Paper SAS596-2017

The Art of Overlaying Graphs for Creating Advanced Visualizations

Vineet Raina, SAS Research and Development, India

ABSTRACT

SAS® provides an extensive set of graphs for different needs. But as a SAS programmer or someone who uses SAS® Visual Analytics Designer to create reports, the number of possible scenarios you have to address outnumber the available graphs. This paper demonstrates how to create your own advanced graphs by intelligently combining existing graphs. This presentation explains how you can create the following types of graphs by combining existing graphs: a line-based graph that shows a line for each group such that each line is partly solid and partly dashed to show the actual and predicted values respectively; a control chart (which is currently not available as a standard graph) that lets you show how values change within multiple upper and lower limits; a line-based graph that gives you more control over attributes (color, symbol, and so on) of specific markers to depict special conditions; a visualization that shows the user only a part of the data at a given instant, and lets him move a window to see other parts of the data; a chart that lets the user compare the data values to a specific value in a visual way to make the comparison more intuitive; and a visualization that shows the overall data and at the same time shows the detailed data for the selected category. This paper demonstrates how to use the technique of combining graphs to create such advanced charts in SAS® Visual Analytics and SAS® Graph Builder as well as by using SAS procedures like the SGRENDER procedure.

INTRODUCTION

SAS provides a large set of graphs that satisfy common needs, but often we end up in situations where we have to contrive our own visualizations to show the data in the best possible way. Such visualizations might not be directly available in SAS graphs but can be created by combining existing graphs. This technique can be used by SAS programmers as well as by users of products like SAS Visual Analytics. This paper does not aim to educate readers in all possible graphs and their options, but aims to demonstrate the power of the overlay process by showcasing examples in which advanced visualizations were created to address uncommon requirements. This paper also describes a few other ways of combining graphs in addition to the overlay technique. These examples encourage readers to explore this technique further to find newer possibilities of combining graphs to address their specific needs.

PREDICTING VALUES OF GROUPS

Often we are required to display the actual values of groups along with the predicted values. A common way to accomplish this is by showing a grouped series plot that shows a line for each group that is partly solid and partly dashed to depict the actual and predicted values respectively. A series plot by itself does not enable a user to display a line that is partly solid and partly dashed. There are several constraints in this situation:

- All this information should be displayed in a single coordinate system, so the x and y ranges of actual and predicted data points need to be combined.
- Each line should be a unique color, distinct from other lines. The color of a line should be consistent between actual and predicted regions.
- The legend should show what each line represents.
- The predicted values for all groups might start at different values of the x variable.

A simple overlay of two series plots (one for actual values and one for predicted values) gives us a visualization close to what we want. The following code results in this simple overlay:

```
data work.actualpredicted;
   year=1990; actual=20; predicted=.; product="TV";   output;
   year = 1991;actual=21;predicted=.; product="TV";   output;
```

```
    year = 1992;actual=23;predicted=.; product="TV";   output;
    year = 1993;actual=22;predicted=.; product="TV";   output;
    year = 1994;actual=24;predicted=.; product="TV";   output;
    year=  1995; actual=.; predicted=23; product="TV";   output;
    year = 1996;actual=.;predicted=25; product="TV";   output;
    year = 1997;actual=.;predicted=24; product="TV";   output;
    year = 1998;actual=.;predicted=22; product="TV";   output;
    year = 1999;actual=.;predicted=25; product="TV";   output;
    year=1990; actual=22; predicted=.; product="Laptop";   output;
    year = 1991;actual=25;predicted=.; product="Laptop";   output;
    year = 1992;actual=27;predicted=.; product="Laptop";   output;
    year = 1993;actual=23;predicted=.; product="Laptop";   output;
    year = 1994;actual=.;predicted=26; product="Laptop";   output;
    year=  1995; actual=.; predicted=28; product="Laptop";   output;
    year = 1996;actual=.;predicted=30; product="Laptop";   output;
    year = 1997;actual=.;predicted=27; product="Laptop";   output;
    year = 1998;actual=.;predicted=26; product="Laptop";   output;
    year = 1999;actual=.;predicted=31; product="Laptop";   output;
    year=1990; actual=19; predicted=.; product="Cellphone";   output;
    year = 1991;actual=21;predicted=.; product="Cellphone";   output;
    year = 1992;actual=24;predicted=.; product="Cellphone";   output;
    year = 1993;actual=22;predicted=.; product="Cellphone";   output;
    year = 1994;actual=21.;predicted=.; product="Cellphone";   output;
    year=  1995; actual=25; predicted=.; product="Cellphone";   output;
    year = 1996;actual=22;predicted=.; product="Cellphone";   output;
    year = 1997;actual=23;predicted=.; product="Cellphone";   output;
    year = 1998;actual=.;predicted=25; product="Cellphone";   output;
    year = 1999;actual=.;predicted=23; product="Cellphone";   output;
run;
ods path(prepend) work.templat(update);
proc template;
  define statgraph seriesplot;
    begingraph;
      entrytitle "Annual Sales";
      layout overlay /cycleattrs=true
        xaxisopts=(label="Year")
        yaxisopts=(label="Revenue (million dollars)");
        seriesplot x=year y=actual / group=product lineattrs=(thickness=4)
name='actualrevenue';
        seriesplot x=year y=predicted / group=product display=(markers)
markerattrs=(size=10px) lineattrs=(thickness=4 pattern=12)
name='predictedrevenue' ;
          discretelegend 'actualrevenue'/title="Product";
      endlayout;
    endgraph;
  end;
run;
proc sgrender data=work.actualpredicted template=seriesplot;
run;
```

Figure 1 shows the output for this program. Some of the constraints have been addressed. For example, there is a solid line for each group showing the actual values and a dashed line for each group showing the predicted values. Also, the x axis and y axis ranges have been combined to cover the actual and predicted ranges. However, there are several problems in this output:

- The dashed part of a line is not joined to the solid part of the corresponding group.
- The color of the dashed part does not match the solid part.
- Legend entries do not include the dashed parts.

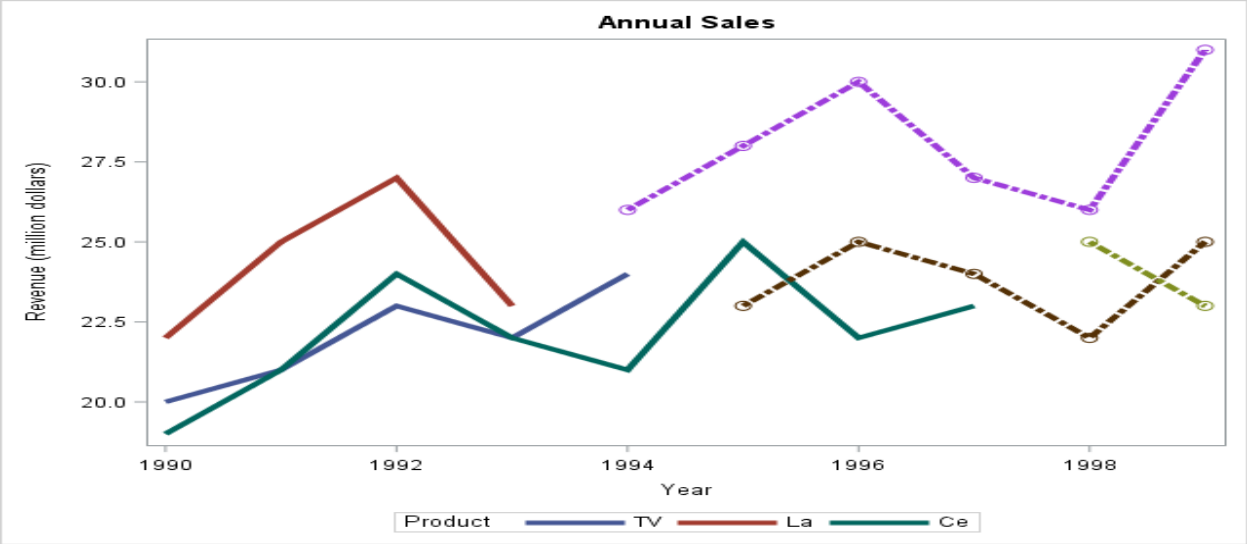

Figure 1. Output Using Basic Overlay

REFINING THE VISUALIZATION

The problems with the output shown in Figure 1 can be addressed by completing these steps:

- Data Preparation: To ensure that the dashed part of a group is joined to the solid part, we need to insert a dummy predicted value for the last point that had a valid actual value such that the predicted and actual values for this point are same. This step needs to be completed for each group.
- Changes to SAS procedures: The lines for the groups in the first series plot in this overlay use colors GRAPH_DATA_1 to GRAPH_DATA_N, where N is number of groups. The lines in the second series plot use GRAPH_DATA_N+1 to GRAPH_DATA_2N as cycleattrs is set to true. This can be solved by setting cycleattrs to false, which causes each series plot in the overlay to start the colors again from GRAPH_DATA_1. This setting ensures that the dashed lines use the same color as the solid lines.

Following is the modified program:

```
data work.modactualpredicted;
    year=1990; actual=20; predicted=.; product="TV";   output;
   year = 1991;actual=21;predicted=.; product="TV";   output;
   year = 1992;actual=23;predicted=.; product="TV";   output;
   year = 1993;actual=22;predicted=.; product="TV";   output;
   year = 1994;actual=24;predicted=24; product="TV";   output;
   year=  1995; actual=.; predicted=23; product="TV";   output;
   year = 1996;actual=.;predicted=25; product="TV";   output;
   year = 1997;actual=.;predicted=24; product="TV";   output;
   year = 1998;actual=.;predicted=22; product="TV";   output;
   year = 1999;actual=.;predicted=25; product="TV";   output;
   year=1990; actual=22; predicted=.; product="Laptop";   output;
   year = 1991;actual=25;predicted=.; product="Laptop";   output;
   year = 1992;actual=27;predicted=.; product="Laptop";   output;
   year = 1993;actual=23;predicted=23; product="Laptop";   output;
   year = 1994;actual=.;predicted=26; product="Laptop";   output;
   year=  1995; actual=.; predicted=28; product="Laptop";   output;
```

```
   year = 1996;actual=.;predicted=30; product="Laptop";   output;
   year = 1997;actual=.;predicted=27; product="Laptop";   output;
   year = 1998;actual=.;predicted=26; product="Laptop";   output;
   year = 1999;actual=.;predicted=31; product="Laptop";   output;
   year=1990; actual=19; predicted=.; product="Cellphone";   output;
   year = 1991;actual=21;predicted=.; product="Cellphone";   output;
   year = 1992;actual=24;predicted=.; product="Cellphone";   output;
   year = 1993;actual=22;predicted=.; product="Cellphone";   output;
   year = 1994;actual=21.;predicted=.; product="Cellphone";   output;
   year=  1995; actual=25; predicted=.; product="Cellphone";   output;
   year = 1996;actual=22;predicted=.; product="Cellphone";   output;
   year = 1997;actual=23;predicted=23; product="Cellphone";   output;
   year = 1998;actual=.;predicted=25; product="Cellphone";   output;
   year = 1999;actual=.;predicted=23; product="Cellphone";   output;
run;
ods path(prepend)
work.templat(update);
proc template;
  define statgraph modseriesplot;
    begingraph;
      entrytitle "Annual Sales";
      layout overlay / cycleattrs=false
        xaxisopts=(label="Year")
        yaxisopts=(label="Revenue (million dollars)");
        seriesplot x=year y=actual / group=product lineattrs=(thickness=4)
name='modactualrevenue';
       seriesplot x=year y=predicted / group=product display=(markers)
markerattrs=(size=10) lineattrs=(thickness=4 pattern=12)
name='modpredictedrevenue' ;
        discretelegend 'modactualrevenue'/;
      endlayout;
    endgraph;
  end;
run;
proc sgrender data=work.modactualpredicted template=modseriesplot;
run;
```

Figure 2 shows the output using the refined program. Note that the dashed parts and solid parts of the lines are joined. Also, the color of the dashed part also matches the color of the solid part of the corresponding group, giving the impression of a continuous line. Now the legend entries match the color of the corresponding solid and dashed line.

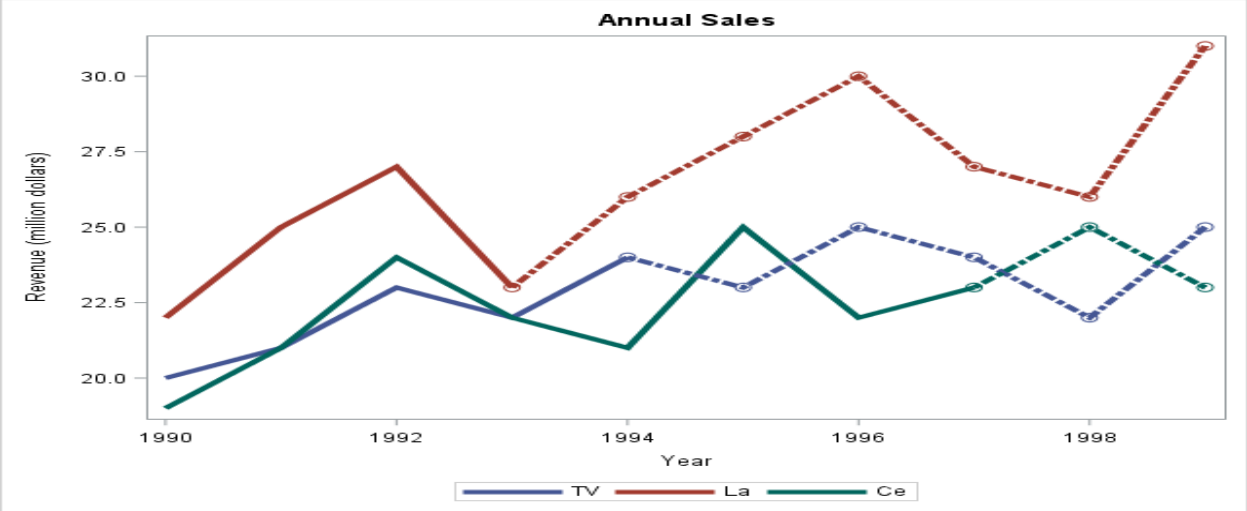

Figure 2. Output Using Refined Program

OVERLAYING A REFERENCE LINE TO CREATE NEW VISUALIZATIONS

We are often required to compare response values in a chart to a specific threshold in order to make certain conclusions about the data. Consider the following SAS program:

```
proc template;
  define statgraph scatterplot;
    begingraph;
      entrytitle "Height and Weight by Sex";
      layout overlay;
        scatterplot x=height y=weight / group=sex name='scatter'
datalabel=name;

        discretelegend 'scatter';
      endlayout;
   endgraph;
 end;
run;
proc sgrender data=sashelp.class template=scatterplot;
run;
```

Figure 3 shows the output for this program.

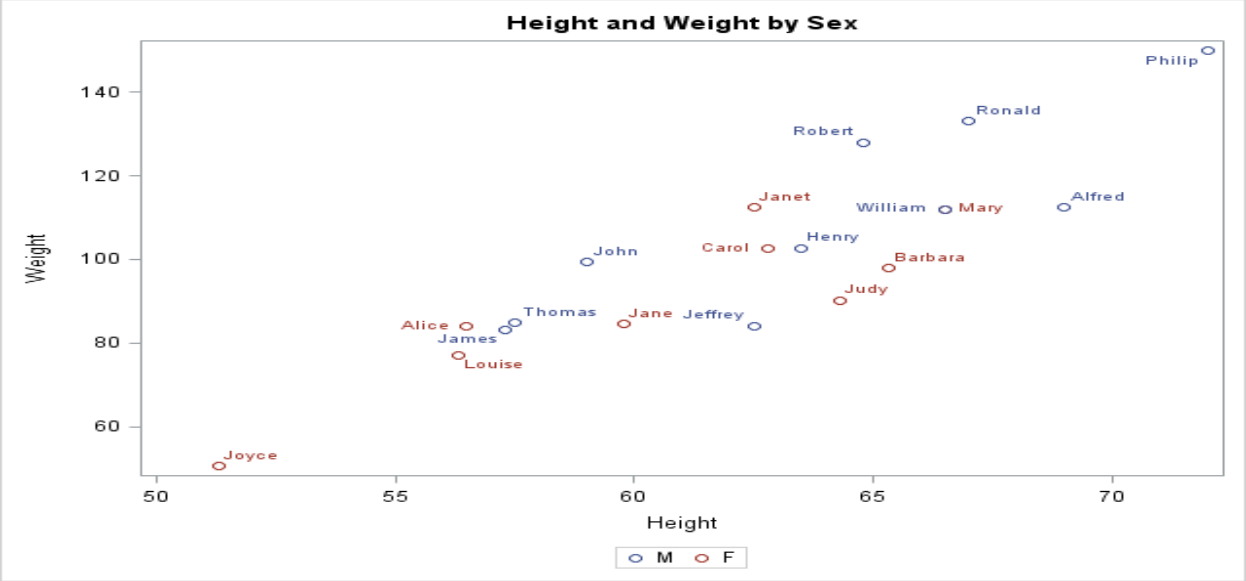

Figure 3. Scatter Plot without Reference Line

Suppose we want to know how many students weigh more than 100 pounds. One way to do this is to approximately locate the threshold value on the corresponding axis and then compare each data point to that location. But this method has several disadvantages:

- It might be difficult to ascertain whether a point is above or below the threshold when its value is close to the threshold. For example, it's difficult to determine whether John and Barbara are above or below the threshold.
- For points far from the threshold value, you might not get an accurate impression of the difference between the point's value and the threshold value.

We can solve these two problems by drawing a line at the threshold value. In graphs, you can do this by overlaying a reference-line component on top of an x-y chart. The following SAS program overlays a reference line at weight 100 on the graph shown in Figure 3:

```
proc template;
  define statgraph scatterplot;
    begingraph;
      entrytitle "Height and Weight by Sex";
      layout overlay;
        scatterplot x=height y=weight / group=sex name='scatter'
datalabel=name;
         referenceline y=100;
        discretelegend 'scatter';
      endlayout;
   endgraph;
 end;
run;
proc sgrender data=sashelp.class template=scatterplot;
run;
```

Figure 4 shows the output of this program. Note how both problems have been solved. One can easily infer that the John and Barbara fall slightly below the threshold, and that Carol and Henry fall above the threshold.

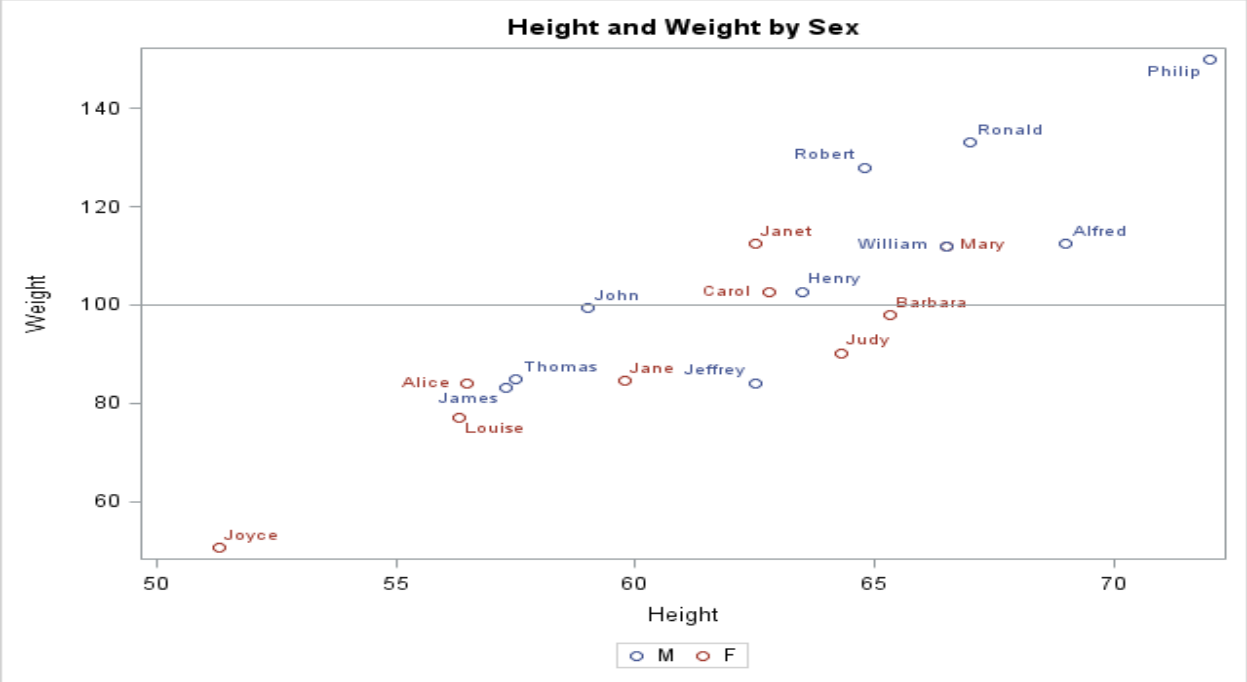

Figure 4. Scatter Plot with Reference Line

BASIC CONTROL CHART USING OVERLAY WITH REFERENCE LINE

A control chart is a graph used to study how a measured value changes over time. Data is plotted in time order. A control chart always has a central line for the average, an upper line for the upper control limit, and a lower line for the lower control limit. These lines are determined from historical data. We can create a control chart in SAS by plotting the measured value using a series plot and overlaying reference lines that show the mean line, lower limit line, and upper limit line. The following program demonstrates how to do this. It plots the temperature of a room measured over several days and displays the reference lines for the mean and limits:

```
data work.temperaturedata;
    format date Date9.;
    date=18250; temp=60;    output;
    date=18251; temp=65;    output;
    date=18252; temp=78;    output;
    date=18253; temp=72;    output;
    date=18254; temp=68;    output;
    date=18255; temp=59;    output;
    date=18256; temp=71;    output;
    date=18257; temp=70;    output;
    date=18258; temp=62;    output;
    date=18259; temp=49;    output;
    date=18260; temp=56;    output;
    date=18261; temp=62;    output;
    date=18262; temp=64;    output;
    date=18263; temp=68;    output;
    date=18264; temp=70;    output;
run;
ods path(prepend)
work.templat(update);
proc template;
  define statgraph simplecontrolplot;
```

```
    begingraph;
      entrytitle "Temperature Monitoring Using Basic Control Chart";
      layout overlay / cycleattrs=true
        xaxisopts=(label="Date" type=discrete)
        yaxisopts=(label="Temperature");
          referenceline y=50/ curvelabel= "lower_limit" curvelabelattrs=

(color=red) lineattrs=(color=red thickness=2);
          seriesplot x=date y=temp / lineattrs=(thickness=2)
name='meanvalues';
           referenceline y=64.933/curvelabel="mean" curvelabelattrs=

(color=red) lineattrs=(color=green thickness =2);
          referenceline y=77/curvelabel="upper_limit" curvelabelattrs=

(color=red) lineattrs=(color=red thickness =2);
       discretelegend 'meanvalues'/;
      endlayout;
    endgraph;
  end;
run;
proc sgrender data=work.temperaturedata template=simplecontrolplot;
run;
```

Figure 5 shows the output of this program. You can see that the graph shows how the temperature varies between the upper and lower limit and how it compares to the mean at every instant. You can control the attributes of the reference lines. Note that the limit lines use increased thickness and red color to make them more prominent. The mean line is shown in green. You can also easily figure out that the temperature on 21st and 28th were outside the limit range. Such points are said to be out of control.

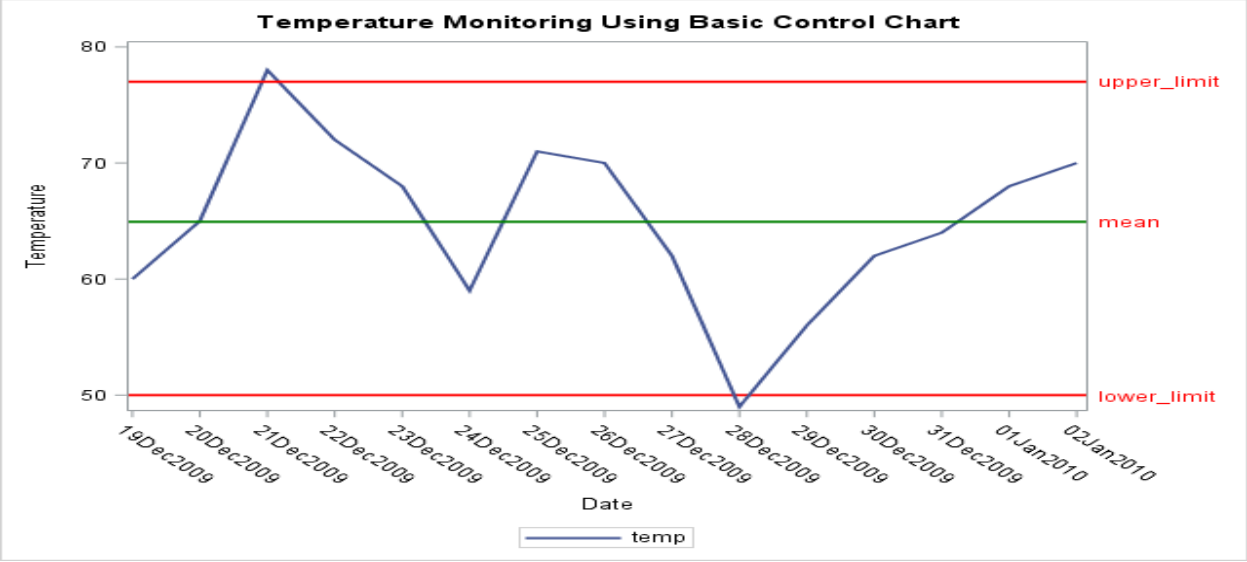

Figure 5. Simple Control Chart Using Reference Lines

BASIC CONTROL CHART IN SAS VISUAL ANALYTICS

You can also use the overlay technique when you create reports in SAS Visual Analytics. To overlay a reference line on a chart, SAS Visual Analytics provides a simple button on the **Properties** tab for the chart that opens a window. You can specify the value for the reference line and its attributes like color

and thickness in this window, which is shown in Figure 6.

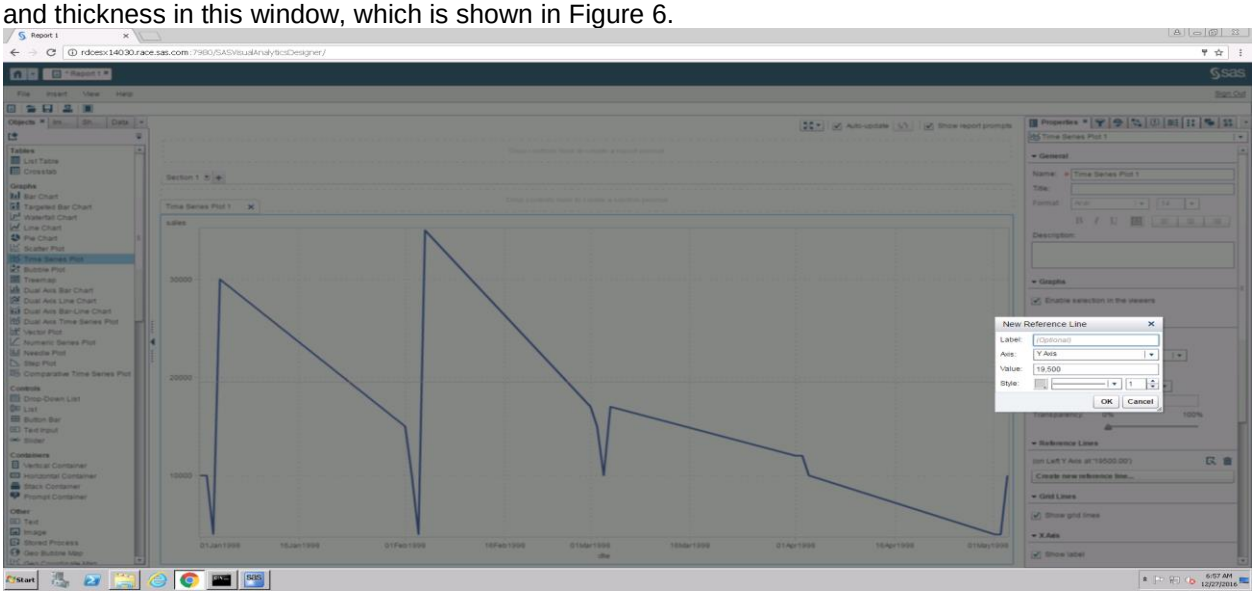

Figure 6. Reference Line Window in SAS Visual Analytics

You can add multiple reference lines. Figure 7 shows a control plot in SAS Visual Analytics that was created by adding multiple reference lines on a time series plot.

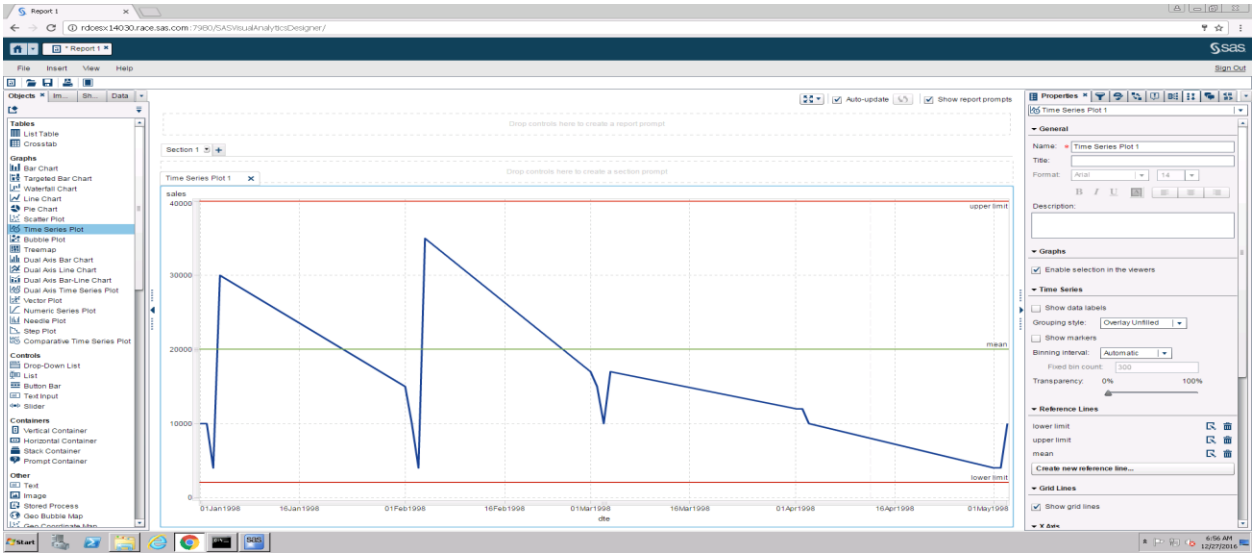

Figure 7. Control Chart in SAS Visual Analytics

ADVANCED CONTROL CHART USING OVERLAY

You have seen that a control chart shows how a value changes between upper and lower control limits. These limits are often constant and are determined using historical data, as shown in the previous section. A control chart that has varying upper and lower limits is called an advanced control chart. Let us look at an example of such a control chart created for the sales data of products over many quarters. We want to display the mean sale across products for each quarter and how it varies between the minimum sale and maximum sale for that quarter. We also want to show how the sales of products vary from the mean by displaying the (mean-standard deviation) and (mean+standard deviation) for each quarter. The following program demonstrates how we can create such an advanced control chart by overlaying a series plot for mean values with band plots for standard deviation ranges and extreme (min/max) ranges:

```
data work.controlchartdata;
    format date Date9.;
    date=18250; minimum=10; maximum=100; std2_lower=20; std2_upper=90;
std1_lower=30; std1_upper=80; mean=55;    output;
    date=18340; minimum=10; maximum=200; std2_lower=15; std2_upper=195;
std1_lower=20; std1_upper=190; mean=105;    output;
    date=18430; minimum=0; maximum=200; std2_lower=10; std2_upper=190;
std1_lower=20; std1_upper=180; mean=100;    output;
   date=18520; minimum=10; maximum=200; std2_lower=15; std2_upper=195;
std1_lower=20; std1_upper=190; mean=105;    output;
    date=18615; minimum=20; maximum=100; std2_lower=25; std2_upper=95;
std1_lower=30; std1_upper=90; mean=60;    output;
    date=18705; minimum=10; maximum=100; std2_lower=20; std2_upper=90;
std1_lower=30; std1_upper=80; mean=55;    output;
    date=18795; minimum=0; maximum=150; std2_lower=15; std2_upper=135;
std1_lower=30; std1_upper=120; mean=75;    output;
    date=18885; minimum=30; maximum=60; std2_lower=36; std2_upper=54;
std1_lower=41; std1_upper=48; mean=45;    output;
    date=18980; minimum=30; maximum=100; std2_lower=40; std2_upper=90;
std1_lower=50; std1_upper=80; mean=65;    output;
    date=19070; minimum=30; maximum=100; std2_lower=35; std2_upper=95;
std1_lower=40; std1_upper=90; mean=65;    output;
    date=19165; minimum=40; maximum=50; std2_lower=41; std2_upper=49;
std1_lower=41; std1_upper=48; mean=45;    output;
    date=19260; minimum=0; maximum=150; std2_lower=15; std2_upper=135;
std1_lower=30; std1_upper=120; mean=75;    output;
    date=19345; minimum=30; maximum=100; std2_lower=35; std2_upper=95;
std1_lower=40; std1_upper=90; mean=65;    output;
    date=19435; minimum=20; maximum=100; std2_lower=25; std2_upper=95;
std1_lower=30; std1_upper=90; mean=60;    output;
    date=19525; minimum=10; maximum=200; std2_lower=15; std2_upper=195;
std1_lower=20; std1_upper=190; mean=105;    output;
run;
ods path(prepend)
work.templat(update);
proc template;
  define statgraph controlplot;
    begingraph;
      entrytitle "Control Chart";
      layout overlay / cycleattrs=true
        xaxisopts=(label="Date" type=discrete)
        yaxisopts=(label="Revenue (million dollars)");
          bandplot x=date limitlower=minimum limitupper=maximum /
name='extremes';
          bandplot x=date limitlower=std1_lower limitupper=std1_upper /
name='standard_deviation';
          seriesplot x=date y=mean / lineattrs=(thickness=4)
name='meanvalues';
          discretelegend 'extremes' 'standard_deviation' 'meanvalues'/;
      endlayout;
    endgraph;
  end;
run;

proc sgrender data=work.controlchartdata template=controlplot;
run;
```

Figure 8 shows the output for this program. Notice how mean values are shown as a line. It's easy to see how it varies in the regions representing the standard deviation ranges and extremes. The legend shows what the line represents and what the two regions represent.

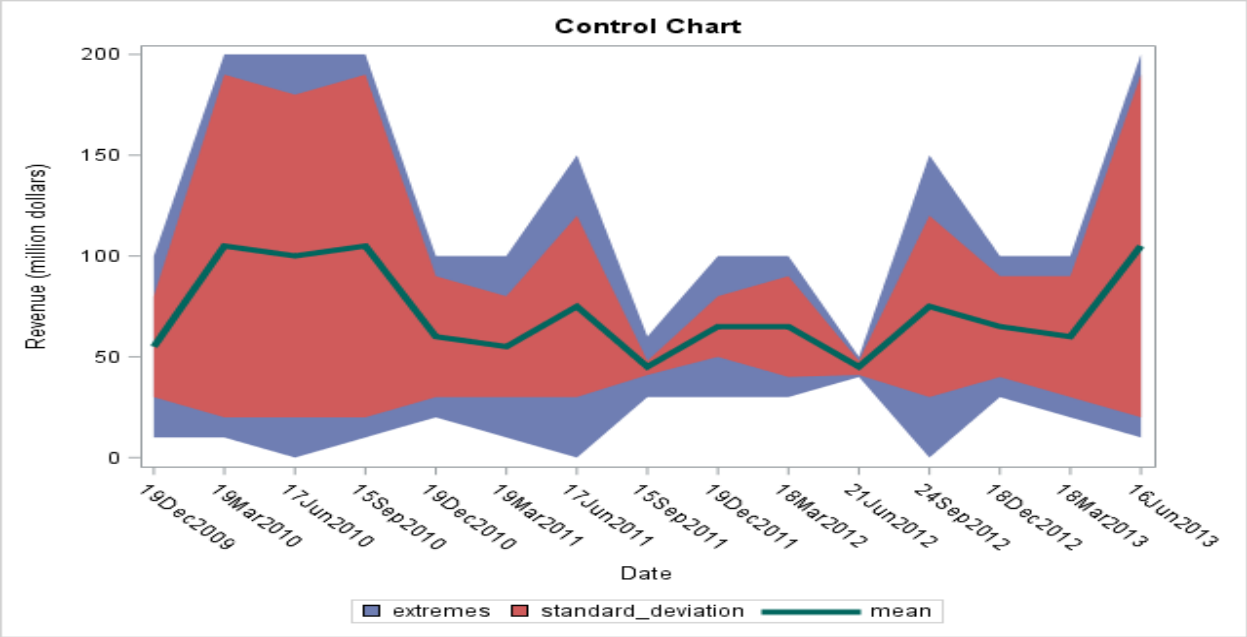

Figure 8. Advanced Control Chart

DEPICTING SPECIAL CONDITIONS

Sometimes you might want to depict data points that meet certain constraints in a special way in a graph. For example, you might have a grouped series plot that shows a line and markers for each group. The following program creates a grouped series plot for stock data, showing the closing value on each day for each stock:

```
proc template;
  define statgraph seriesplot;
    begingraph;
      entrytitle "Tech Stock Trends";
             layout overlay;
        seriesplot x=date y=close / group=stock name='stocks'
lineattrs=(thickness=3) display=(markers);

        discretelegend 'stocks';
      endlayout;
    endgraph;
  end;
run;
proc sgrender data=sashelp.stocks template=seriesplot;
  where date > '31dec1999'd;
run;
```

Figure 9 shows the output for this program.

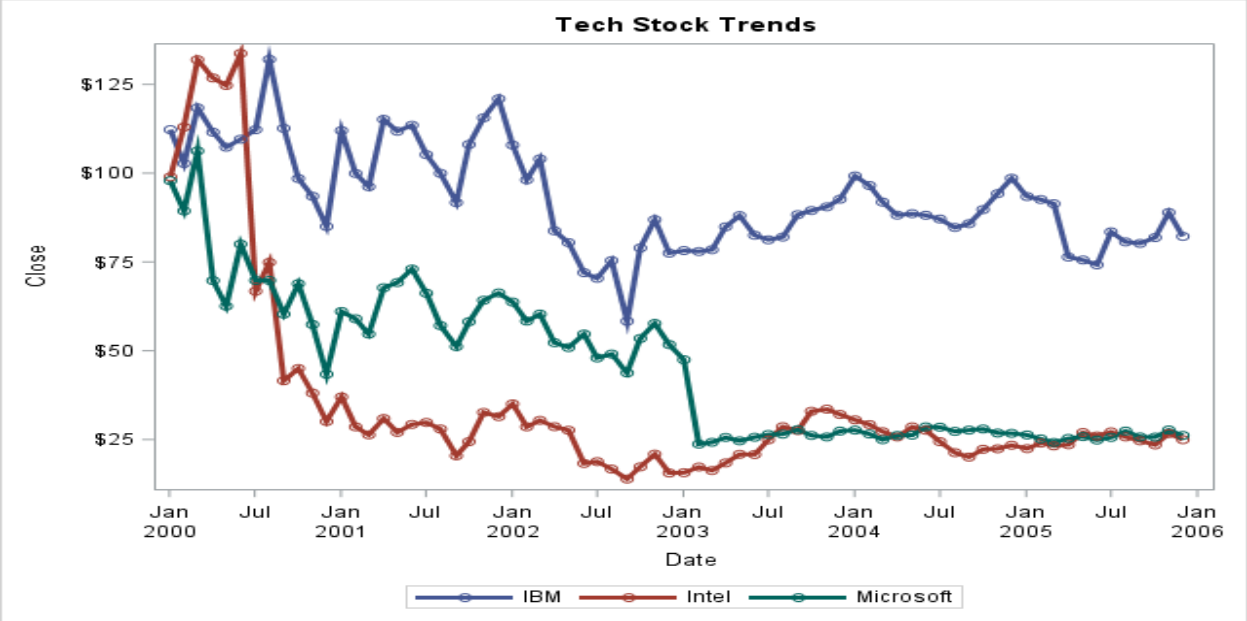

Figure 9. Stock Data

Supposed you want to highlight the instances when a stock was unstable and exhibiting a lot of variation in its value in a single day by assigning the corresponding point a red triangular filler marker. You could do this in the following ways:

- You can use an attribute map to specify the desired marker color and symbol for a whole group. But the points that satisfy the condition might be scattered across groups
- In SAS Visual Analytics, you can use the display rules functionality to assign a specific color to points that satisfy a certain condition. But display rules currently don't allow you to change the marker symbol for those points.

We again solve this by overlaying multiple charts and using some other graph features. The following points summarize the solution:

- Data Preparation: Add a new calculated column to the data set whose value is Stable or Unstable depending on whether the stock was stable or unstable on that particular day.
- Turn off the markers on series plot.
- Overlay a scatter plot on the series plot for drawing the markers.
- Set the new calculated column as the group role for the scatter plot so that the scatter plot draws two types of markers: one for stable stock instances and the other for unstable stock instances.
- Use an attribute map to make the relevant points in the unstable group red-filled triangle shapes. Specify a normal circle shape and black color to the stable group, and set its transparency to 0.6 so that these points fade out a bit to give more prominence to unstable points.
- Add another legend to the chart for the scatter plot to depict what each type of marker means.

The following program shows how to specify this information to achieve the desired output:

```
data work.modStocks;
     set sashelp.stocks;
     if abs(close-open) >15 then stable = 'Unstable';
     else stable = 'Stable';
run;
proc template;
  define statgraph seriesplot;
```

```
    begingraph;
      entrytitle "Tech Stock Trends";
        discreteattrmap name="stabilitymap" / ignorecase=true;
    value "Stable" /
      markerattrs=GraphData1(color=black symbol=circle size=10

transparency=0.6);

    value "Unstable" /
      markerattrs=GraphData2(color=red symbol=trianglefilled size=12);
       enddiscreteattrmap;
       discreteattrvar attrvar=stabilityvar var=stable
    attrmap="stabilitymap";
      layout overlay;
        seriesplot x=date y=close / group=stock name='stocks' lineattrs=

(thickness=3);
          scatterplot x=date y=close/ group=stabilityvar name='stability';
         discretelegend 'stocks'/title='Stock' ;
         discretelegend 'stability'/autoalign=(topright) location=inside;;
      endlayout;
    endgraph;
  end;
run;
proc sgrender data=work.modStocks template=seriesplot;
  where date > '31dec1999'd;
run;
```

Figure 10 shows the output for this program. Note the following characteristics in the output:

- Each line shows a stock and has a unique color.
- The points satisfying the condition are scattered across groups, and they are red, triangle shapes.
- All other points have faint circular black markers.
- The legend at the bottom shows which stock each line represents.
- The legend at the top right shows what condition each marker type represents.

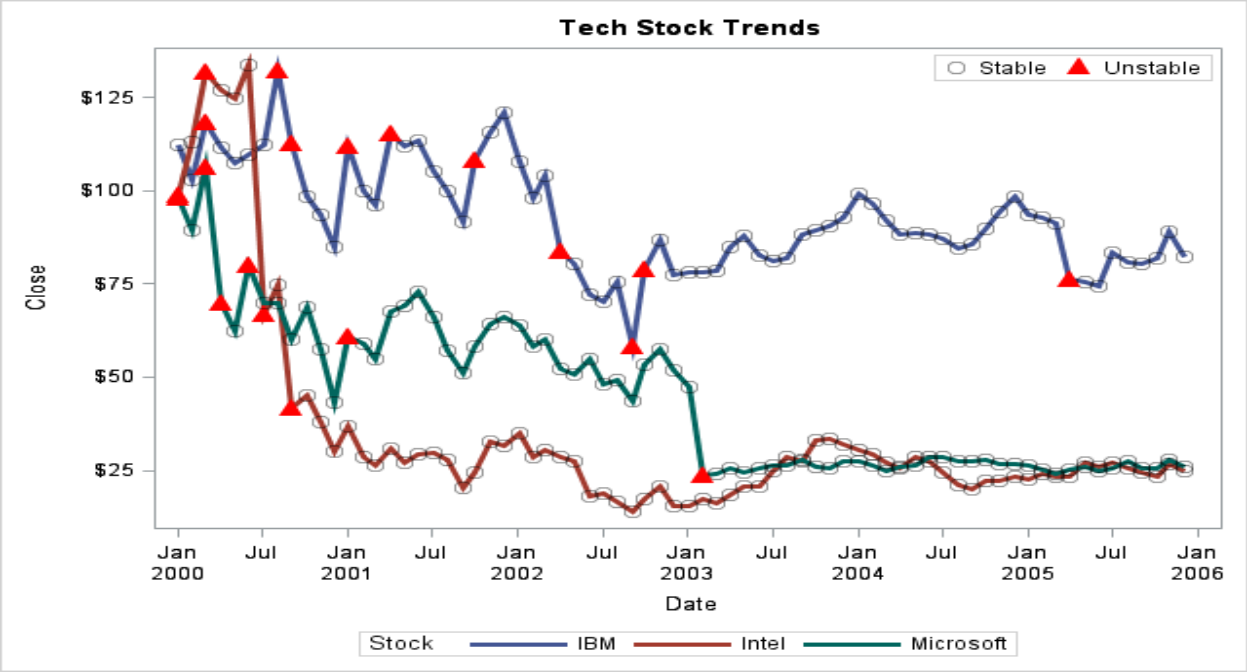

Figure 10. Stock Data Depicting Special Conditions

A slightly modified scenario might require each normal marker to match its line color and specify that only the unstable points should have red triangular markers. To achieve this, we can make the following changes in the code:

- Turn on the markers of the series plot, which automatically makes the markers match line colors.
- Set the transparency of the stable markers to 1 in the attribute map of the scatter plot. This ensures that only unstable, red triangular markers overwrite the series plot markers; the stable ones are not visible, leaving the original series plot markers visible.

The following program contains these and a few other modifications:

```
data work.modStocks;
     set sashelp.stocks;
     if abs(close-open) >15 then stable = 'Unstable';
     else stable = 'Stable';
run;
proc template;
  define statgraph seriesplot;
    begingraph;
      entrytitle "Tech Stock Trends";
        discreteattrmap name="stabilitymap" / ignorecase=true;
    value "Stable" /
      markerattrs=GraphData1(color=black symbol=circle size=10
transparency=1);

    value "Unstable" /
      markerattrs=GraphData2(color=red symbol=trianglefilled size=12);
enddiscreteattrmap;
  discreteattrvar attrvar=stabilityvar var=stable
    attrmap="stabilitymap";
      layout overlay;
```

```
        seriesplot x=date y=close / group=stock name='stocks'
display=(markers)
lineattrs=(thickness=3) markerattrs=(symbol=circlefilled);
          scatterplot x=date y=close/ group=stabilityvar name='stability';
        discretelegend 'stocks'/title='Stock' ;
      endlayout;
    endgraph;
  end;
run;
proc sgrender data=work.modStocks template=seriesplot;
  where date > '31dec1999'd;
run;
```

Figure 11 shows the output for this program.

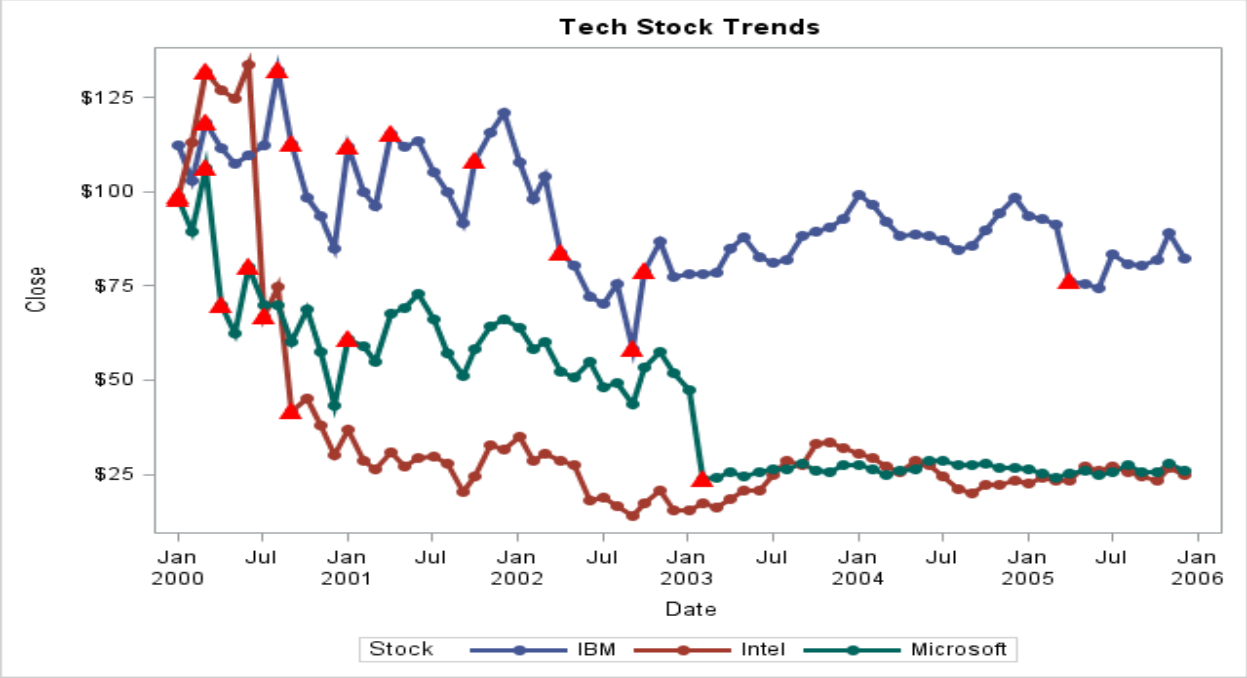

Figure 11. Stock Data with Marker Color Matching Line Color

Note the following characteristics of this output:

- Marker colors of stable stock instances match the line color.
- Only unstable markers are red triangles.
- The legend shows what each line/marker combination represents.

VISUALIZING OVERVIEW WITH SPECIFIC DETAILS USING OVERVIEW AXIS

With the volume of data growing to gigantic sizes, it is sometimes cumbersome to analyze the whole data in one go. You might want to look at an overview of all the data, and then choose a specific subset of data to focus on. You can subsequently move on to a different subset. SAS graphs provide such a window mechanism by which you can choose one subset of data at a time and then move the window to view different parts of the data. You can achieve this by combining an overview axis with other x-y charts. Using an overview axis is not strictly part of the overlay technique as per graph terminology, but it is a useful combinational technique. SAS procedures do not yet support an overview axis, but the grammar is defined as follows:

```
proc template;
  define statgraph overviewaxis;
    begingraph;
      layout overlay / xaxisopts=(name='axis1');;
        seriesplot x=date y=close / group=stock name='stocks';
        discretelegend 'stocks';
      endlayout;
      overviewaxis axis="axis1" / maxplotsize=150 minplotsize=100;
      endoverviewaxis;
    endgraph;
  end;
run;
proc sgrender data=sashelp.stocks template=overviewaxis;
where date > '31dec1999'd;
run;
```

Figure 12 shows the expected output. The output displays an overview component that shows all the data, using the same charts that are defined in the layout overlay. The actual graph above the overview axis shows only a subset of the data. In future, you could use the overview axis syntax to specify the lower and upper bound of the corresponding axis; the main graph would then show the data points between these bounds.

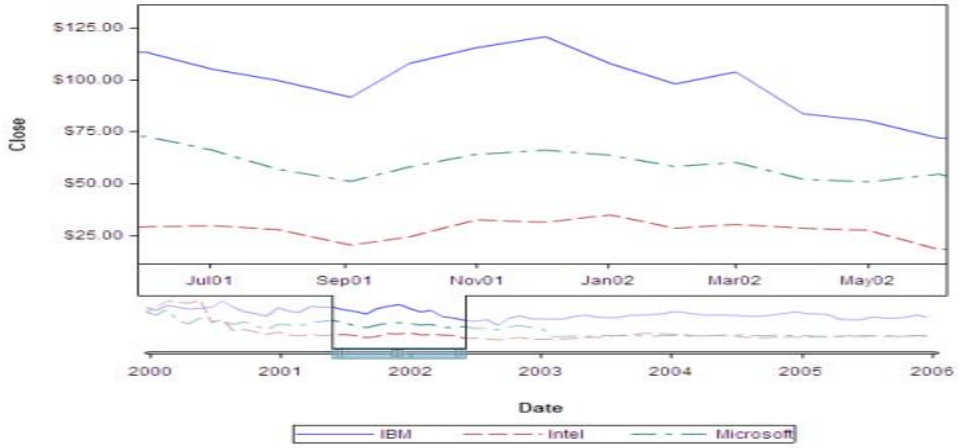

Figure 12. Overview Axis

MOVABLE WINDOW IN SAS VISUAL ANALYTICS

Even though SAS procedures do not yet support an overview axis, SAS Visual Analytics does provide this mechanism in a way that is very simple to use. Figure 13 shows a window in SAS Visual Analytics that shows a time series plot. Note that the **Properties** tab for this chart contains a simple check box for displaying the overview axis. Also note that the main chart shows only the data that lies in the interval selected in the overview axis.

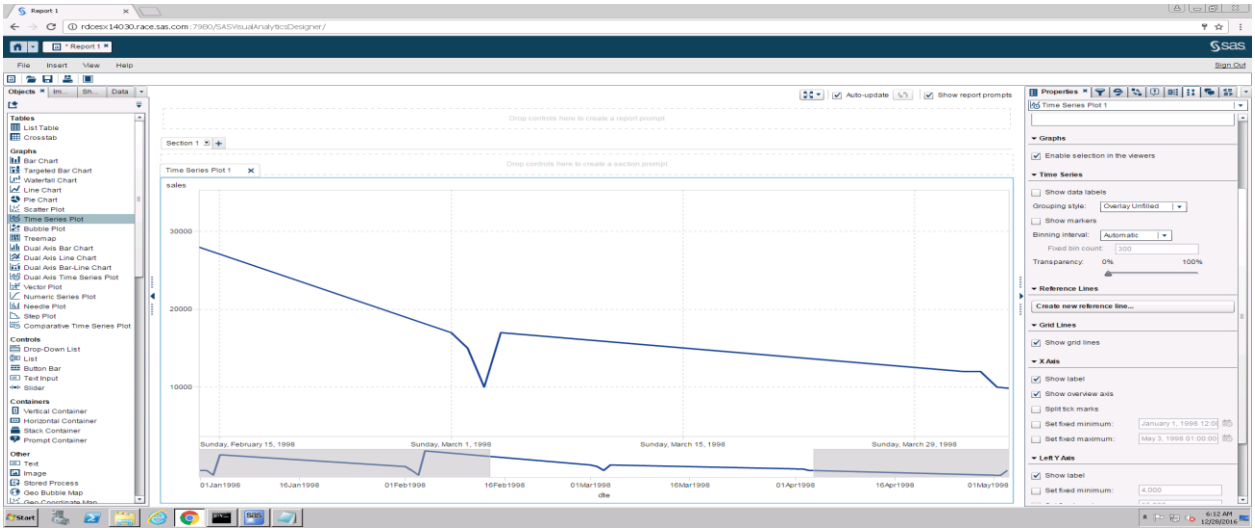

Figure 13. Overview Axis in SAS Visual Analytics

The overview axis bounds are interactive in SAS Visual Analytics. You can move them to choose a different interval. Figure 14 shows the same chart with a different interval selected.

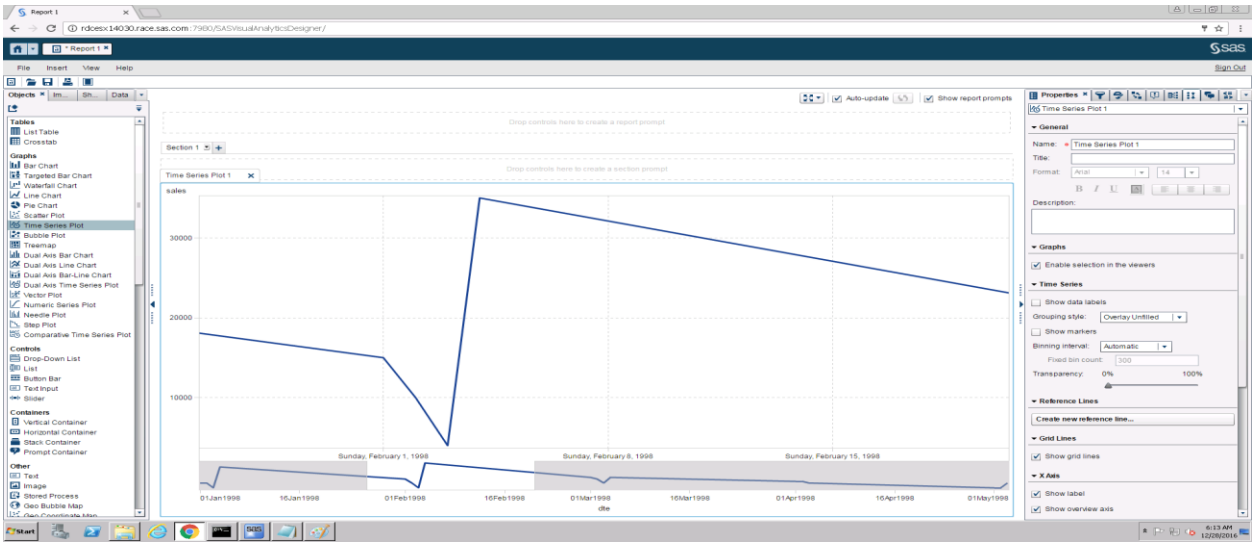

Figure 14. Overview Axis with Shifted Window

VISUALIZING OVERVIEW WITH SPECIFIC DETAILS USING INTERACTIONS

There is another way you can look at an overview of all the data and choose a subset from it to focus on. SAS Visual Analytics enables you to view all the data using one chart, and simultaneously view detailed data for the selected element of this chart in a different chart. (Since this is an interactive functionality involving selections, it is not available by using SAS procedures). Figure 15 shows a SAS Visual Analytics window with two charts that represent the overview and detailed data. In this example, the pie chart shows the distribution of all cars across dealers. The bar chart shows the distribution of all cars across manufacturers.

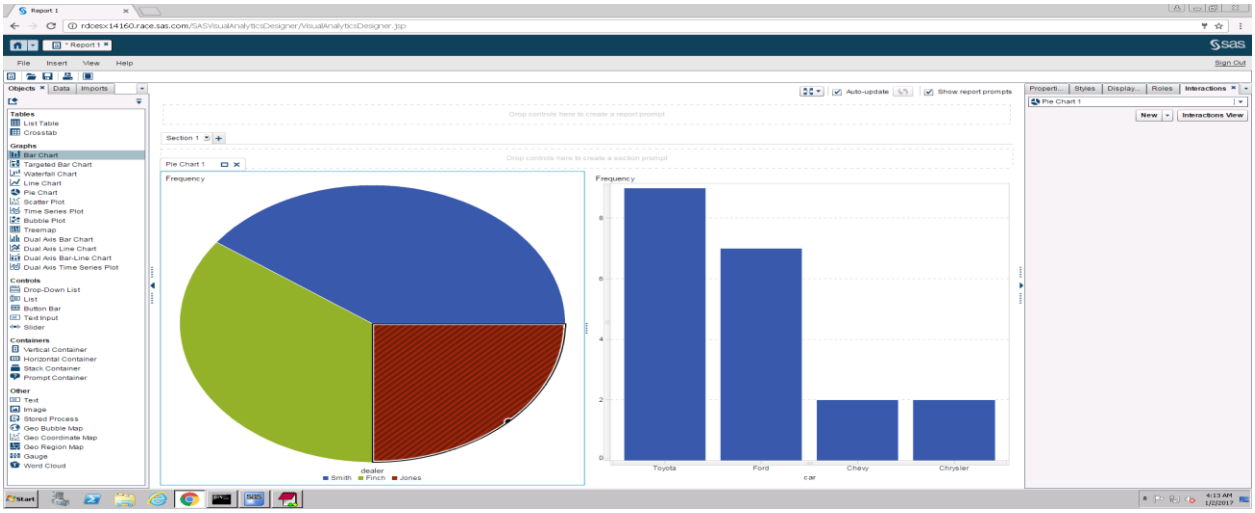

Figure 15. Two Charts for Overview and Detailed View

To use the Interactions view, click the **Interactions** tab for either chart. Then, click **Interactions View**. Define a filter interaction from the first chart to the second by clicking on the first chart and dragging the mouse pointer to the second chart. Figure 16 demonstrates this action for our scenario.

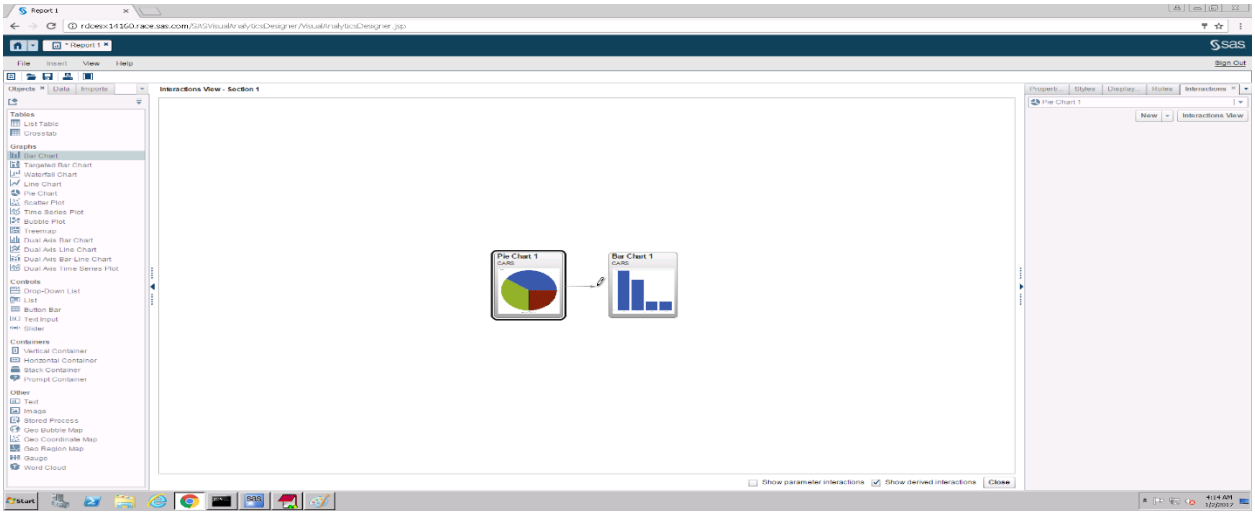

Figure 16. Interaction View

Click **Close** in the bottom right corner of the Interactions view to go the regular view. After you have defined the filter interaction, clicking on any element of the first chart causes the second chart to display only the data that corresponds to the selected element in the first chart. Figure 17 shows how clicking on the slice for dealer Finch causes the bar chart to display the distribution of cars across manufacturers for

only the Finch dealer.

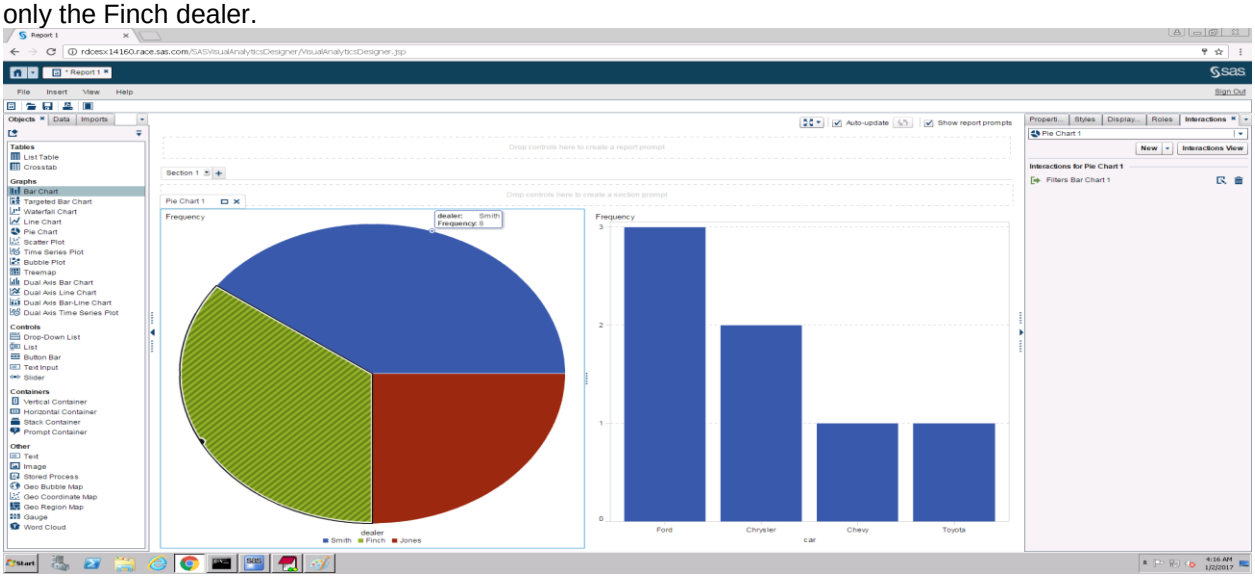

Figure 17. Filter Interaction Example

Figure 18 shows the filter interaction for a different dealer. Clicking dealer Smith causes the bar chart to show only the data for Smith.

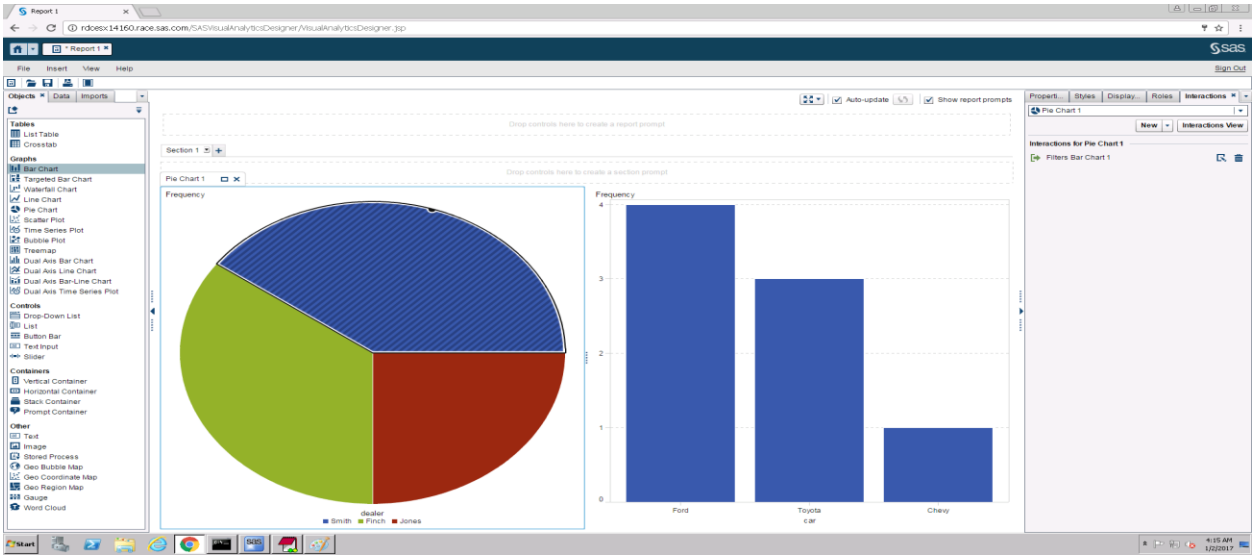

Figure 18. Filter Interaction Sample II

CONCLUSION

The examples described in this paper demonstrate the power of combining graphs to create advanced visualizations. These examples are just a handful of the numerous possible combinations. I encourage you to explore this technique further to create the visualizations that meet your particular needs. Not only can you combine different graphs, but you can also set options on the participating graphs to generate more variations of the advanced visualizations.

REFERENCES

SAS Graph Documentation (http://support.sas.com/documentation)

ACKNOWLEDGMENTS

This paper is a result of my interactions with product teams in helping them create visualizations that meet their needs. I sincerely thank the team members with whom I interacted to create these visualizations. I also thank Vijay Chougule for his valuable suggestions in rephrasing the abstract. I also thank Kush Shukla for evaluating these examples and providing valuable inputs.

CONTACT INFORMATION

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

Vineet Raina
SAS Research and Development, India
+91 (20) 4911 8339
Vineet. Raina@sas.com
https://www.linkedin.com/in/vineet-raina-8b83a254