `DATAPANEL`, `LATTICE`, and `GRIDDED`. GTL is chosen over SG PROCEDURES because of its greater graphics capability.

KEY WORDS: Graph Template Language, SG Procedures, ODS Statistical Graphics, Version 9.2 SAS®.

SAS code and data itemized at the end of the paper can be downloaded from the ODS Statistical Graphics folder on <http://www.PerryWatts.org>. Just select the paper by title and scroll down to the attachments tab.

INTRODUCTION AND DESCRIPTION OF THE DATA THAT SUPPORT THE GRAPHS

An incremental approach is taken in this paper. We go from preliminary graphs that are less than optimal to output that conveys its message more effectively. Problems are defined along the way, and suggestions for solving them take advantage of the new features available in ODS statistical graphics. Source code is integrated into the discussion about the graphs; however the intention here is not to present a tutorial on the Graph Template Language. Instead, the paper focuses on how to graph dense and challenging data sets. For a discussion about an effective programming strategy to adopt when coding in GTL see [Yang \(2011\)](#).

Four data sets motivate the graphs presented in this paper. The first is the `sashe1p.heart` data set that comes from the Framingham Heart Study of 5,209 men and women that have been followed biennially for the development of cardiovascular disease since 1948 ([Hubert et al., 1983, p.968](#)). The second data set from the airlines industry contains coordinates for an ordered progression of 100 time series plots; about 97 more than are typically graphed. The progression is ordered by date of departure which spans 100 consecutive Wednesdays between 12/10/08 and 11/03/10 ([Derby and Vo, 2010](#)). Each plot in the progression is a cumulative distribution of booking lead times measured in days prior to the scheduled date for departure. What the graph is attempting to do is to trace the relationship between booking lead times and flight departure dates.

Unlike the airlines data, there is no inherent order provided by a date variable for plotting barley yields collected from six different sites in Minnesota during the early 1930's. The original data was analyzed by R.A. Fisher in the 40's, but it took another 40 years for William S. Cleveland to invent the dot plot that revealed a serious error in the data. Besides highlighting the data error, Cleveland demonstrates that ordering by median can reveal underlying patterns in the data ([Cleveland, 1993b, pp.328-338](#)). In 9.2 SAS, we look at the surprising challenge that arises when we attempt to order by median.

With the `sashe1p.stocks` data set we return to challenges that arise when time series plots are graphed. In this example, only three series plots have to be accommodated. However, the visual display is still chaotic, because plot lines overlap. Plotting them separately in a `DATAPANEL` plot with three panels is not satisfactory, because the separation of plot lines makes data comparison difficult. To overcome the difficulty, we develop a new interleaving band plot and make pair-wise comparisons sending the output to another 3-paneled display.

HEART DATA: DEALING WITH DATA POINT OVERLAY

INITIAL VERSION:

The original graph of the heart data appears in a publication that introduced Graph Template Language to attendees at the 2009 SAS Global Forum ([Mantagne et al., 2009](#)). The graph displays heights and weights for participants in the Framingham Heart Study as a scatter plot with marginal histograms. The graph highlights the need for dealing with symbol overlap that hides the full data display from the viewer. Cleveland recommends combining unfilled circles as plotting symbols with jittering to create partial overlap in a graphics display that has total overlap ([Cleveland, 1994, p.158](#)). Unfortunately this method won't work in Figure 1, because overlap in the raw data is *both partial and*

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

complete. In this situation, jittering would only move the problem slightly, elsewhere. Therefore, what the SAS developers did was to use filled circles as plotting symbols while setting transparency, a new capability in ODS graphics, to 95%. This method would work well with sparser data. Symbols would simply become progressively darker as overlay increases. However, it doesn't solve the problem here. Outliers are barely visible, and symbol overlay results in a scatter plot that contains a large, undifferentiated dark "blob" in the middle. While the marginal histograms offset the overlay in the scatter plot it is difficult to map histogram frequencies to indefinable grids in the scatter plot. There are also additional problems with this graph that are addressed in the first revision below.

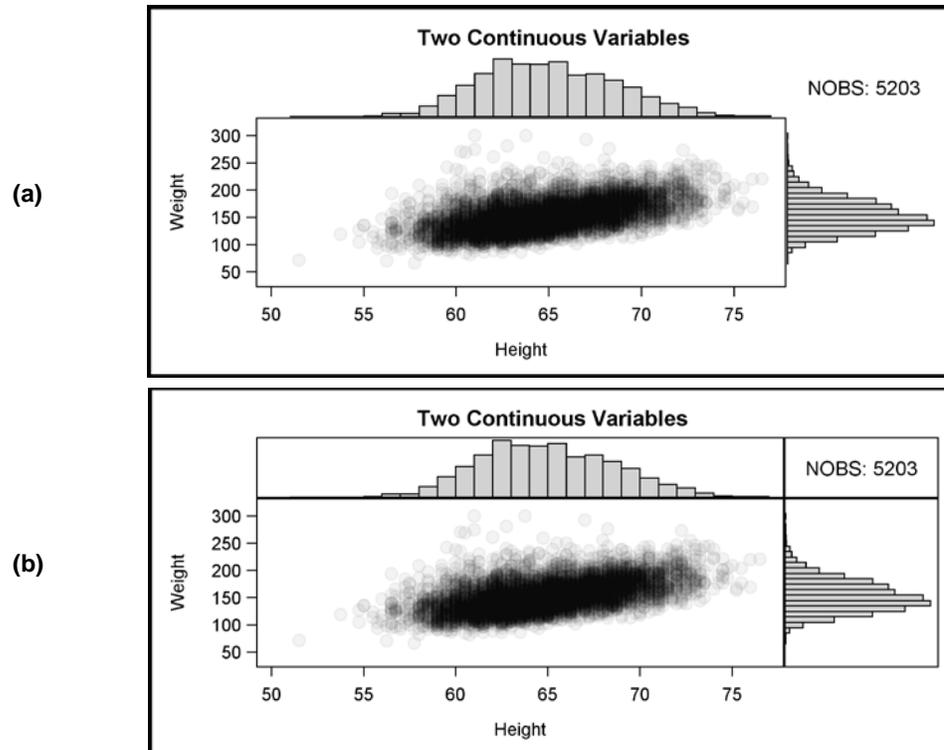


Figure 1. The original graph is stretched out horizontally to increase visibility (a). The `WALLDISPLAY=outline` option is added to show the four cell boundaries in (b).

While source code is fragmentary in much of the paper, `PROC TEMPLATE` and `PROC SGRENDER` from SAS Sample #35172 (SAS Institute, 2009a) used by Mantagne et al. (2009) are reproduced almost in full below. Graphs produced in GTL are defined in a `STATGRAPH` template and generated in `PROC SGRENDER` where `STATGRAPH` template meets input data set. Two types of templates are used in GTL: `STATGRAPH` and `STYLE`. An example of `STATGRAPH` is presented below. `STYLE` is addressed shortly.

GTL works hierarchically and top-down in code blocks to produce a graph. In the source code that follows, the hierarchy is outlined by rectangle overlays:

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

```

PROC TEMPLATE;
  DEFINE STATGRAPH scatterhist;
    DYNAMIC xvar yvar title;

    BEGINGRAPH / DESIGNWIDTH=600px DESIGNHEIGHT=400px BORDERATTRS = (thickness=3px);
      ENTRYTITLE title;

      LAYOUT LATTICE / ROWS=2 COLUMNS=2 ROWWEIGHTS = (.2 .8) COLUMNWEIGHTS = (.8 .2)
        ROWDATARANGE=union COLUMNDATARANGE=union
        ROWGUTTER=0 COLUMNGUTTER=0;

        /* Row#1, Column#1) HISTOGRAM at X2 axis position */
        LAYOUT OVERLAY / WALLDISPLAY = (fill) XAXISOPTS = (DISPLAY=none)
          YAXISOPTS = (DISPLAY=none OFFSETMIN=0);
          HISTOGRAM xvar / binaxis=false;
        ENDLAYOUT; /*OVERLAY*/

        /* Row#1, Column#2) TEXT ENTRY*/
        LAYOUT OVERLAY;
        ENTRY 'NOBS = ' EVAL( N( xvar ) );
        ENDLAYOUT; /*OVERLAY*/

        /* Row#2, Column#1) SCATTER PLOT */
        LAYOUT OVERLAY;
        SCATTERPLOT Y=yvar X=xvar / DATATRANSARENCY=.95
          MARKERATTRS = (SYMBOL=circlefilled SIZE=11px);
        ENDLAYOUT; /*OVERLAY*/

        /* Row#2, Column#2) HISTOGRAM at Y2 axis position */
        LAYOUT OVERLAY / WALLDISPLAY = (fill)
          XAXISOPTS = (display=none offsetmin=0)
          YAXISOPTS = (display=none);
          HISTOGRAM yvar / ORIENT=horizontal BINAXIS=false;
        ENDLAYOUT; /*OVERLAY*/

      ENDLAYOUT; /*LATTICE*/

    ENDGRAPH; /*END GRAPH BLOCK*/
  END; /*END DEFINE BLOCK*/
RUN;
...
PROC SGRENDER DATA=sashelp.heart TEMPLATE=scatterhist;
  DYNAMIC yvar="Weight" xvar="Height" title="Two Continuous Variables";
RUN;

```

Figure 2. The SAS code for Figure 1 comes from [SAS Institute \(2009a\)](#). Code highlighted in blue is reviewed below.

Examining SAS Code at the Block Level: Introducing LATTICE and OVERLAY Layouts

The outermost `DEFINE` block names the `STATGRAPH` template `scatterhist`. `DYNAMIC` variables that make the template re-usable are declared here and assigned values in `PROC SGRENDER`. Next up is the `GRAPH` block where the graph is sized and title is inserted.

The `LAYOUT` block in `scatterhist` supports both `LATTICE` and `OVERLAY` statements. `LAYOUT LATTICE` defines a multi-cell grid of graphs that can automatically align plot areas and tick display areas across grid cells to facilitate data comparisons among plots.... The number of cells must be predefined and *you define the content of each cell separately* ([SAS Institute, 2009c, p.155](#)). (Emphasis is added for later contrast to the `DATAPANEL` statement).

In `scatterhist`, a 2X2 matrix of cells holds four graphs in top down, left to right order. The cells are not equal in size. Instead, row #1 with the top-most histogram and text entry is $\frac{1}{4}$ the height of the plotting region and row #2 with the scatter plot takes up the remaining $\frac{3}{4}$ in height. Column widths also display the same $\frac{1}{4}:\frac{3}{4}$ ratio. In this example, axes dimensions are equalized with `ROWDATARANGE` and `COLUMNDATARANGE` options, although the equality is difficult to see since the corresponding histogram X and Y axes are hidden.

`LAYOUT OVERLAY` is used to fill all four cells in the graph. `LAYOUT OVERLAY` “builds a 2D, single-cell graph by overlaying the results of statements that are contained in the layout block” ([SAS Institute, 2009c, p.39](#)). If multiple plotting statements are issued in a single `OVERLAY` statement, the second plot is placed over the first, the third goes over the second and so on. While axes ranges are defined in `Layout LATTICE`, additional options are specified for the histogram axes in `LAYOUT OVERLAY`. `WALLDISPLAY=` is included as an axis option in `OVERLAY`, because the wall or plotting region is delimited by the rectangle that results when the four axes scale lines X1, X2, Y1 and Y2 are joined ([Cleveland, 1994, p. 33](#)). Wall outlines are removed from both marginal histograms in first graph from Figure 1 when `WALLDISPLAY` is set to `FILL`. By default, the plotting region is outlined in the scatter plot.

The `HISTOGRAM`, `ENTRY` and `SCATTERPLOT` statements are embedded in `LAYOUT OVERLAY` in this graph. The `HISTOGRAM` statement is used to create “a univariate histogram from input data” ([SAS Institute, 2009b, p.309](#)). As

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

explained in many basic statistics textbooks (e.g., [Siegel and Morgan \(1996, p. 29\)](#), [McClave and Sincich \(2003, pp. 31-33\)](#)), a *histogram* displays frequencies (or percentages) of measurements falling into specified intervals as proportional rectangles.

The `ENTRY` Statement “displays a line of text in the plot area” ([SAS Institute, 2009b, p.641](#)) whereas the `SCATTERPLOT` statement is used to create “a scatter plot of input data” ([SAS Institute, 2009b, p.401](#)). As explained by statistical textbooks (e.g., [Siegel and Morgan \(1996, p. 517\)](#), [McClave and Sincich \(2003, p. 86\)](#)), a scatter plot is a two-dimensional plot of the data points, representing two numeric variables. What we will learn when we look at dot plots is that the `SCATTERPLOT` statement is expanded to accommodate nominal data in SAS. The nominal (unordered) character values are simply arranged alphabetically along an axis.

PROBLEMS WITH THE INITIAL VERSION ARE ADDRESSED

Unfortunately the count assigned to `NOBS` is wrong. By definition, points graphed in a scatter plot must have values for both X and Y coordinates that are within axes bounds. Missing values simply cannot be accommodated. When `NOBS` is set to `EVAL(N(height))` for the graph in Figure 1, three individuals with missing weight values are erroneously added to the count. A simple solution to this problem involves setting `NOBS` equal to `EVAL(N(xvar + yvar))`. Using the `+` operator returns a missing value when at least `xvar` or `yvar` are missing; just what we want! This is one instance where the `SUM` function would not give us the results we are looking for.

GTL summary statistic functions are used with `EVAL` such that `NUMBER = EVAL(function-name(numeric-column))`. There are 32 summary statistic functions available in GTL: pretty much a subset of the statistics that are provided by `PROC UNIVARIATE`. A full listing can be found in [SAS Institute \(2009c, pp. 262-263\)](#). In addition to being used for labeling graphs, summary statistic functions can help compensate for the single data set restriction imposed upon GTL from `PROC SGRENDER`. [Kuhfeld \(2010, pp. 30-31\)](#), for example, uses the `MEAN` and `STDERR` functions to define coordinates on the fly for the `ERRORLOWER` and `ERRORUPPER` options in the `BARCHARTPARM` statement. Nevertheless, while `EVAL` can often serve as a workaround, restricting GTL to a single input data set increases the complexity of the data processing task defined as programming strategy step #2 in [Yang \(2011\)](#). We address this issue several more times in the paper.

A second problem with the graph in Figure 1 involves relative bin heights in the two marginal histograms. Since all points in a scatter plot must have values for both X and Y coordinates it follows that total counts for both histograms should be the same. That means bin dimensions for the two histograms should be comparable, but they are not.

There are two reasons why you can't visually compare the marginal histograms in Figure 1. First of all, the plotting region for the scatter plot is not square, and secondly, the hidden axes maximum values are different for the two histograms. For `weight`, the maximum is little less than 15% whereas it is roughly 11% for `height`. In Figure 3, a corrected version of the Figure 1 graph is presented along with a side-by-side version of the marginal histograms.

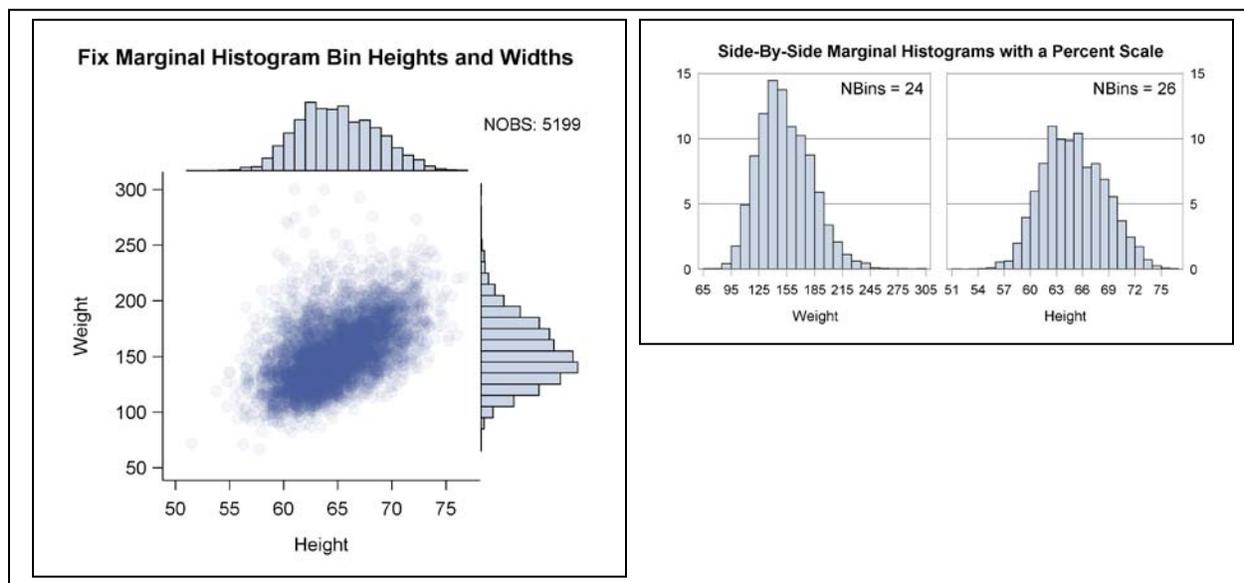


Figure 3. `NOBS` is now correct with 5,199 individuals having both heights and weights. Bin heights and widths in the two histograms are now comparable, since `weight` is taller and slightly wider than `height`.

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

Relevant Code Fragments for Figure 3

Style Changes are made in `PROC TEMPLATE` to remove all wall borders with `FRAMEBORDER`. With their removal, bin ranges in both histograms become more visible.

```
PROC TEMPLATE;
  DEFINE STYLE MyStyle;
  PARENT = styles.statistical;
  CLASS graphWalls /
    FRAMEBORDER=off;
RUN;
```

To find out more about integrating style templates into GTL, see programming strategy step #4 in [Yang \(2011\)](#).

An `AXIS LENGTH` option does not exist in GTL that would guarantee a square plotting region. Instead the whole graph is squared off with `DESIGNWIDTH=DEFAULTDESIGNHEIGHT`. The vertical axis has to accommodate titles, and the more titles, the greater the disparity between axes lengths. Nevertheless this is the best that can be done.

```
PROC TEMPLATE;
  DEFINE STATGRAPH scatterhist;
  DYNAMIC xvar yvar title;
  BEGINGRAPH / DESIGNWIDTH=DEFAULTDESIGNHEIGHT;
  ...
  ENDGRAPH;
END;
RUN;
```

A brief code addition to `YAXISOPTS` and `XAXISOPTS` for `xvar` and `yvar` extends the maximum viewable percent for the response axis in both histograms to 15%.

```
/* For the histogram at scatter plot's X2 axis */
LAYOUT OVERLAY / XAXISOPTS=( DISPLAY=none )
                  YAXISOPTS=( DISPLAY=none OFFSETMIN=0 LINEAROPTS=( VIEWMAX=15 ) );
HISTOGRAM xvar / BINAXIS=false;
ENDLAYOUT;
/* Histogram at scatter plot's Y2 axis */
LAYOUT OVERLAY / YAXISOPTS=( OFFSETMIN=0 DISPLAY=none )
                  Xaxisopts=( OFFSETMIN=0 DISPLAY=none LINEAROPTS=( VIEWMAX=15 ) );
HISTOGRAM yvar / ORIENT=horizontal BINAXIS=false;
ENDLAYOUT;
```

APPLY ROUNDING TO THE SCATTER PLOT SQUARED-OFF VERSION OF THE GRAPH

Now with a squared-off graph that has a smaller plotting region, solving the overlay problem with the scatter plot becomes even more imperative. Instead of jittering, let's do the opposite and round the data so that there is only full symbol overlapping in the output. Unfortunately, the results displayed in Figure 3 are far from satisfactory.

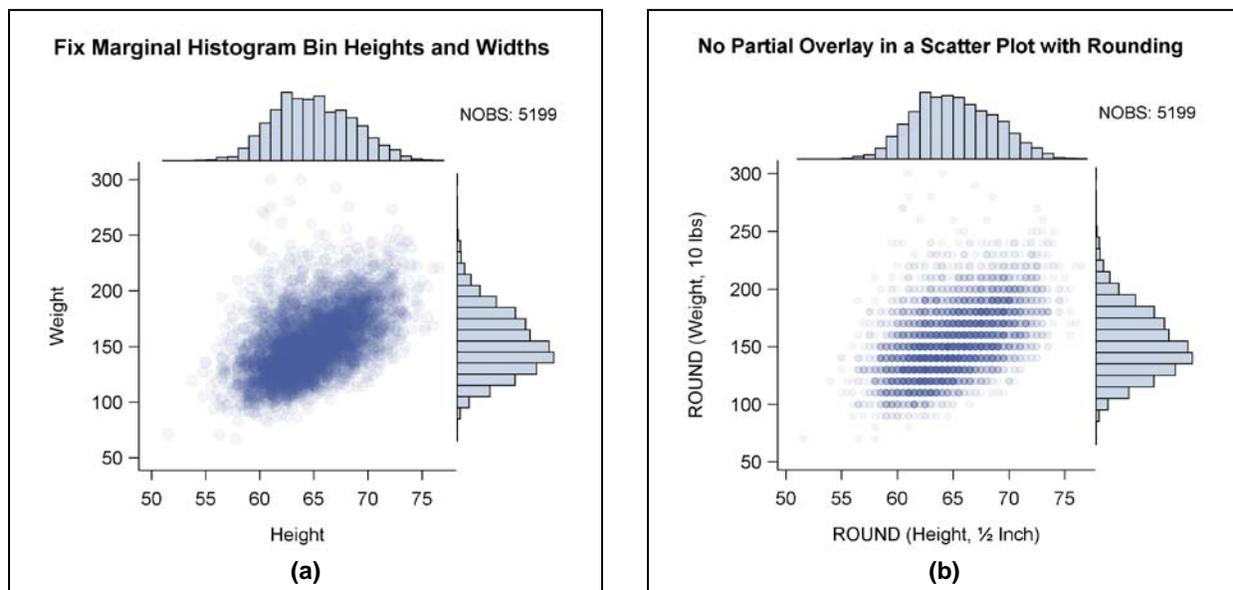


Figure 4. The graph on the left (a) is copied from Figure 3. In addition to rounding, the circle symbol size is reduced in the graph on the right (b). With such small symbols, rounding does not convey additional information about the frequency distribution in the scatter plot. To put it bluntly, we have just created a digitized blob. With rounding, the marginal histogram for `height` in the digitized plot is also shifted slightly to the right.

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

Despite the disappointing results in Figure 4, let's continue to explore rounding as a new technique for managing densely packed data displays. As hinted above, jittering and rounding are inverses of each other. The inverse relationship between these functions is depicted in Figure 5:

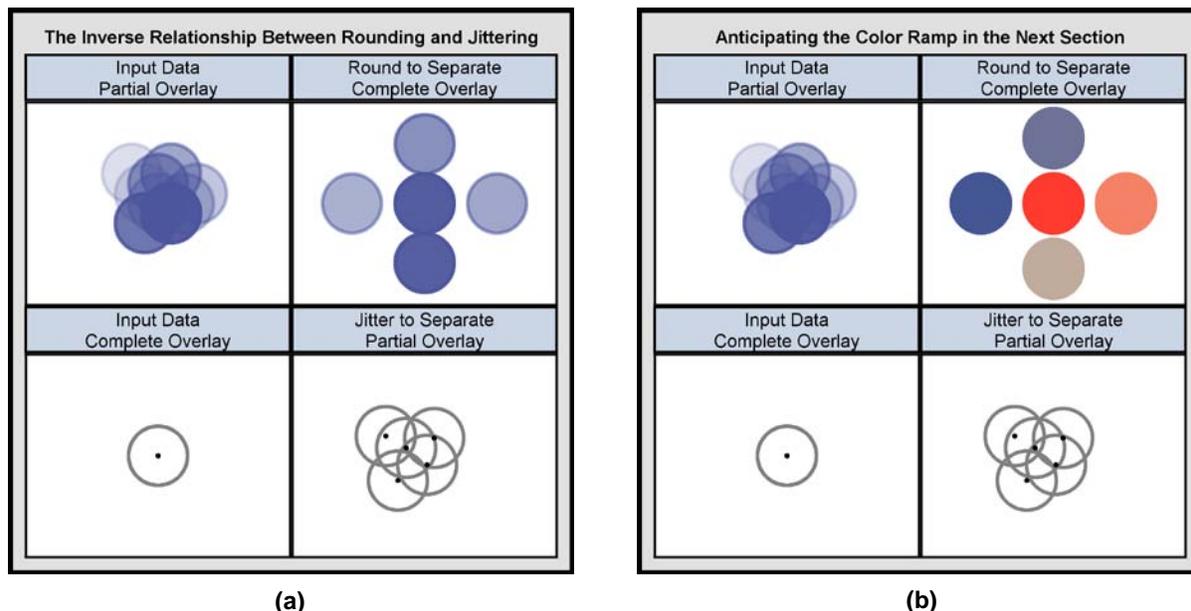


Figure 5. Both (a) and (b) demonstrate the inverse relationship between rounding and jittering where ROUND(partial overlay) → complete overlay and JITTER(complete overlay) → partial overlay. In the rounded panel for (a), it is hard to rank frequencies when the blues are almost identical. In (b), frequencies go from cold blue for low values to hot red for high ones.

Additional Comments about the Graph in Figure 5(a)

While jittering depends on hollow circles to promote visibility, partial overlay with transparency breaks down when dense data sets are plotted. It doesn't take too many partial overlays to generate dark blue regions in a graph. Nevertheless the problem is still not solved when data points are separated by rounding. Transparency loses its effect so that all you see are separate circles with slightly different shades of blue. In 5(a), for example, it is not possible to rank the five points by their associated frequencies.

Long before the advent of desktop computers, Josef Albers described a study in *Interaction of Color* where 60% of the advanced painting students in a basic color class were unable to distinguish lighter colors from darker ones [Albers \(1975, p. 12\)](#). As Albers points out, humans are much more adept at distinguishing between higher and lower musical tones. Therefore, when it comes to working with lightness scales in color, we are just about "tone deaf".

Despite our lack of skill when it comes to color, Albers insists we can develop a more discriminating sensitivity by looking at color scales [Albers \(1975, p.16\)](#). Color scales can be found in [Watts \(2002, 2003, 2004\)](#) and [Zdeb \(2002, pp. 128-130\)](#).

APPLY A SAS COLOR "RAMP" TO THE ROUNDED SCATTER PLOT

Since we are almost "tone deaf" when it comes to distinguishing different shades of the same color, we need an alternative color scale that will in Cleveland's view provide us with an:

effortless perception of the *order* of the encoded quantities, and clearly perceived *boundaries between adjacent levels* [Cleveland \(1994, forward\)](#). (Our emphasis added).

[Brewer \(1994\)](#) achieves these goals when she uses diverging sequences of lightness steps from two hues to compare employment rates among labor forces in European countries. SAS translates Cleveland's and Brewer's insights into a three-color RAMP that is applied to the scatter plot in Figure 6.

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

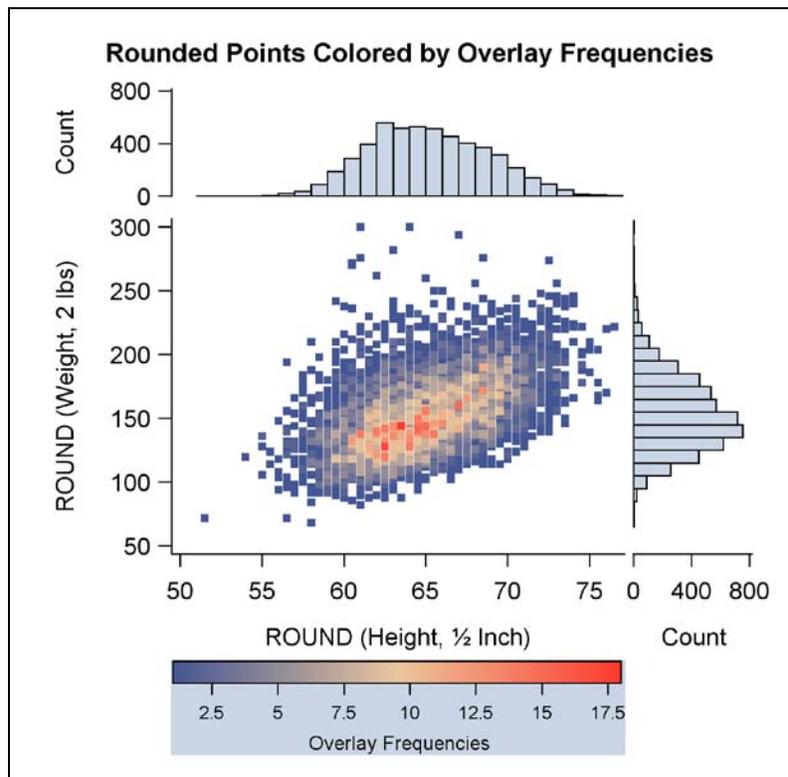


Figure 6. The three color ramp reflected in the legend is defined in the `COLORMODEL` option of the `SCATTERPLOT` statement.

Updates Made to the Graph in Figure 6

First noted is that the marker symbol in the graph is changed from `CIRCLEFILLED` to `SQUAREFILLED` so that histogram bin heights can be easily aligned with scatter plot frequencies. Now it is possible to see that the tallest bins line up with the brightest points in the data. Next, with the removal of transparency from the graph, all data points become fully visible. Full visibility makes it easy to line up histogram end bins with data outliers. Finally, the legend text is set off with a light blue background to distinguish it from the X-axis label and values. Adding a horizontal legend results in a graph that is stretched out in the horizontal direction similar to the one displayed in Figure 1. Histogram response axes are fully displayed to partially compensate for the distortion caused by the stretching. The more descriptive `COUNT` is selected as the axis scale in the display.

Relevant Code Fragments for Figure 6

Below, the derived data set `sTiedObs` (sorted-tied-observations) is created then used to generate the output displayed in Figure 6. The new variable, `count` is discussed when the code for `PROC TEMPLATE` is reviewed. Remember, more rounding leads to a greater incidence of tied observations. Less rounding means less distortion of the original data. Tradeoffs have to be made. In this instance `roundWeight`, originally an integer, can be off by 0 or 1 pounds; `roundHeight` recorded to the nearest $\frac{1}{4}$ inch in the raw data can be off by at most $\frac{1}{4}$ inch.

```
DATA roundHeart;
  SET sashelp.heart;
  roundWeight=ROUND(weight,2);
  roundHeight=ROUND(height,.5);
RUN;
PROC SUMMARY DATA=roundHeart NWAY;
  CLASS roundWeight roundHeight;
  OUTPUT OUT=TiedObs (KEEP=roundWeight roundHeight _FREQ_ RENAME = (_FREQ_ = count));
RUN;
PROC SORT DATA=tiedObs OUT=sTiedObs;
  BY count;
RUN;
```

An ascending sort is applied to the data so that coordinates with the highest frequencies will be fully visible, since they are plotted last. The code fragment for `PROC TEMPLATE` below shows how the newly created variable `count` in data set `sTiedObs` is used to create the desired graphics output.

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

```

PROC TEMPLATE;
  DEFINE STATGRAPH scatterhist;
  ...
  /* histogram at X2 axis position */
  LAYOUT OVERLAY / ...;
    HISTOGRAM roundHeight / FREQ=count BINAXIS=false;
  ENDLAYOUT;
  /* scatter plot: */
  LAYOUT OVERLAY / ...;
    SCATTERPLOT
      Y=roundWeight X=roundHeight / MARKERCOLORGRADIENT = freq;
  ENDLAYOUT;
  /* histogram at Y2 axis position */
  LAYOUT OVERLAY / ...;
    HISTOGRAM RoundWeight / FREQ=count ORIENT=horizontal BINAXIS=false;
  ENDLAYOUT;
  ...
END;
RUN;

```

MARKERCOLORGRADIENT is the option in the **SCATTERPLOT** statement that specifies the column (variable) used for mapping marker colors to a continuous gradient. Even though the **HISTOGRAM** statement works on raw data, it still supports a **FREQ** option with **FREQ=count**. Complete source code for the Figure 6 graph can be downloaded from <http://www.PerryWatts.org>.

USE PROC KDE TO ADD A THIRD DIMENSION TO DENSELY PACKED DATA

A slightly modified version of the plot in Figure 7(a) can be found in the GTL Reference Manual where the **CONTINUOUSLEGEND** statement is discussed (SAS Institute, 2009b, pp. 609-610). While **sashelp.gridded** is sent to **PROC SGRENDER** in the example from the manual, we reproduce the gridded data set from scratch in **PROC KDE** so that digitized contour plots can be generated from any data set that contains two continuous numeric variables.

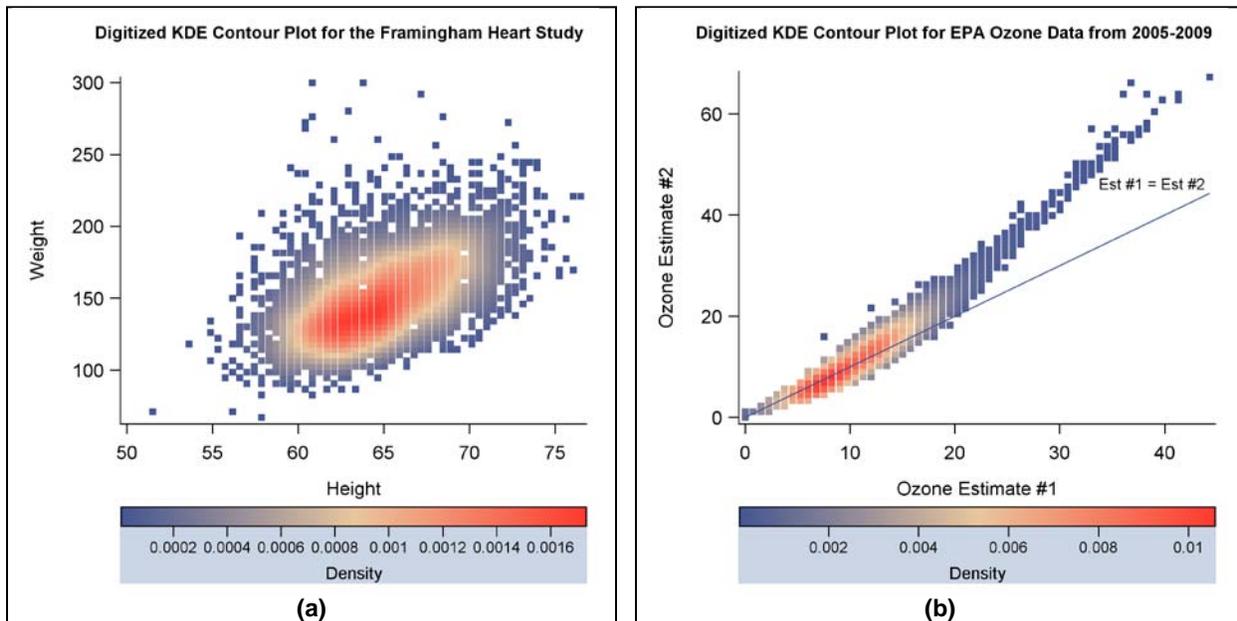


Figure 7. **PROC KDE** is used to generate digitized contour plots from two different input data sets. **X** and **Y** variables are identified in the input data set, and **PROC KDE** generates **density** that becomes the **Z** variable. The line plotted at a 45-degree angle with the **LINEPARM** statement in the EPA graph (b) shows where the plot would be positioned if the two ozone estimates were equal. The origin for **LINEPARM** is (0,0) and the slope is 1.

While the plots in Figures 6 and 7 have a similar appearance, their construction is very different. In Figure 6, **x** and **y** variables are rounded, and **z** is calculated as the frequency of tied observations at a given point. Probability estimates play no role here. In Figure 7, on the other hand, the plotting region is divided into a 60X60 grid of cells. This means that there are always 3,600 observations in any output data set from **PROC KDE**. Cell assignment is based on the kernel density estimate where “a known density function (the kernel) is averaged across observed data points to create a smooth approximation (SAS Institute, 2008, p.3460). In the digitized contour plot, color is assigned by the value for **density** at the intersection of horizontal and vertical grid points.

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

Relevant Code Fragments for Figure 7

A density variable `KDEdensity` is assigned to `MARKERCOLORGRADIENT` in the `STATGRAPH` template for Figure 5. `height` and `weight` become `gridHeight` and `gridWeight` to reflect the data transformations that occur in `PROC KDE`. The variable, `count`, also supplied by `PROC KDE`, contains the number of observations at a given grid point. `count` predictably sums to 5,199. Only those observations where `count` is greater than zero are passed through to `PROC TEMPLATE` from `PROC SGRENDER`. Otherwise white spaces in the graphics output area where zero counts have been excluded would be changed to dark blue to indicate low density.

```
PROC TEMPLATE;
  DEFINE STATGRAPH digitizedKDE;
    BEGINGRAPH /...;
    LAYOUT OVERLAY / ...;
      SCATTERPLOT
        Y=gridWeight X=gridHeight / MARKERCOLORGRADIENT=KDEdensity ...;
    ENDLAYOUT;
  ENDGRAPH;
END;
RUN;
PROC KDE DATA=sashelp.heart;
  BIVAR height weight / PLOTS=NONE OUT=KDEGridded (KEEP=value1 value2 density count
    RENAME = (VALUE1=gridHeight VALUE2=gridWeight DENSITY=KDEdensity));
RUN;
PROC SGRENDER DATA=KDEGridded (WHERE=(count>0)) TEMPLATE=digitizedKDE; ...;
RUN;
```

ADD THE BODY MASS INDEX TO THE DIGITIZED CONTOUR PLOT AND GROUP BY GENDER

An article published by the American Heart Association describes what happens over a 26 year period to subjects who participated in the Framingham Heart Study. Those studied had to be free of cardiovascular disease at their first medical examination. The paper concludes that “obesity, measured by Metropolitan Relative Weight, was a significant independent predictor of CVD, particularly among women” ([Hubert et al., 1983, p.968](#)).

Unfortunately we are unable to underscore the study’s findings graphically, because we lack complete information, and some of what we have is inconsistent. First off, there is no way to identify the 5,070 out of 5,209 men and women who were disease free at the beginning of the study. Also we cannot duplicate the SAS column MRW calculated as $(Actual\ Weight / Desirable\ Weight) \times 100$ by using an *adjusted* value for *Desirable Weight* supplied from Table 1 in the publication. We assume a different table with an *unadjusted* desirable weight was used to calculate MRW in the `sashelp.heart` data set. What we can do, however, is to categorize study subjects by their BMI scores to confirm that as a group they were considerably overweight especially by standards for the 1950’s. The categorization is done graphically by plotting BMI curves over the familiar digitized contour plot. The output is displayed in Figure 8.

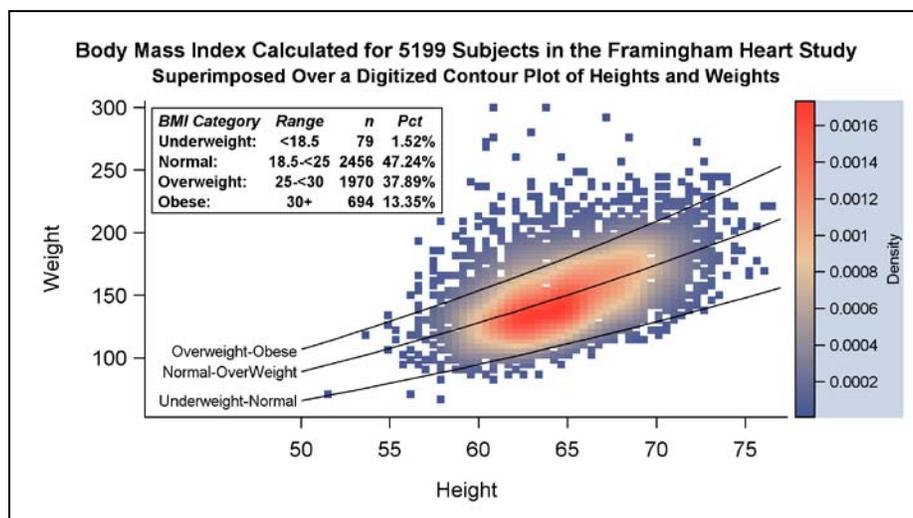


Figure 8. All the components of the graph are nested within a single `LAYOUT OVERLAY` statement. The boxed inset is constructed from a `GRIDDED` layout. `SERIESPLOT` statements are used to get BMI boundary lines, and the digitized contour plot is generated with a `SCATTERPLOT` statement. Stretching the graph along the horizontal axis increases readability and does not compromise the results, since there are no marginal histograms in the display.

Due to paper-length restrictions, source code is not being reviewed in this section. The complete SAS program can be downloaded from <http://www.PerryWatts.org>. When looking at the source code it is important to keep in mind that only a single data set can be processed in `PROC TEMPLATE`. Three would be preferable: one for the scatter plot, a second for the BMI boundary lines, and a third that captures the statistics reported in the inset.

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

The only way around this restriction is to concatenate multiple data sets with a `SET` statement. With concatenation, different variables can be constructed independently for exclusive assignment to the various plotting statements. For example, `height_BMI` and `weight_BMI` are assigned to `x` and `y` in the `SERIESPLOT` statement whereas `x` and `y` point to `gridHeight` and `gridWeight` in the `SCATTERPLOT` statement. When SAS is plotting `height_BMI`, the value for `gridHeight` in the same observation is missing, since it is not defined prior to concatenation. In this instance, the missing value is our friend; nothing is plotted by mistake.

Concatenation rather than any form of merging or joining greatly simplifies the data processing often required for constructing complicated graphs. It can also be used to extend the reach of an `ANNOTATE` data set in conventional SAS/GRAPH software. To our knowledge, this technique has not been described elsewhere in the SAS literature.

AIRLINES DATA: WORKING WITH A PROGRESSION OF TIME SERIES PLOTS

In this section, we work with a time series plot that incorporates the requirement for pre-sorted data into its definition. As defined in [Brockwell and Davis \(1996, p. 1\)](#), a *time series* is simply a set of observations indexed by time. The airlines data here is also "indexed by time, where the order of the observations matters" ([Derby and Vo, 2010](#)). In SAS, however, time series plots are implemented with the `SERIESPLOT` statement that accommodates both sorted and unsorted data. Since the language is so flexible, care must be taken to sort the input data by time or, if you have access to 9.3 SAS, to explicitly set `CONNECTORDER=XAXIS` ([SAS Institute, 2011, p.477](#)). The default setting is `CONNECTORDER=XVALUES` for X values that can be in any order.

A time series *progression* orders the series plots by two time dimensions. In the case of the Airlines data, the input data are sorted first by departure date and then within each departure date by the number of days before departure that a passenger books a flight.

GRAPHING THE AIRLINES DATA IN A SINGLE CELL

In Figure 9(a), departure date, one of the two time dimensions in the progression, is missing from the display. In Figure 9(b), color has been reorganized as a fourth dimension to order the series plots by their departure date range.

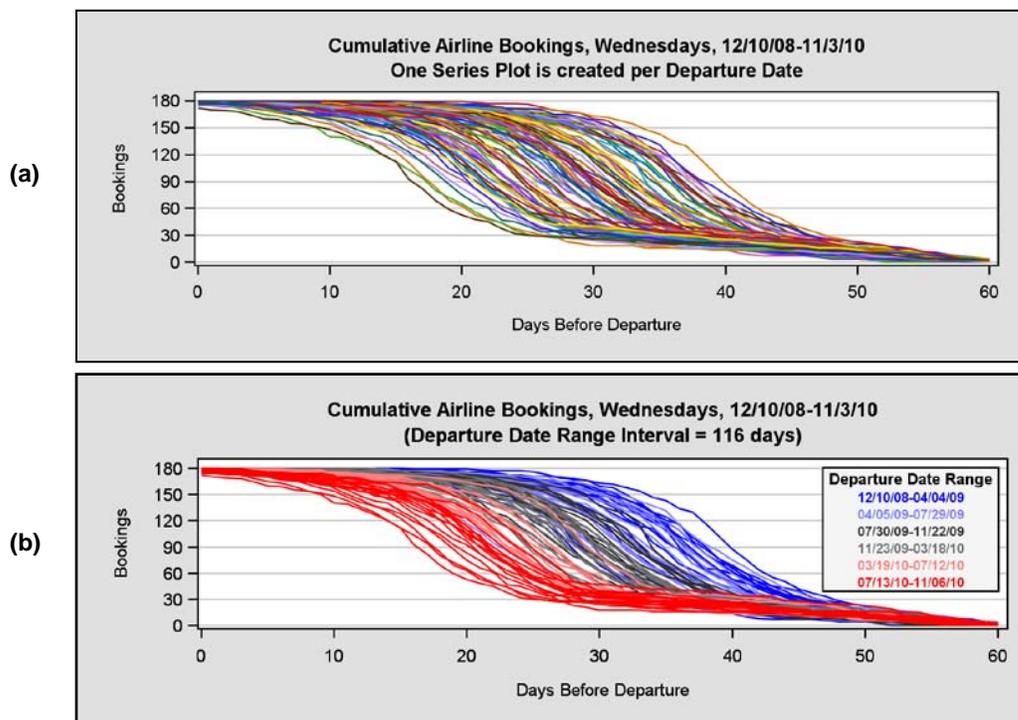


Figure 9. In (a), there is no way to connect a series plot to its departure date. In (b), color is based on departure date. The trend is obvious: in 2008-2009 passengers rushed out early to purchase their tickets whereas recent flights are booked at a more leisurely pace.

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

Relevant Code Fragments for Figure 9

A single `SERIESPLOT` statement is used in `PROC TEMPLATE` to create the 100 lines in Figure 9(a). In the code fragment below, the three dimensions from the input data are easily identified by variable name. They are `daysLeft`, `bookings` and `dDate` (for departure date). As a `GROUP` variable, each time the value for `dDate` changes a new plot line is generated. Color is also assigned by `dDate` on a cyclical basis from the `GRAPHDATA1 - GRAPHDATA12` style elements in the styles.default template for Figure 9(a). All the time series plots have solid lines with `PATTERN=1`.

```
LAYOUT OVERLAY / ...;
  SERIESPLOT X=daysLeft Y=bookings / GROUP=dDate LINEATTRS = (PATTERN=1);
ENDLAYOUT;
```

Code for the graph in Figure 9(b) is more intricate with color being used as a fourth dimension to link departure date to the number of days before departure that a flight is booked. To add the fourth dimension, variables `x1-x6` are created based on classification variable `byDateLb1`. For example, when `dDate` is between 12/10/08 and 04/04/09, `daysLeft` is assigned to `x1` and a point is plotted. When `dDate` is out of range, a missing value is assigned to `x1` and nothing is plotted. The same process involving `byDateLb1` determines the remaining assignments for `x2-x6`.

While `dDate` continues to define each of the 100 series plots, line color only changes when a new `SERIESPLOT` statement is issued. In addition, the inset created with `LAYOUT GRIDDED` is nested within `LAYOUT OVERLAY` along with the `SERIESPLOT` statements. Each `ENTRY` statement in the `GRIDDED` layout becomes a row in the one-column inset table (SAS Institute, 2009c, p.273). An inset is used in preference to a legend in this graph, because text with a bold weight is thicker and more visible than a legend line that is only 1 pixel wide.

```
LAYOUT OVERLAY / ...;
  %do i= 1 %to 6;
    SERIESPLOT Y=bookings X=x&i / GROUP=dDate LINEATTRS = (COLOR=&&color&i PATTERN=1);
  %end;
LAYOUT GRIDDED / COLUMNS=1 ...;
  ENTRY HALIGN=center TEXTATTRS = (WEIGHT=bold) "Departure Date Range";
  %do j = 1 %to 6;
    ENTRY HALIGN=center TEXTATTRS = (WEIGHT=bold COLOR=&&color&j) "&&Range&j";
  %end;
ENDLAYOUT;
```

GRAPHING THE AIRLINES DATA MULTI-CELL LAYOUT

Although the graph in Figure 9(b) confirms the existence of a relationship between booking lead times and flight departure dates, line overlay can be significantly reduced if the date range groups are displayed separately in `LAYOUT DATAPANEL`. Group comparisons can also be facilitated if the output is arranged in a 6(row) X 1(column) grid of graphs. The modified graph is displayed in Figure 10.

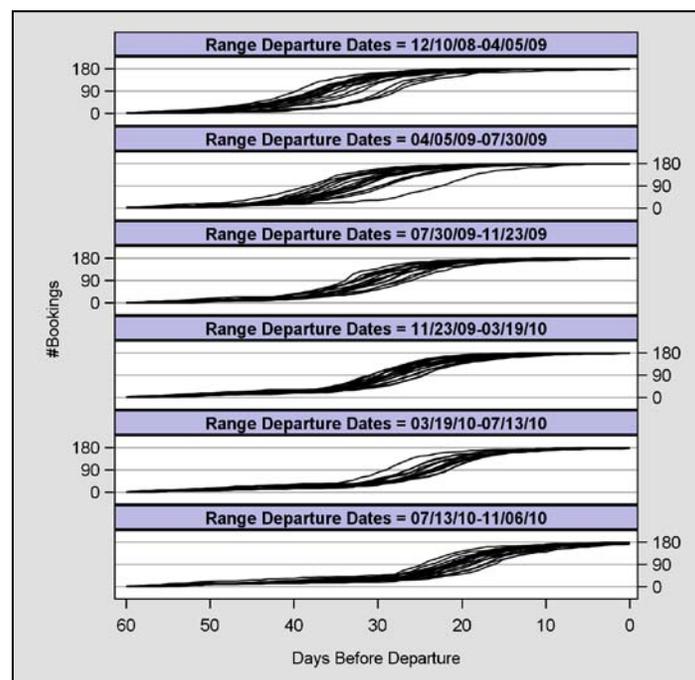


Figure 10. The six time series graphs are more visible with a `DATAPANEL` overlay. Color is no longer necessary, since each group is displayed separately. The X-axis is reversed so that cumulative distributions now go from left to right.

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

Relevant Code Fragments for Figure 10 Introduce the DATAPANEL Layout

The LAYOUT block in the SERIESPLT template below supports a DATAPANEL statement that:

creates a grid of graphs based on one or more classification variables and a graphical prototype. By default, a separate instance of the prototype (a data cell) is created for each actual combination of classification variables (SAS Institute, 2009b, p.59).

DATAPANEL is easier to work with than LATTICE. With the LATTICE layout illustrated in Figure 1, each cell in a grid of cells has to be defined separately whereas cell definitions are determined automatically by classification variable affiliation in DATAPANEL. In a DATAPANEL layout a single ROWAXISOPTS option replaces multiple YAXISOPTS whereas one COLUMNAXISOPTS replaces multiple XAXISOPTS. Likewise a single LAYOUT PROTOTYPE replaces multiple LAYOUT OVERLAY statements. With a simplified DATAPANEL structure, all panels in a graph have an identical appearance. Besides ease of use, the REVERSE option works for LAYOUT DATAPANEL, whereas it doesn't in 9.2 SAS for LAYOUT LATTICE. Below is an abbreviated code listing that shows how DATAPANEL produces the graph in Figure 10. Code in blue is discussed further below. Obtain complete source code for Figure 10 from <http://www.PerryWatts.org>.

```
PROC TEMPLATE;
  DEFINE STATGRAPH SERIESPLT;
  BEGINGRAPH / ...
    LAYOUT DATAPANEL CLASSVARS = (ByDDateLbl) /
      COLUMNDATARANGE = union
      HEADERLABELATTRS = (WEIGHT=bold SIZE=8px)
      HEADERBACKGROUNDCOLOR = CXBCB9E5
      ROWAXISOPTS = (GRIDDISPLAY=on DISPLAY=all ALTDISPLAY=none
        DISPLAYSECONDARY=none ALTDISPLAYSECONDARY = (ticks tickvalues))
      COLUMNAXISOPTS = (LABEL="Days Before Departure" REVERSE=true...);
    LAYOUT PROTOTYPE;
      SERIESPLOT Y=bookings X=DaysLeft / GROUP=DDate
        LINEATTRS = (PATTERN=1 COLOR=black);
    ENDLAYOUT; /*prototype*/
  ENDLAYOUT; /*dataPanel */
ENDGRAPH;
end;
run;
```

As in the discussion for Figure 9(b), the variable, `byDDateLbl`, contains six unique values that correspond to the six date ranges; one for each paneled display. If you look at the graph in Figure 10, you will see the vertical axis label "Cumulative #Bookings". This text string represents the label assigned to the `bookings` variable in the input data set. `DISPLAY=all` automatically picks up the data set label if there is one. `DISPLAY=all` also controls which axis features are displayed in *odd* row occurrences along the primary (left-side) axis (SAS Institute, 2009b, p.584). To determine a row's status as odd or even, count down from the top of the graph. Correspondingly, `ALTDISPLAY= none` controls what is displayed in the *even* rows of the graph. The same dichotomy is copied in the secondary Y axis by the `DISPLAYSECONDARY=` and the `ALTDISPLAYSECONDARY=` options. Manipulating the four display options is how to get tick marks to alternate along the primary and secondary axes in LAYOUT DATAPANEL.

BARLEY DATA: BRINGING ORDER TO MULTIWAY DOT PLOTS

Unlike all other plot types presented in the paper, dot plots have a named inventor; William S. Cleveland. As early as 1984, Cleveland described their structure and demonstrated why the dot plot is superior to the horizontal bar chart:

Dot charts can be used when the values in a data set have names that one wants to convey. When there are many values, the light dotted lines on the graph enable the visual connection of a plotted point with its name. ... The dotted lines on the dot chart have been made light to keep them from visually imposing and obscuring the large dots that portray the data (Cleveland, 1984, p. 271). (Our emphasis added).

By 1994, Cleveland changed the name *dot chart* to *dot plot* adding that "the dot plot has several different forms depending on the nature of the data and the structure of the labels" (Cleveland, 1994, p.17). One such form is the *multiway dot plot* where *panels* contain the individual dot plots and *levels* reference the rows in a single plot (Cleveland, 1993b, p. 303). With the multiway dot plot, 3-D is fixed from the outset: *panel* and *level* require two categorical variables whereas the third numeric response variable is reserved for the horizontal axis.

Probably Cleveland's most famous multiway dot plot is the one that displays barley yields collected from six different sites in Minnesota during the early 1930's. In *Visualizing Data*, Cleveland states that the barley study is significant, because R. A. Fisher used it early on to illustrate his ANOVA method of experimental design (Cleveland, 1993b, pp. 328-329). Unfortunately, however, the original data contained a serious error that Cleveland discovered years later when he graphed the results. It turns out that yearly barley yields were reversed at the Morris site.

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

The version of Cleveland's multiway dot plot reproduced in Figure 11 comes from [Rodriguez and Cartier \(2009, p. 3-10\)](#).¹ The source code is taken from the SAS publication, but the input data are from [Cleveland \(1993a\)](#). The example was presented at a SAS Global Forum Conference to show how easy it is to use the DOT statement in PROC SGPANEL. SAS automatically generates a 2X3 paneled display. Cleveland's original, however, was a 6X1 trellis plot.

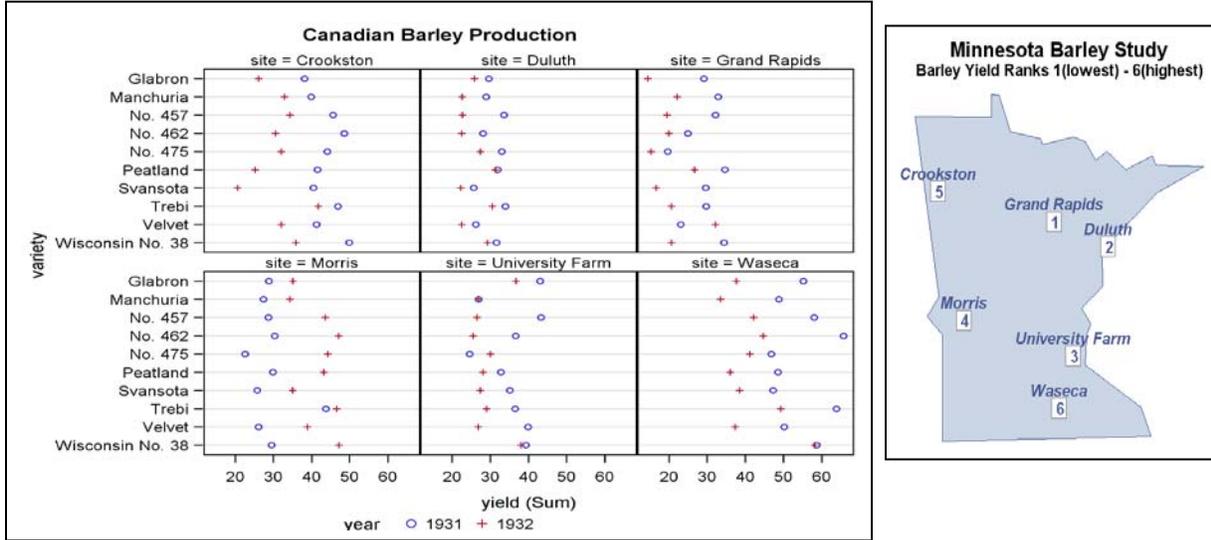


Figure 11. Unfortunately the graph is mislabeled as *Canadian Barley Production*. The graph also fails to reveal a second important pattern in the data. Results become obscure when sites are ordered alphabetically and when the output is displayed in a 2X3 rather than a 6X1 matrix of panels. A hint about the second pattern can be found by examining site ranks associated with barley yields in the map of Minnesota on the right. The map is generated using an algorithm adapted from [Zdeb \(2002, pp.70-74\)](#).

ORDERING BY MEDIAN

In *The Elements of Graphing Data*, Cleveland addresses the problem of nominal variables such as `site` that have no intrinsic order. He suggests ordering by median ([Cleveland, 1994, p. 153](#)). Therefore, in this example the medians are calculated separately for categorical variables `site` and `variety` on the response variable `yield`. Source code from PROC UNIVARIATE is listed below:

```

PROC UNIVARIATE NOPRINT DATA=barley;
  VAR yield;
  CLASS site;
  OUTPUT OUT=mdnbySite
    MEDIAN=mdnSiteYield;
RUN;

PROC UNIVARIATE NOPRINT DATA=barley;
  VAR yield;
  CLASS variety;
  OUTPUT OUT=mdnbyVariety
    MEDIAN=mdnVarietyYield;
RUN;
    
```

Ordering by median only uncovers patterns if they are present in the data. In Figure 12 below two dot plots are presented. The first is an enlargement of the Crookston site with `variety` listed on the Y axis in overall median order. No pattern can be detected. In the second dot plot, the median number of days before departure is calculated at the 90th booking for each departure date category. Results confirm the pattern seen in Figure 10.

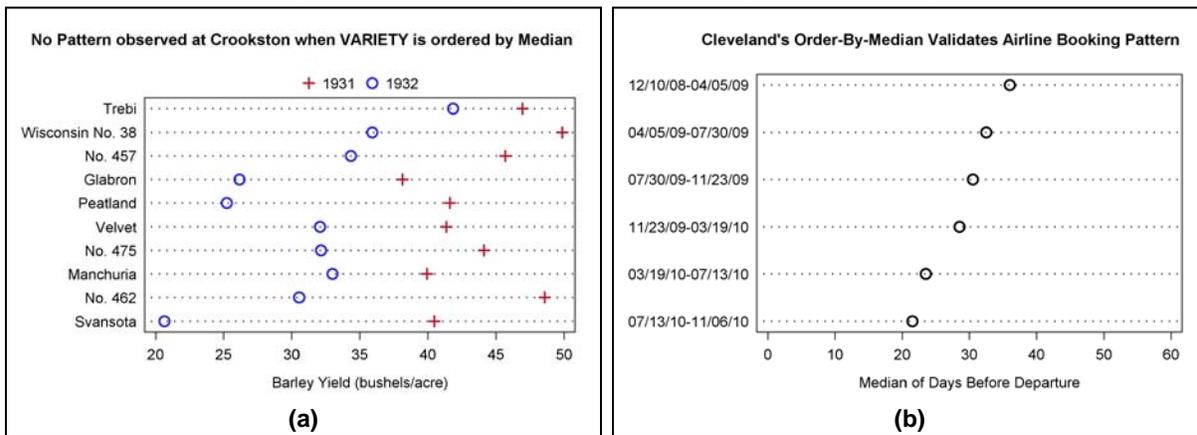


Figure 12. Barley varieties are ordered by median for Crookston, MN in (a). No pattern can be detected by looking at the dot plot. On the other hand, the medians calculated for the airlines data (b) repeat the pattern observed earlier: In 2010, people wait longer to book their flights meaning that they have the lowest median number of days before departure.

¹ Reproduced with permission of SAS Institute Inc., SAS Institute Inc., Cary NC, USA. All Rights Reserved.

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

CREATING DOT PLOTS IN GTL

What is needed for showing how `site` affects `yield` is a multiway dot plot ordered by median. To reorder the dot plot from Figure 11, we need to switch from `PROC SGPANEL` to `GTL`. However, when we go to make the switch we learn that there is no `DOTPLOT` statement in `GTL`. A work-around can be found by adding a `TMPLOUT` option to `PROC SGPANEL`:

```
PROC SGPANEL DATA=gtlDat.barley TMPLOUT='C:\Barley\sgpanelDotPlot.tpl';
  TITLE1 "Canadian Barley Production";
  PANELBY site;
  DOT variety / RESPONSE=yield GROUP=year;
RUN;
```

`TMPLOUT` stores a copy of the `GTL` template that is generated behind the scenes when `PROC SGPANEL` is executed ([SAS Institute, 2009d, p.129](#)). To get a listing of the template, click on **File** → **Open** in the SAS program window. Then from the folder of interest, scroll down to **Files of type** and with **All Files (*.*)** select the `TPL` file. Abbreviated output is shown below:

```
PROC TEMPLATE;
  DEFINE STATGRAPH sgpanel; ...;
  BEGINGRAPH /;
  ENTRYTITLE "Canadian Barley Production" /;
  LAYOUT GRIDDED / ROWGUTTER=5;
  LAYOUT DATAPANEL CLASSVARS = (site) / ...;
  LAYOUT PROTOTYPE / __SGPROC;
  SCATTERPLOT Y=variety X=_Sum1_yield_ / GROUP=year ...;
  ENDLAYOUT;
  ENDLAYOUT;
  ENDLAYOUT;
  ENDGRAPH;
END;
RUN;
```

From `sgpanelDotPlot.tpl` we learn that dot plots are created with the `SCATTERPLOT` statement in `GTL`. This means that the definition for *scatter plot* is extended in SAS to include nominal character variables where axes values are assigned to tick marks in alphabetical order. As you can see by looking at Figure 13(a), character data has the capability of wrecking havoc on the output from a `SCATTERPLOT` statement. Below is the code that generates Figure 13(a).

```
PROC TEMPLATE;
  DEFINE STATGRAPH barleyPanel6x1;
  BEGINGRAPH / ...;
  LAYOUT DATAPANEL CLASSVARS=( site ) / COLUMNS=1 ROWS=6
  ROWAXISOPTS=(DISPLAY=(line tickvalues) GRIDDISPLAY=on
  LINEAROPTS=( TICKVALUELIST=( 1 2 3 4 5 6 7 8 9 10 )
  TICKVALUEFORMAT=VarMdnf. ) )
  COLUMNAXISOPTS=( label="Barley Yield ( bushels/acre )" );
  LAYOUT PROTOTYPE;
  SCATTERPLOT X=yield Y=varietyMdnNum/GROUP=year NAME="year";
  ENDLAYOUT;
  ENDLAYOUT;
  ENDGRAPH;
END;
RUN;
```

Two formats are needed for `variety` so that `ROWAXISOPTS` will work as intended. `varietyMdnNum` matches `TICKVALUELIST`, and `varMdnf` is applied to the numeric list so numbers are translated into barley varieties listed along the Y axis. Unfortunately, the list viewed in Figure 13(a) is not complete. What SAS has done is to remove intermediate tick values from the axis with default `TICKVALUEFITPOLICY=thin` ([SAS Institute, 2009c, p.71](#)). Unlike data symbol overlay, SAS does not permit text collision in version 9.2 when it comes to labeling axes. Text collision prevention can be overridden in 9.3 SAS by specifying `TICKVALUEFITPOLICY=none` in `LINEAROPTS`.

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

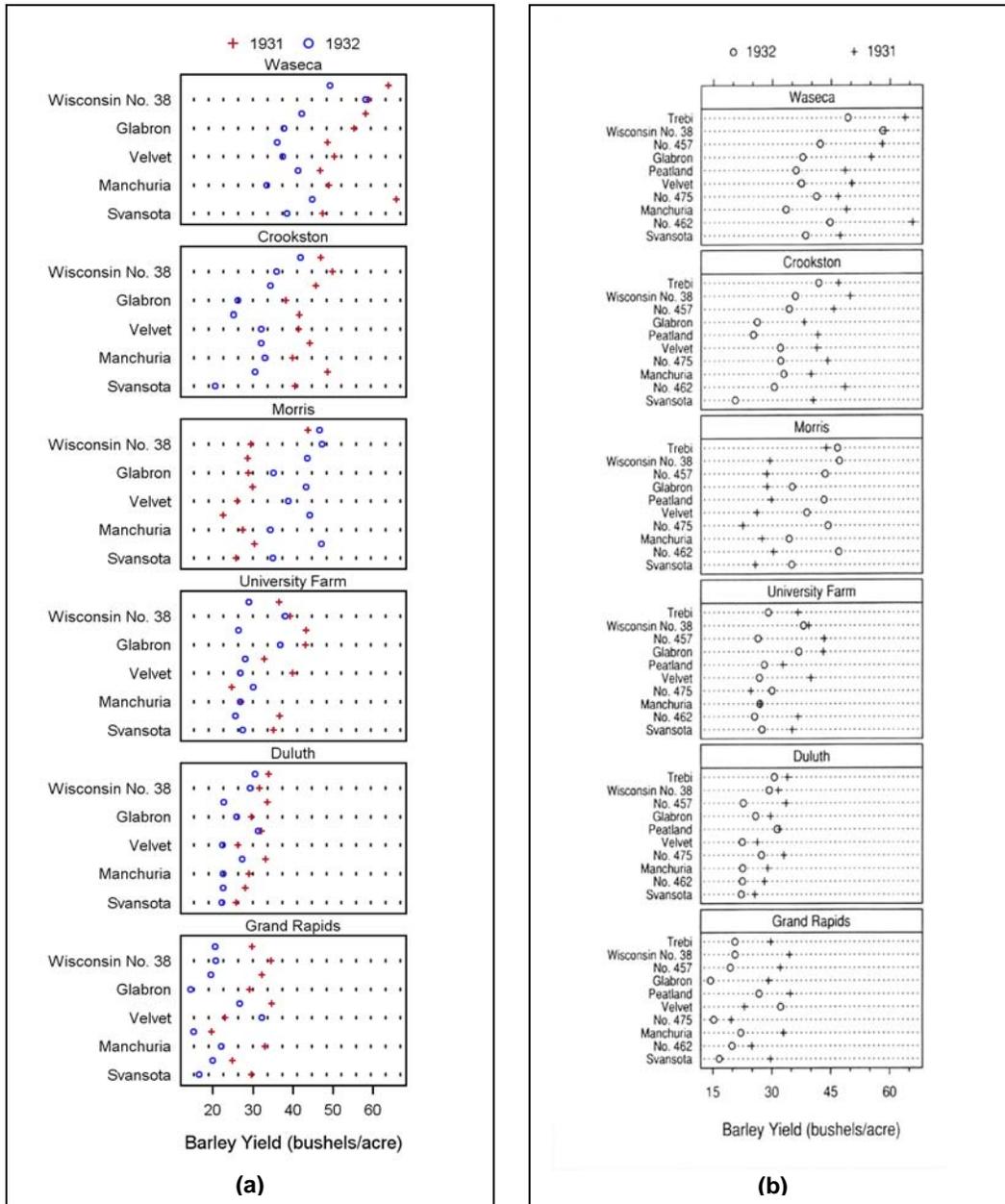


Figure 13. In both paneled graphs it is possible to see that barley yields gradually increase when sites are sorted by median. Note that every other variety with associated grid line is missing from the SAS GTL version (a), whereas Cleveland (b) is able to list all 10 VARIETY values in his 6X1 trellis plot (Cleveland, 1994, p.225).² Zoom in for a better view of both (a) and (b).

The graph displayed in Figure 13(a) confirms that axis thinning should be confined to numeric data. When numbers are deleted, the viewer typically has no trouble filling in the missing values. However, there is no way to know that *variety No. 457* is greater than *Glabron* and less than *Wisconsin No. 38* in Figure 13(a). Thus at 120 observations, it can be argued that there is too much summary data in the barley display to visualize it adequately in 9.2 SAS.

² Reproduced with permission from Hobart Press, Summit, NJ.

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

STOCK DATA: WORKING WITH INTERLEAVING TIME SERIES PLOTS

Warren Kuhfeld uses the `sashelp.stocks` data set to illustrate how time series plots are constructed in GTL and `PROC SGPLOT` [Kuhfeld \(2010, p.49\)](#). His GTL graph is reproduced in Figure 14(a) and updated in Figure 14(b). Unfortunately, the graph in Figure 14(b) doesn't represent a significant improvement over the graph in Figure 14(a). Unlike the 100 plot lines in the Airlines Data, the three lines in the Stock data are interleaved, and interleaving creates its own unique set of challenges that are addressed in this section.

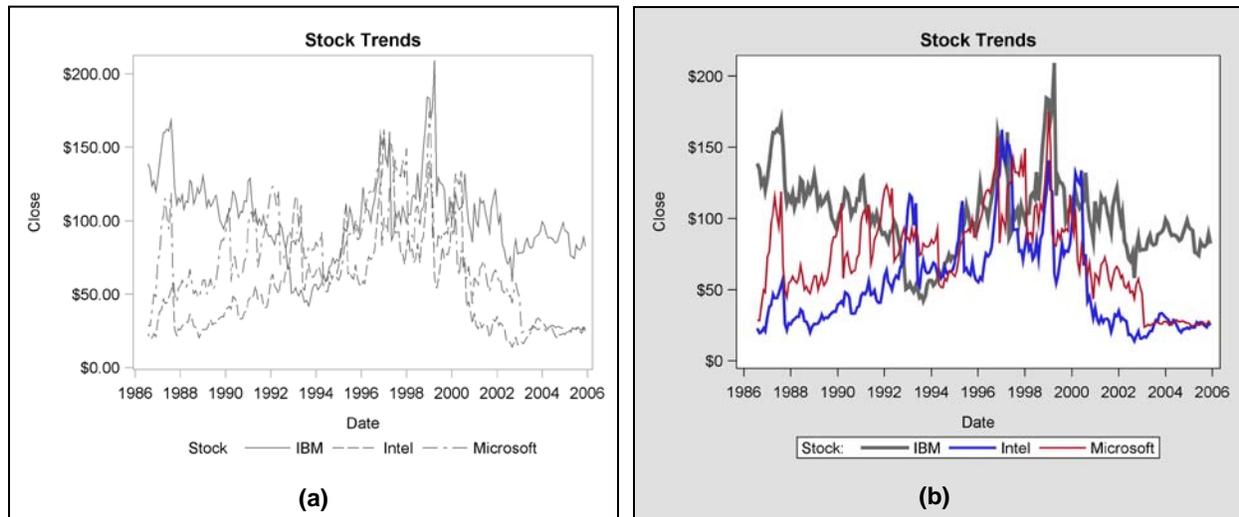


Figure 14. With interleaving lines, it is still difficult to see what happens between 1996 and 2000 in graph (b).

A number of changes are made to the graph in Figure 14(b). To start, the `STATISTICAL` style is replaced by the `DEFAULT` style so that the plotting region is set off from the rest of the graph. Next, the plotting region is extended slightly by reformatting the Y-axis values. Solid lines then replace dashed ones, because dashes interfere with line tracking. Finally, anti-aliasing is applied to improve resolution, and different line widths and colors make it easier but not completely possible to identify each series plot in the display.

It should be mentioned at this point that generating a single graph with multiple line widths and symbol sizes is not easy to do in SAS. These attributes are fixed in the `GRAPHDATADEFAULT` style element, so changes cannot be made to `GRAPHDATA1-GRAPHDATA12` where customization occurs. A macro that generates multiple line widths is available upon request.

[Robbins \(2005, pp.172-173\)](#) recommends placing each plot line into a separate panel of a multi-paneled display. Her trellis plot is similar in structure to the graph generated with a `DATAPANEL` layout in Figure 15(a). Now the time series plots are completely distinguishable, but their comparability diminishes with the increased distance between them. Robbins, in fact, hints that *visibility* and *comparability* have a conflicting relationship when she discusses the role between *distance* and *detection* in decoding a graph [Robbins \(2005, p. 63\)](#).

Results from an initial effort to increase comparability while maintaining visibility are displayed in Figure 15(b). Overall and individual average stock closings are added to the three plots with `LINEPARM` statements. Since each panel requires customization in the graph, `LAYOUT DATAPANEL` is replaced by `LAYOUT LATTICE`. Unfortunately, however, labels for the lines of averages shorten the widths of the plotting regions in each panel of Figure 15(b).

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

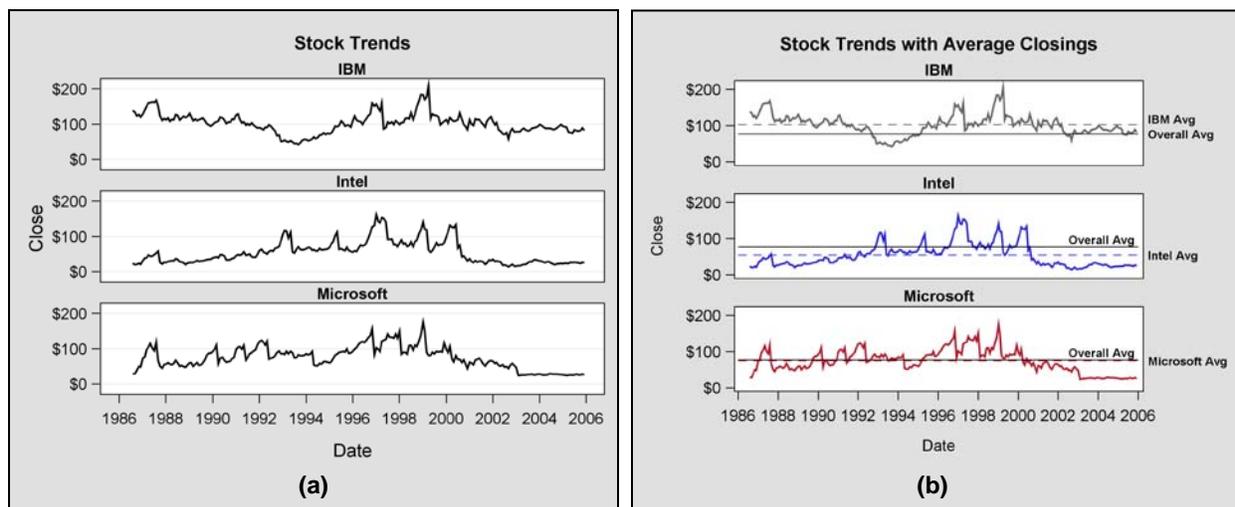


Figure 15. Comparability is compromised in (a), but partially restored in (b) with the addition of individual and overall average plot lines. Now individual variations around the plot lines of averages can be compared.

Another way to increase comparability while maintaining visibility is to graph pair-wise comparisons. Initial versions of the comparisons are displayed in Figure 16. In graph (a) two `SERIESPLOT` statements are issued for each panel, and in graph (b), transparent bands are added to emphasize differences in stock closings between two corporations that are being compared.

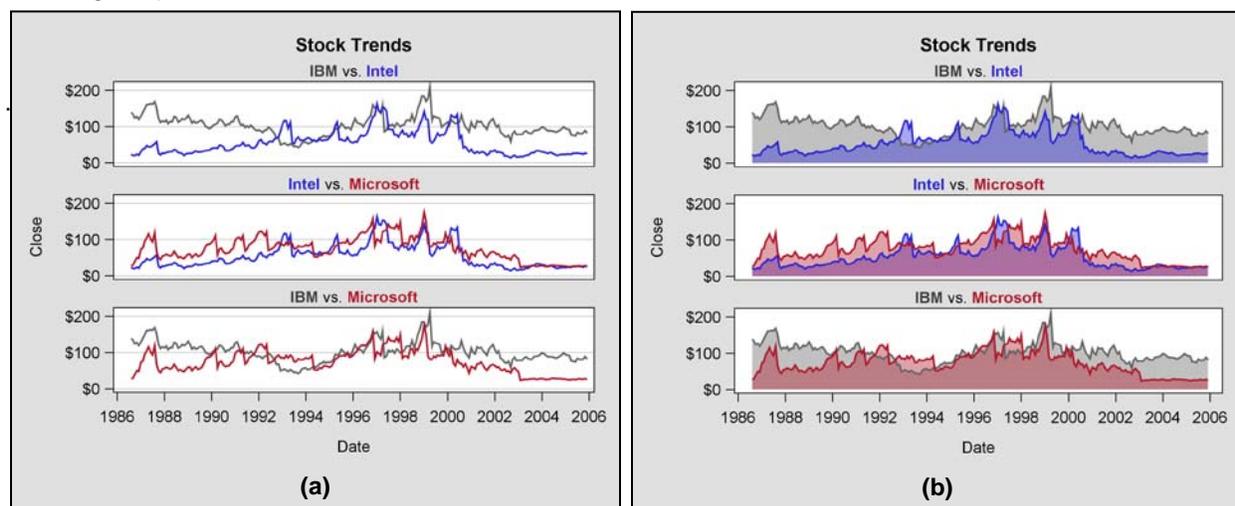


Figure 16. The area under the curve (AUC) becomes the focus of attention when transparent bands are added to the graph in (b). Observe that the larger area between the minimum of the two series plots and zero dollars is also shaded. This area is of no interest. However, with the double overlay it is the most emphasized region in the graph.

WORKING WITH BAND PLOTS IN A CELL BLOCK OF A LATTICE LAYOUT

Before getting too involved with band plots, let's review the relatively simple code for the `latticeSeriesPlt` template that is used to generate the graph in Figure 16(b). While `LAYOUT LATTICE` has been discussed before, the subordinate `CELL` block is new. When cell contents are specified in a `CELL` block, customized headers can then become an integral part of the specification ([SAS Institute, 2009b, p.92](#)). Since the headers of both graphs in Figure 16 serve as "quasi-legends" with identifying colors, their customization requires a `CELL` block within `LAYOUT LATTICE`.

```
PROC TEMPLATE;
  DEFINE STATGRAPH latticeSeriesPlt;
  BEGINNGRAPH;
  LAYOUT LATTICE / ...;
  /* Show code for first plot only */
  CELL;
  CELLHEADER;
  ENTRY TEXTATTRS = (COLOR=CX404040 WEIGHT=bold) "IBM"
    TEXTATTRS = (COLOR=black) " vs. "
    TEXTATTRS = (COLOR=graphdata2:CONTRASTCOLOR WEIGHT=bold) "Intel";
  ENDCELLHEADER;
  LAYOUT OVERLAY / ...;
```

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

```

BANDPLOT X=x1 LIMITUPPER=y1 LIMITLOWER=0 /
  FILLATTRS = (COLOR=graphdata&i:CONTRASTCOLOR) DATATRANSOPRENCY=0.8;
BANDPLOT X=x2 LIMITUPPER=y2 LIMITLOWER=0 /
  FILLATTRS = (COLOR=graphdata&j:CONTRASTCOLOR) DATATRANSOPRENCY=0.8;
SERIESPLOT Y=y1 X=x1 / LINEATTRS=graphData1;
SERIESPLOT Y=y2 X=x2 / LINEATTRS=graphData2;
ENDLAYOUT; /*overlay*/
ENDCELL;
ENDLAYOUT; /*lattice */
ENDGRAPH;
END;
RUN;

```

Band plots are not explicitly defined anywhere in the SAS literature. All that is said in the Language Reference manual is that the `BANDPLOT` statement is typically used to show “confidence or prediction limits” (SAS Institute, 2009b, p.157). In addition, when `x` is specified, as it is in the code fragment above, `LIMITUPPER` and `LIMITLOWER` expect references by variable or constant to a `y` value. What is left unsaid is that `LIMITUPPER` can never go below `LIMITLOWER`. When that occurs, the “band” is completely erased from the output. What is needed in this situation is *fake interleaving* where `LIMITUPPER` looks like it can go below `LIMITLOWER` and conversely `LIMITLOWER` looks like it can go above `LIMITUPPER`. Such flexibility is reflected in the final version of the time series graph for the `sashelp.stocks` data set in Figure 17.

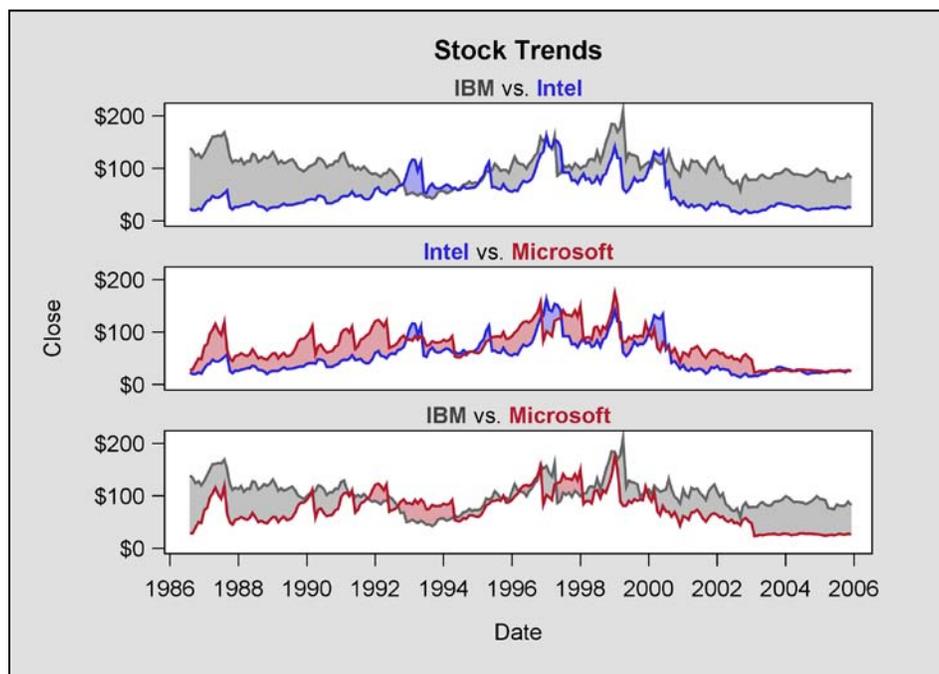


Figure 17. This time the area *between* the curves becomes the focus of attention. Now the two series plots in each panel are fully visible *and* comparable.

To see how it is possible to construct an interleaving band plot, compare Figures 17 and 18 for **Intel vs. Microsoft**. On January 4, 1988, the value for `close` is \$55.75 for Microsoft and \$25.50 for Intel. With these numbers, the first band plot in light red extends from \$55.75 down to \$25.50. A *second* virtually invisible band in Figure 17 extends from \$25.50 to \$25.49; a difference of one cent! One cent is still greater than zero, so `LIMITUPPER` continues to be greater than `LIMITLOWER`. How interleaving works is expressed visually in Figure 18 by increasing the lower boundary drop from one cent to \$10.00.

The `REFERENCELINE` statement is used for the first time in Figure 18 to mark January 4, 1988 on the graph. The code for the statement is as follows:

```

REFERENCELINE X='04Jan1988'D / LINEATTRS = (THICKNESS=4 COLOR=CX505050)
  CURVELABEL="January 4, 1988"
  DATATRANSOPRENCY=0.5;

```

`REFERENCELINE` differs from `LINEPARM` in that only an `x` or `y` coordinate has to be specified. Both coordinates are required for `LINEPARM` along with a value for `SLOPE`. Here is the corresponding code fragment for `LINEPARM` used in Figure 15(b) for the average lines:

```

LINEPARM X=EVAL(MIN(date)) Y=EVAL(MEAN(close)) SLOPE=0 / CURVELABEL="Overall Avg";

```

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

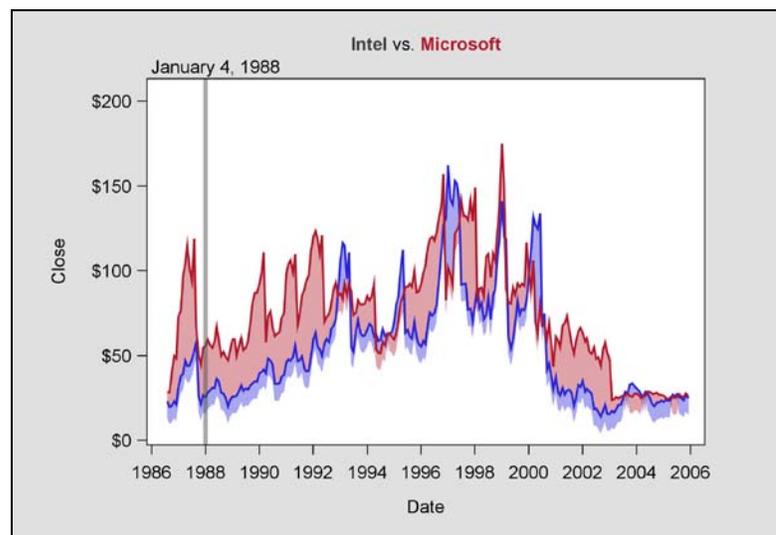


Figure 18. Two distinct continuous bands each with `LIMITUPPER` and `LIMITLOWER` values become visible when the lower boundary drop is increased from 1 cent to \$10.00 in the stock data.

SUMMARY, CONCLUSIONS AND FUTURE DIRECTIONS

An incremental approach has been taken in this paper to address issues associated with input data that translate into graphs with overlapping points, lines or axes labels. With the Framingham Heart Study we started with a scatter plot with marginal histograms that has a large undifferentiated “blob” of overlapping data points in the center. The graph was squared off so that the two histograms with uniform response axes specifications could be compared. Next *rounding*, the inverse of Cleveland’s *jittering*, was applied so that the “blob” could be color-coded by frequency. The digitized KDE contour plot was also color coded, but this time the coding was based on density, not frequency. In both updated plots, color became an important third dimension in the output.

Color was also systematically applied to ranges of departure dates in the airlines data so that a connection between departure date and booking lead times could be made. Unlike the heart data the airlines data could also be transformed into a more readable six-panel `DATAPANEL` plot.

Lattice plots were used to promote readability with the stock data. Unlike the airlines data where the progression of time series plot lines shared the same S-curve, the stock plot lines were uncoordinated, jagged and interleaved. Several display structures were presented as improvements over the original display. We favor the new *interleaving* band plot where the area between two curves becomes the focus of attention.

What surprised us the most is that we were not able to replicate Cleveland’s 6X1 multiway dot plot in version 9.2 SAS. The “before” SAS plot showed a 2X3 matrix with the data listed in alphabetical order. We wanted a 6X1 matrix where the data are sorted by median. Despite the small input data set size, a work-around proved to be elusive, because each dimension claiming its own separate geographical domain was explicitly defined in the data from the outset. Only with the latest release of SAS software is it now possible to replicate Cleveland’s graph created in 1994.

New graphics techniques have been introduced in this paper to stretch the limits of GTL. First there is “rounding” that adds dimensionality to a graph. Interleaving band plots also make their debut in this paper. They differ in structure from the grouped ban plots illustrated in *Statistical Graphics Procedures by Example: Effective Graphs Using SAS®* (Mantange and Heath, 2011, p.91).

Besides the new graphics techniques, two data processing techniques have been described that provide relatively simple workarounds for the single data set restriction in PROC TEMPLATE. The first is data set concatenation that accommodates different X and Y variables from sub-datasets that are then used in independent plotting statements. The second technique defines new variables in a single data set that are based on the values of a classification variable. The new variables are then used in multiple, related plotting statements. Both techniques map to plotting statements nested in a single `LAYOUT OVERLAY`, and they exploit missing values to get the desired graphics output.

In future papers look for additional techniques that will extend the limits of GTL. For the airlines data it will be possible to add a continuous legend to a graph that assigns a unique color to each of the 100 series plots by departure date. The `CONTINUOUSLEGEND` statement now only works with scatter and contour plots. Minor tick marks, not part of ODS statistical graphics, will also be programmed to bring the barley graph into closer alignment to what Cleveland produced, and finally a true *n-bin*, *end-point* histogram described in (Watts, 2009) will be translated into GTL.

A lot of ground has been covered in this paper, and if you are not familiar with GTL you may want to focus on the graphics output now and then return to the paper at a later date if you need to generate something similar in your own

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

work. In the meantime, read the SAS literature. The GTL manuals are particularly helpful, since they are motivated by example, and have complete listings of available options. To make the paper more accessible on a return visit, the following table lists GTL statements, layouts and additional features ordered by figure number.

Figure	Data	GTL Statements	GTL Layouts	Additional Features
1, 3	Heart	HISTOGRAM, SCATTERPLOT	LATTICE, OVERLAY	Marginal histograms EVAL= option
4, 5, 6	Heart	HISTOGRAM, SCATTERPLOT	LATTICE, OVERLAY	“Rounding”
7	Heart	SCATTERPLOT, LINEPARM	OVERLAY	PROC KDE data set
8	Heart	SERIESPLOT, SCATTERPLOT	GRIDDED, OVERLAY	Concatenate to add ROWS from different data sets.
9	Airlines	SERIESPLOT	GRIDDED, OVERLAY	Add COLUMNS from the same source to input data
10	Airlines	SERIESPLOT	DATAPANEL, PROTOTYPE	Axis Reversal
11	Barley	NA (SGPanel)	NA (SGPanel)	DOT statement in PROC SGPanel
12	Barley	SCATTERPLOT	OVERLAY	Ordering by median
13	Barley	SCATTERPLOT	DATAPANEL, PROTOTYPE	TMPLOUT=option
14	Stock	SERIESPLOT	OVERLAY	Statistical v. Default Style Varying line widths
15	Stock	SERIESPLOT, LINEPARM	DATAPANEL, LATTICE	Make comparable with LINEPARM
16	Stock	SERIESPLOT, BANDPLOT	LATTICE, OVERLAY	CELL block
17	Stock	SERIESPLOT, BANDPLOT	LATTICE, OVERLAY	Interleaving band plots
18	Stock	SERIESPLOT, BANDPLOT, REFERENCeline	LATTICE, GRIDDED	Explain interleaving band plots

REFERENCES

- Albers, J. (1975), *Interaction of Color*, revised edn, Yale University Press, Westford, MA.
- Brewer, C. A. (1994), Color use guidelines for mapping and visualization, in A. M. MacEachren and D. E. F. Taylor (eds), *Visualization in Modern Cartography*, Vol. 2, Elsevier Science, Inc., New York, pp. 123–147.
- Brockwell, P. J. and Davis, R. A. (1996), *Introduction to Time Series and Forecasting*, Springer Verlag, New York.
- Cleveland, W. S. (1984), Graphical Methods for Data Presentation: Full Scale Breaks, Dot Charts, and Multibased Logging, *The American Statistician*, **38**(4), 270–280.
- Cleveland, W. S. (1993a), Data from his Visualizing Data book.
<http://lib.stat.cmu.edu/datasets/visualizing.data.zip>
- Cleveland, W. S. (1993b), *Visualizing Data*, Hobart Press, Summit, NJ.
- Cleveland, W. S. (1994), *The Elements of Graphing Data, revised edn*, Hobart Press, Summit, NJ.
- Derby, N. and Vo, L. (2010), Graphing a progression of time series plots, *Proceedings of the 2010 Western Users of SAS Software Conference*.
http://www.wuss.org/proceedings10/analy/2954_6_ANL-Derby.pdf
- Hubert, H. B., Feinleib, M., McNamara, P. M. and Castelli, W. P. (1983), Obesity as an independent risk factor for cardiovascular disease: a 26-year follow-up of participants in the Framingham Heart Study, *Circulation*, **67**, 968–977.
- Kuhfeld, W. F. (2010), *Statistical Graphics in SAS®: An Introduction to the Graph Template Language and the Statistical Graphics Procedures*, SAS Institute, Inc., Cary, NC.
- Mantange, S., Cartier, J. and Heath, D. (2009), *SAS/GRAPH®: Graph Template Language* distributed as a SAS Institute handout at SAS Global Forum, 2009. Code for SCATTERHIST.SAS in the handout is at [SAS Institute \(2009a\)](#).
- Mantange, S. and Heath, D. (2011), *Statistical Graphics Procedures by Example: Effective Graphs Using SAS®*, SAS Institute, Inc., Cary, NC.
- McClave, J. T. and Sincich, T. (2003), *Statistics, ninth edn*, Prentice-Hall, Inc., Upper Saddle River, NJ.
- Robbins, N. B. (2005), *Creating More Effective Graphs*, John Wiley and Sons, Inc., Hoboken, NJ.
- Rodriguez, R. N. (2008), Getting started with ODS statistical graphics in SAS 9.2, *Proceedings of the 2008 SAS® Global Forum*, paper 305-2008.

Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize, continued

<http://www2.sas.com/proceedings/forum2008/305-2008.pdf>

Rodriguez, R. N. and Cartier, J. (2009), *Creating Statistical Graphics with ODS in SAS 9.2*, Seminar notes.

SAS Institute (2008), *SAS/STAT® 9.2 User's Guide*, SAS Institute, Inc., Cary, NC.

SAS Institute (2009a), *Sample 35172: Distribution plot*.

<http://support.sas.com/kb/35/172.html>

SAS Institute (2009b), *SAS/GRAPH® 9.2 Graph Template Language Reference*, SAS Institute, Inc., Cary, NC.

SAS Institute (2009c), *SAS/GRAPH® 9.2 Graph Template Language User's Guide*, SAS Institute, Inc., Cary, NC.

SAS Institute (2009d), *SAS/GRAPH® 9.2 Statistical Graphics Procedures Guide*, SAS Institute, Inc., Cary, NC.

SAS Institute (2011), *SAS/GRAPH® 9.3 Graph Template Language User's Guide*, SAS Institute, Inc., Cary, NC.

Siegel, A. F. and Morgan, C. J. (1996), *Statistics and Data Analysis: An Introduction, second edition*, John Wiley and Sons, Inc. New York.

Watts, P. (2002), Using ODS and the Macro Facility to Construct Color Charts and Scales for SAS® Software Applications, *Proceedings of the Twenty-Seventh SAS Users Group International Conference*, paper 125-27.

<http://www2.sas.com/proceedings/sugi27/p125-27.pdf>

Watts, P. (2003), Working with RGB and HLS Color Coding Systems in SAS® Software, *Proceedings of the Twenty-Eighth SAS® Users Group International Conference*, paper 136-28.

<http://www2.sas.com/proceedings/sugi28/136-28.pdf>

Watts, P. (2004), Advanced Programming Techniques for Working with Color in SAS® Software, *Proceedings of the Twenty-Ninth SAS Users Group International Conference*, paper 091-29.

<http://www2.sas.com/proceedings/sugi29/091-29.pdf>

Watts, P. (2009), Using SAS® Software to Generate Textbook Style Histograms. *Proceedings of the SAS® Global Forum 2009 Conference*, paper 216-2009. <http://support.sas.com/resources/papers/proceedings09/216-2009.pdf>

Yang, D. (2011), Creating a half error bar using Graph Template Language, *Proceedings of the 2011 SAS® Global Forum*, paper 284-2011.

<http://support.sas.com/resources/papers/proceedings11/284-2011.pdf>

Zdeb, M. (2002), *Maps Made Easy Using SAS®*, SAS Institute, Inc., Cary, NC.

WHAT'S IN GTL4TOOMUCHDATA.ZIP:

- | | |
|------------------|-----------------|
| 1) SAS Data Sets | 2) SAS Programs |
| •HeartBMI_MRW | •Figure6.sas |
| •groupedAirData | •Figure8.sas |
| | •Figure10.sas |

To request additional source code, send e-mail to Perry Watts with the Figure Number in the subject line.

ACKNOWLEDGEMENTS

The authors thank Jean-Jacques Dubois from the US Environmental Protection Agency for sharing the data set that is graphed in Figure 7(b). We also wish to express our gratitude to William S. Cleveland for his seminal contributions to the field of statistical graphics. His two books, *Visualizing Data* and *The Elements of Graphing Data* are gems of clarity. They inspired a lot of what appears in this paper.

TRADEMARK CITATION

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

CONTACT INFORMATION

Comments and questions are valued and encouraged. Contact one of the authors:

Perry Watts
Stakana Analytics
pwatts@stakana.com

Nate Derby
Stakana Analytics
nderby@stakana.com

