

## Easy, Elegant, and Effective SAS® Graphs: Inform and Influence with Your Data

LeRoy Bessler, Bessler Consulting & Research, Fox Point, Milwaukee, Wisconsin, USA

### Abstract

Are you an existing SAS/GRAPH® user who would like to create charts that are more communication-effective? Or are you one of those software users who crunch their data with the SAS System, but then export their output to some other tool to graph it? This tutorial assumes no prior knowledge of SAS/GRAPH. It cuts through all the documentation and bypasses the “Options OverChoice” to empower you to make elegant and effective pie charts, bar charts, and trend charts—to get them right from the start. You will come away with widely applicable design principles and examples, and with supporting code that you can understand, and can point at your own data when you get back to the office. SAS/GRAPH *can* be easy, and, based on good design and good examples, communication-effective use of SAS/GRAPH gives you The Power to Show: to inform and influence, to reveal and persuade, with charts and data. The graphs were created with SAS Release 8.2, the PNG driver, and Windows 98 Second Edition, and then imported into this paper, which was written with Microsoft Word 2000.

### Introduction

This is a tutorial about effective visual communication of information, taught with examples and with all—and only—the code required. It is impossible to present all options available for SAS/GRAPH statements used, and all assignment possibilities for the options. If code must be modified to suit your situation, see “SAS/GRAPH Software: Reference, Version 8”, available from SAS Institute Inc. Also get the documents on the software enhancements for Releases 8.1 and 8.2 of the SAS System. My scope is limited to common graphs used for management reports and presentations, and what fits in the page and time constraints. The code is also available via email in a zip file. If you have comments or questions, or suggestions on what to cover in a future edition, please send them. The stand-up presentation includes explanation of the code, but here there are only a few comments in or near the code. See the end of the paper for discussion of all syntax used in the examples, the common preliminary code not listed in the examples, input data, device drivers, and how to web publish the graphs.

### Easy, Elegant, and Effective—The Power of Simplicity

**Easy.** The example code should work for you, by pointing it at your data and making obvious adaptive modifications to a few statements.

**Elegant.** The defaults of software are not intended to be elegant. They are intended to work, and usually give an adequate result if you do not have high expectations. To get a better result entails customization. However, customization that decorates, or needlessly complicates, does not yield elegance. Simplicity is almost always a key to elegance. Complexity also can yield elegance, if it incorporates only the essential.

**Effective.** “Effective” certainly means “It works.” But here I mean “communication-effective”. Needless complexity, confusion, and distortion are common obstacles to effective graphic communication. Graphs accelerate inferences and decisions, precise data (often tabular) assures reliable inferences and decisions. The designs here combine quickly assimilated pictures, and as much precision as possible.

**The Power of Simplicity.** The design style used here is simple. Traditional graphic paraphernalia, a holdover from the olden days of grid paper, pen, and ink, are stripped away. My focus is on the data, and what it is doing. “Let the data talk.” The graph should reveal the data and its characteristics, and should persuade the viewer—if there is a legitimate inference that can be drawn. It should reveal and persuade, inform and influence.

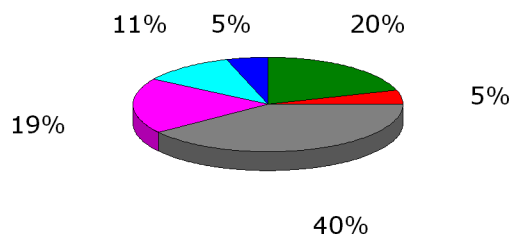
### Fonts and Colors

For titles and footnotes, the Georgia font is used. For graph “body” text, which often must be smaller, the Verdana font is used. These Windows TrueType fonts were designed by Matthew Carter for improved

readability on the web. For the web, it is best to use the Browser-Safe subset of RGB colors. For more about them, and why they are important, please see Reference 1. Default colors are specified by the device driver, default fonts by GOPTIONS ftext= in the Common Preliminary Code.

### Examples: What To Do, and What Not To

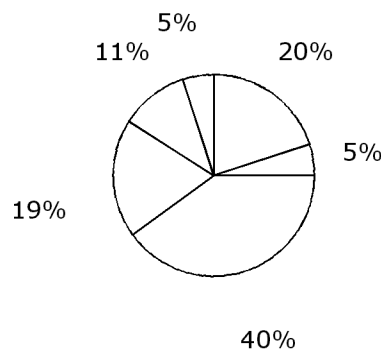
**Figure 1: 3D Pie Chart.**  
Always Distorts Apparent Relative Size of Slices.



5% slice at rear is twice as big as 5% slice at right  
11% slice is almost as big as 19% slice

I did not deliberately pick the values and the arrangement. The above is a problem encountered with real business data, with the slices ordered by the software default algorithm (based on the suppressed slice names).

**Figure 2: 2D Pie Chart. Simpler is Better.**  
Always Presents Accurate Relative Size of Slices.



Here is the code used to create Figure 2:

```

title2 h=1.00 f='Georgia'
j=L ' Figure 2: ' c=CX0000FF '2D Pie Chart. Simpler is Better.'
j=L ' Always Presents Accurate Relative Size of Slices.';
footnote;
pattern1 v=pempty r=6; /* empty pie slices */
goptions vpos=16 vsize=2.40 IN ymax=2.40 IN ypixels=720;
goptions hpos=34 hsize=3.25 IN xmax=3.25 IN xpixels=975;
proc gchart data=DataForSimpleCharts;
pie Name /
sumvar=Value
noheading /* suppress default pie chart heading */
outline=CX000000 /* color of pie slice outline is Black.
The default is COUTLINE=SAME, which means "same color as

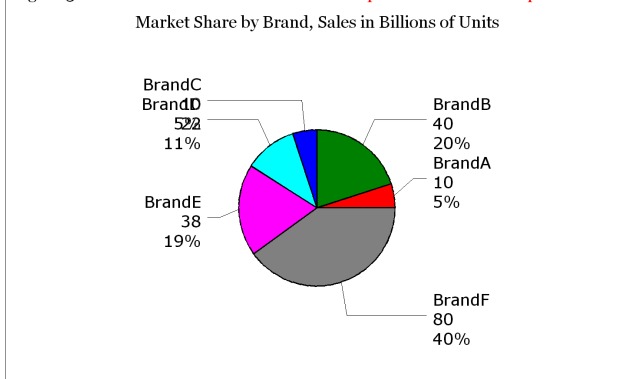
```

slice area fill”, presumably controlled by PATTERN statements. However, though v=empty leaves pie slices empty, SAS/GRAPH colors the slice outlines using the driver's default color list IF you accept the default COUTLINE=SAME. \*/

```
woutline=2 /* thicken the pie outline */
slice=none /* no Name labels */
value=none /* no Value labels */
percent=outside; /* Percent of Whole labels outside the pie */
run; quit;
```

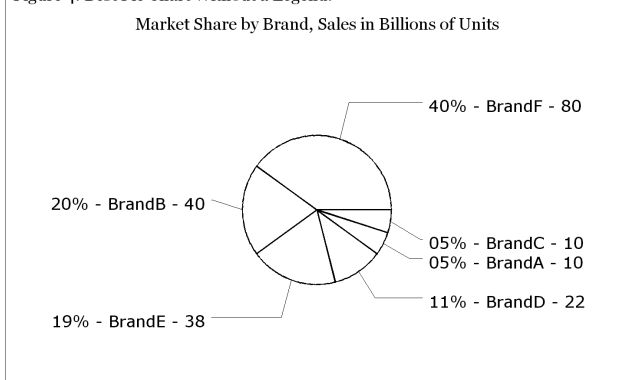
To focus on the 3D problem, SLICE= and VALUE= (used to specify location of the SLICE and VALUE labels) were set to “NONE”. Below is a standard use of the SAS/GRAPH 2D pie chart, displaying slice Name, slice Value, and slice Percent of Whole. You can position this information INSIDE, OUTSIDE, or (with an) ARROW. The ordering of the slices is, by default, alphabetical by slice Name. There is no straightforward way to cure The Overlap Problem. Changing the position of the labels to OUTSIDE does not help. Ordering the slices by DESCENDING slice Value does not help. Consolidating the two smaller slices to end up with fewer to label would defeat the objective of maximum communication. And shuffling the slices around arbitrarily has no communication value. **You CAN build SAS/GRAPH applications that work right the first time every time.** A powerful feature of SAS/GRAPH is that it IS well-suited, after you find a solution, to production applications—that run “hands-off”, with no post-run manipulation of output, and no iterative adjustments and reruns.

Figure 3: Standard Form of 2D Pie Chart. **Susceptible to Label Overlaps.**



I recommend the usually reliable solution in Figure 4. (Both of these figures were created 4.79 inches wide. When imported to this document, they shrunk to 3.25 inches wide. That is why all of their text is smaller than that of the other figures in this paper.)

Figure 4: Best Pie Chart Without a Legend.



There is no communication need to fill the pie slices with color. The slices are laid down in order of DESCENDING slice Value. “Show them what’s important.” Here is the code used to create Figure 4:

```
proc means data=DataForSimpleCharts noprint sum;
var Value;
output out=PieTotal sum=TotalValue; run;
```

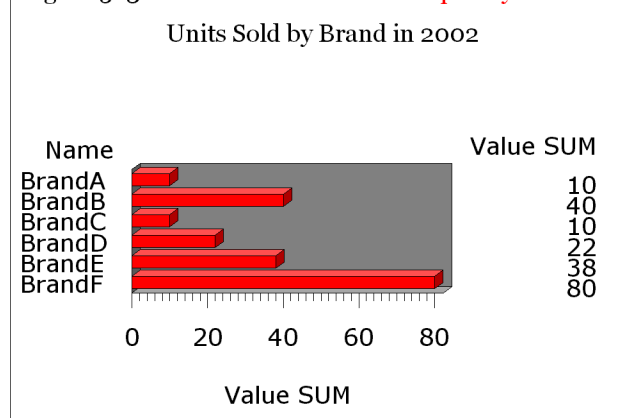
```
data SliceNameWithPercentAndValue;
length NameWithPercentAndValue $ 17;
set DataForSimpleCharts;
if _N_ eq 1 then set PieTotal;
Percent = (Value / TotalValue) * 100;
NameWithPercentAndValue =
trim(left(put(Percent,Z2.))) || '%' || ' ' ||
trim(left(Name)) || ' - ' || trim(left(put(Value,2.)));
run;

title2 h=1.00 f='Georgia' j=L
' Figure 4: Best Pie Chart Without a Legend.';
title3 h=0.50 f=none ' ';
title4 h=1.00 f='Georgia' j=C
'Market Share by Brand, Sales in Billions of Units';
footnote;
pattern1 v=empty r=6;
goptions vpos=21 vsize=3.15 IN ymax=3.15 IN ypixels=945;
goptions hpos=50 hsize=4.79 IN xmax=4.79 IN xpixels=1437;
proc gchart data=SliceNameWithPercentAndValue;
pie NameWithPercentAndValue /
sumvar=Value noheading coutline=CX000000 woutline=2
descending /* order pie slices from large to small */
slice=arrow /* connect label outside pie to slice with arrow */
value=none percent=none;
run; quit;
```

Recent releases of SAS/GRAPH are able to provide a pie legend, but limit what you can put in the legend. A much better solution, Figure 19, is postponed till later in the tutorial, due to its complexity.

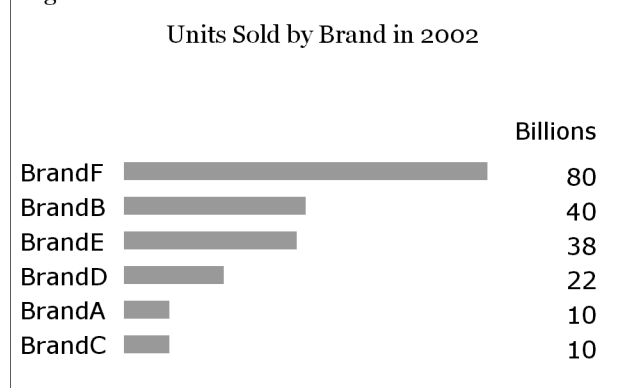
Horizontal bar chart labels cannot suffer the overlap problems of pie charts, and cannot be too wide for their bars like vertical bar charts. But, in Figure 5, **the third dimension adds no communication value.**

Figure 5: 3D Bar Chart. **Needless Complexity.**



I recommend the obviously better 2D horizontal bar chart in Figure 6.

Figure 6: Best Horizontal Bar Chart.



Here are the strengths of Figure 6. The bars are ranked from highest to lowest. "Show them what's important." There is nothing in the image that is dispensable. The bars are lightly shaded. A color would add nothing. Using BLACK as the area fill would cause the image to be unnecessarily dominated by the area fill. Using an EMPTY area fill (hollow rectangles) can cause visual confusion, especially when there are numerous bars, between what are the bars and what are the spaces.

Tip: In a horizontal bar chart with numerous entries (e.g., some measurement for each of the fifty United States of America), rather than ordering bars by size, it may be more useful to support quick look-up of a specific entry with alphabetical order.

Here is the code to create the simple masterpiece of Figure 6:

```

title2 h=1.00 f='Georgia' j=L
' Figure 6: Best Horizontal Bar Chart.';
title3 h=0.50 f=none '';
title4 h=1.00 f='Georgia' j=C
'Units Sold by Brand in 2002';
footnote;
pattern1 v=solid c=CX999999 r=6; /* use light browser-safe gray */
axis1 label=none major=none minor=none style=0;
axis2 label=none major=none minor=none style=0 value=none;
goptions vpos=15 vsize=2.25 IN ymax=2.25 IN ypixels=675;
goptions hpos=34 hsize=3.25 IN xmax=3.25 IN xpixels=975;
proc gchart data=DataForSimpleCharts;
hbar Name /
    sumvar=Value
    sumlabel='Billions' /* label for column of Values */
    width=0.6 /* adjust width of bars */
    space=0.6 /* adjust spacing between bars */
    maxis=axis1 /* axis1 statement defines the Names axis */
    raxis=axis2 /* axis2 statement defines the Values axis */
    descending;
run; quit;

```

With a minor enhancement to Figure 6 we can achieve the considerable benefit provided by Figure 7 below. Namely, by specifying five parameters in the HBAR statement, we get what can be used in lieu of a pie chart in situations where no pie chart is feasible. The alternative of combining multiple small slices into OTHER—is anti-communicative, and just invites two questions, "What is in OTHER? How big are the pieces?" **As a communication tool, your graph should answer questions, not create them.**

Figure 7: Pie Chart Alternative.  
Horizontal Bar Chart Can List Percent of Total.

#### Units Sold and Market Share by Brand in 2002

	Billions	Percent
BrandF	80	40
BrandB	40	20
BrandE	38	19
BrandD	22	11
BrandA	10	5
BrandC	10	5

Useful when a pie chart is infeasible

Here is the code used to create Figure 7:

```

title2 h=1.00 f='Georgia' j=L
' Figure 7: Pie Chart Alternative.' j=L c=CX0000FF
' Horizontal Bar Chart Can List Percent of Total.';
title3 h=0.50 f=none '';

```

```

title4 h=1.00 f='Georgia' j=C
'Units Sold and Market Share by Brand in 2002';
footnote1 h=1.00 f='Georgia' j=C c=CX0000FF
'Useful when a pie chart is infeasible';
footnote2 h=0.50 f=none '';
pattern1 v=solid c=CX999999 r=6; /* use light browser-safe gray */
axis1 label=none major=none minor=none style=0;
axis2 label=none major=none minor=none style=0 value=none;
goptions vpos=17 vsize=2.55 IN ymax=2.55 IN ypixels=765;
goptions hpos=34 hsize=3.25 IN xmax=3.25 IN xpixels=975;
proc gchart data=DataForSimpleCharts;
hbar Name /
    freq=Value
    freq
    freqlabel='Billions'
    percent
    percentlabel='Percent'
    width=0.5 space=0.5 maxis=axis1 raxis=axis2 descending;
run; quit;

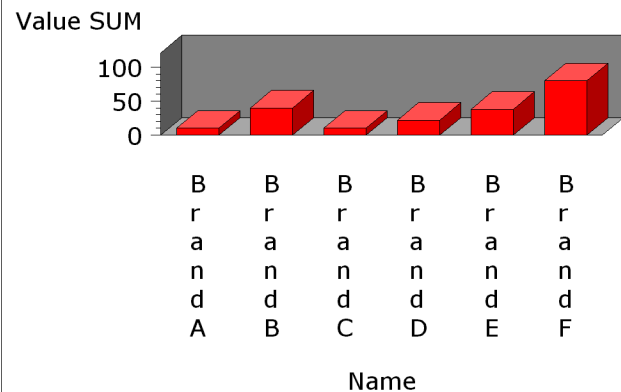
```

The use of FREQ= rather than SUMVAR= above is counterintuitive, but is necessary to get the result shown.

Figure 8 shows you a default 3D vertical bar chart.

Figure 8: 3D Bar Chart. **Difficult to Use.**

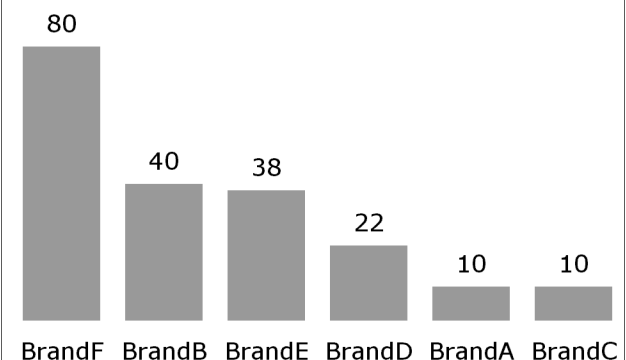
#### Units Sold (in Billions) by Brand in 2002



Here, too, **the third dimension adds no value.** Though the bar labels are short, they cannot fit horizontally. An immensely better result is shown in Figure 9. This is, in effect, a rotation of Figure 6 through 90 degrees. All the design principles are the same.

Figure 9: Best Vertical Bar Chart.

#### Units Sold (in Billions) by Brand in 2002



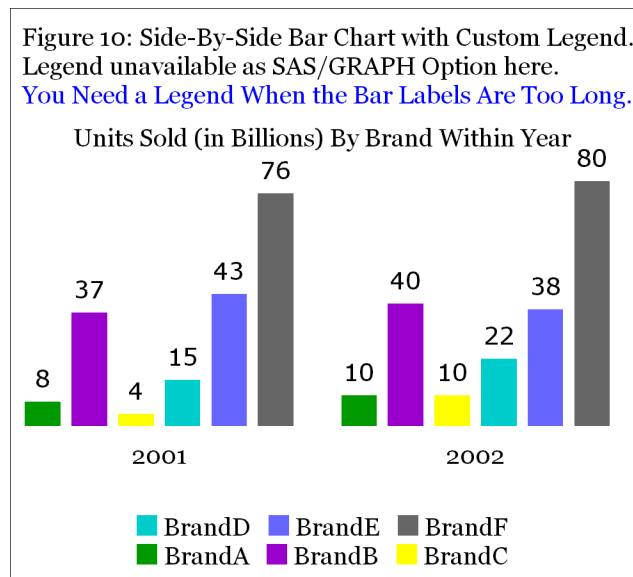
Here is the code used to create Figure 9:

```

title2 h=1.00 f='Georgia' j=L
' Figure 9: Best Vertical Bar Chart.';
title3 h=0.50 f=none '';
title4 h=1.00 f='Georgia' j=C
'Units Sold (in Billions) by Brand in 2002';
footnote;
pattern1 v=solid c=CX999999; /* use light browser-safe gray */
axis1 label=none major=none minor=none style=0;
axis2 label=none major=none minor=none style=0 value=none;
goptions vpos=17 vsize=2.55 IN ymax=2.55 IN ypixels=765;
goptions hpos=34 hsize=3.25 IN xmax=3.25 IN xpixels=975;
proc gchart data=DataForSimpleCharts;
vbar Name /
  sumvar=Value
  sum
  descending
  maxis=axis1 raxis=axis2;
run; quit;

```

When the data requires two-level classification, the solution is the side-by-side bar chart in Figure 10.



The legend has to be “faked” by building it with footnotes because PROC GCHART does not support a legend for this situation. Here is the code used to create Figure 10:

```

title2 h=1.00 f='Georgia' j=L
' Figure 10: Side-By-Side Bar Chart with Custom Legend.' j=L
' Legend unavailable as SAS/GRAPH Option here.' j=L c=CX0000FF
' You Need a Legend When the Bar Labels Are Too Long.';
title3 h=0.50 f=none '';
title4 h=1.00 f='Georgia' j=C
'Units Sold (in Billions) By Brand Within Year';
footnote1 /* create the legend */ h=1.00 j=C
f='Monotype Sorts' c=CX009900 '6E'X /* '6E'X is the square */
f='Georgia' c=CX000000 ' BrandA '
f='Monotype Sorts' c=CX9900CC '6E'X
f='Georgia' c=CX000000 ' BrandB '
f='Monotype Sorts' c=CXFFFF00 '6E'X
f='Georgia' c=CX000000 ' BrandC ' j=C /* now start a new line */
f='Monotype Sorts' c=CX00CCCC '6E'X
f='Georgia' c=CX000000 ' BrandD '
f='Monotype Sorts' c=CX6666FF '6E'X
f='Georgia' c=CX000000 ' BrandE '
f='Monotype Sorts' c=CX666666 '6E'X
f='Georgia' c=CX000000 ' BrandF';
footnote2 h=0.50 f=none '';

```

```

pattern1 v=solid c=CX009900; /* 2 steps darker than RGB green */
pattern2 v=solid c=CX9900CC; /* purple */
pattern3 v=solid c=CXFFFF00; /* yellow */
pattern4 v=solid c=CX00CCCC; /* 1 step darker than RGB cyan */
pattern5 v=solid c=CX6666FF; /* 2 steps lighter than RGB blue */
pattern6 v=solid c=CX666666; /* medium dark gray */
axis1 label=none major=none minor=none style=0 value=none;
axis2 label=none major=none minor=none style=0 value=none
offset=(0.5,0);
axis3 label=none value=(f='Georgia') nobrackets;
goptions vpos=20 vsize=3.00 IN ymax=3.00 IN ypixels=900;
goptions hpos=34 hsize=3.25 IN xmax=3.25 IN xpixels=975;
proc gchart data=UnitsSoldByBrandAndYear;
vbar Group / sumvar=Value sum maxis=axis1 raxis=axis2
  group=Year /* group the midpoints (Brands) by Year */
  gaxis=axis3 /* axis3 statement defines Group (Year) axis */
  patternid=midpoint; /* vary pattern (color) by Brand */
run; quit;

```

Regrettably popular is use of the Stacked Bar Chart. How can you estimate the measurement for the upper bar in Figure 11? Even with a larger vertical scale the problem persists. This graph would be worse if I had not supplied a short y-axis label and shrunk the x-value height.

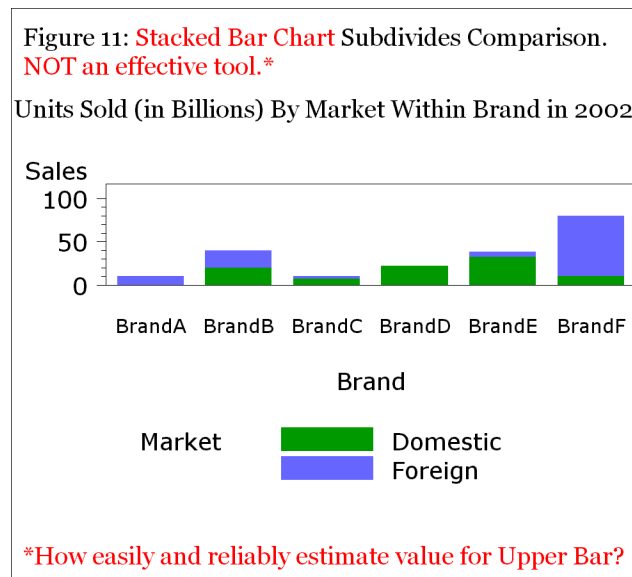
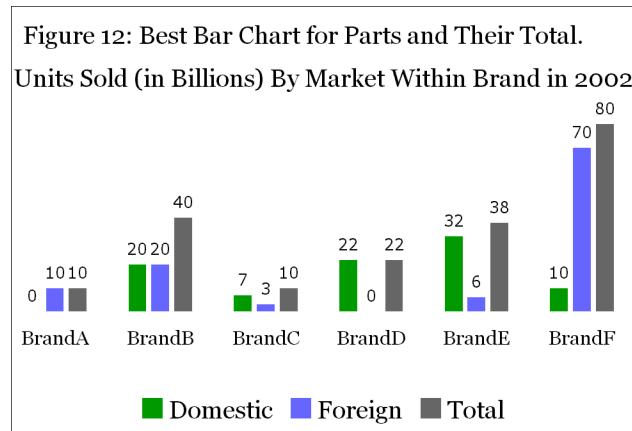


Figure 12 is the better solution when you want to graph multiple components and their total. As in Figure 10, I had to “fake” the legend because PROC GCHART does not support a legend for this situation.



The code to create Figure 12 is the same as that used for Figure 10, except that it includes some preprocessing of the input (and, as you can see, it shortens the graph—to fit the Figure on this page). Here it is:

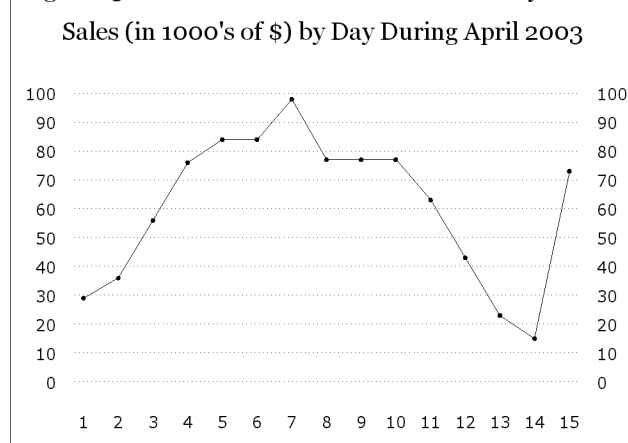
```
proc sort data=UnitsSoldByBrandAndMarket; by Brand Market; run;
```

```
data UnitsSoldWithTotals;
length Market $ 3;
retain Total 0;
set UnitsSoldByBrandAndMarket;
by Brand;
Total = Total + Value;
output;
if last.Brand;
Market = 'Total';
Value = Total;
output;
Total = 0;
run;

title2 h=1.00 f='Georgia' j=L
' Figure 12: Best Bar Chart for Parts and Totals.';
title3 h=0.50 f=none ' ';
title4 h=1.00 f='Georgia' j=C
'Units Sold (in Billions) By Market Within Brand in 2002';
footnote1 h=1.00 j=C
f='Monotype Sorts' c=CX009900 '6E'X
f='Georgia' c=CX000000 ' Domestic '
f='Monotype Sorts' c=CX6666FF '6E'X
f='Georgia' c=CX000000 ' Foreign '
f='Monotype Sorts' c=CX666666 '6E'X
f='Georgia' c=CX000000 ' Total';
footnote2 h=0.50 f=none ' ';
pattern1 v=solid c=CX009900;
pattern2 v=solid c=CX6666FF;
pattern3 v=solid c=CX666666;
axis1 label=none major=none minor=none style=0 value=none;
axis2 label=none major=none minor=none style=0 value=none
offset=(0.5,0);
axis3 label=none value=(f='Georgia' h=0.80);
goptions htext=0.70; /* size of bar-end value labels */
goptions vpos=15 vsize=2.25 IN ymax=2.25 IN ypixels=675;
goptions hpos=34 hsize=3.25 IN xmax=3.25 IN xpixels=975;
proc gchart data=UnitsSoldWithTotals;
vbar Market / group=Brand
sumvar=Value sum maxis=axis1 raxis=axis2
gaxis=axis3 patternid=midpoint;
run; quit;
```

Now let's take up what is probably the commonest use of graphic presentation of data, the trend chart. The vertical bar chart is sometimes used to track a trend, but the plot line is commoner.

Figure 13: Fine lines to facilitate estimation of y-values.



Unless the trend line is rather smooth, automated annotation of each plot point is difficult, as we shall see. So what can we do? Figure 13 makes it easy to achieve a reasonable estimate of the y-values, by using reference lines and reduced size dots.

Note that the y-axis starts at 0, even though no y-value comes below the \$10,000 reference line. This choice of starting range is deliberate.

Tip: Usually start the vertical axis at zero. This smoothes out, somewhat, changes in the y-value. Allocating all the plot space only to the range of y-values present amplifies the size of changes, and can cause needless enthusiasm or anxiety about the magnitude of a change. Assessing the impact of a change has to be more sophisticated than reacting to a graph. E.g., percent change may be a better basis.

Here is the code used to create Figure 13:

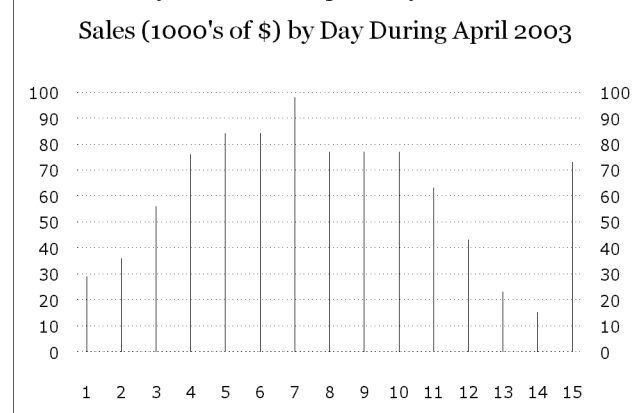
```
title2 h=1.00 f='Georgia' j=L
' Figure 13: Fine lines to facilitate estimation of y-values.';
title3 h=0.50 f=none ' ';
title4 h=1.00 f='Georgia' j=C
'Sales (in 1000's of $) by Day During April 2003';
footnote;
axis1 label=none major=none minor=none style=0 value=(h=0.70);
axis2 label=none major=(c=CXFFFFFF) minor=none style=0
value=(h=0.70);
axis3 label=none major=(c=CXFFFFFF) minor=none style=0
value=(h=0.70 j=L); /* left justify tick mark values for this axis */
symbol1 v=dot h=0.25 i=join;
goptions vpos=17 vsize=2.55 IN ymax=2.55 IN ypixels=765;
goptions hpos=34 hsize=3.25 IN xmax=3.25 IN xpixels=975;
proc gplot data=DataForTrendPlot(where=(YearMonth='200304'));
plot Sales*DayOfMonth /
haxis=axis1 /* axis1 statement defines the horizontal axis */
vaxis=axis2 /* axis2 statement defines the vertical axis */
vzero /* start vertical axis at zero */
autovref /* reference lines at major y-axis tick marks */
lvref=33; /* line type 33 for reference lines */
plot2 Sales*DayOfMonth=1 /* to get the right-hand vertical axis */
vaxis=axis3 /* axis3 statement defines right-hand axis */
vzero; /* start vertical axis at zero */
run; quit;
```

When a trend line is very dense along the x-axis (i.e., has many dates to plot), the correct x-value to associate with a trend plot point can be hard to determine. Figure 14 shows the ultimate in ability to estimate plot point coordinates.

Our scope here is static images—ones that can be printed, can be photocopied, etc. For a web-enabled trend display, one would have some other options. You could implement mouseover text, flyover text, or whatever you choose to call that little box that pops up when you rest the mouse on an item of interest. And/or you could implement drill-down to a hyperlinked table. (For such solutions, see Reference 2.)

Here we have what is, in effect, the world's thinnest vertical bar chart, not a trend line.

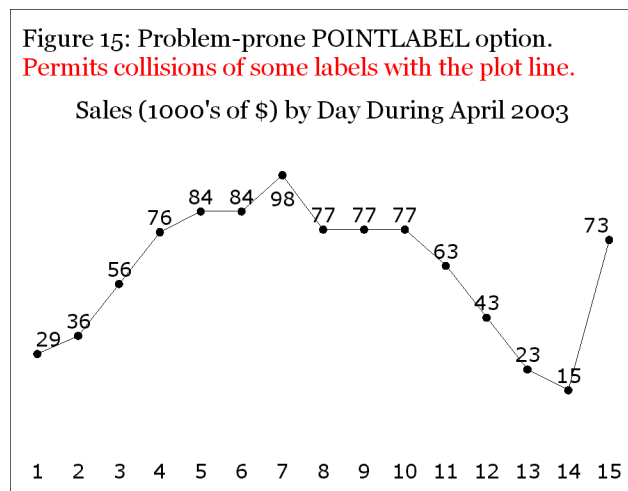
Figure 14: Unambiguously identifies x-values, and uses tiny dots for most precise y-value estimate.



To create Figure 14, use these replacements in the code for Figure 13:

```
symbol1 v=point i=needle;
title2 h=1.00 f='Georgia' j=L
' Figure 14: Unambiguously identifies x-values,' j=L
' and uses tiny dots for most precise y-value estimate.';
```

SAS/GRAPH does provide an option to perform automatic annotation of any plot. It is the POINTLABEL option of the SYMBOL statement. Though it makes an effort to succeed, POINTLABEL is not always reliable, as can be seen in Figure 15. Since the y-values are annotated, there is no need for a y-axis. This data is designed to exercise all 13 possible three-point / two-segment transitions in trend line slope.



Here is the code used to create unacceptable Figure 15:

```
title2 h=1.00 f='Georgia' j=L
' Figure 15: Problem-prone POINTLABEL option.' j=L c=CXFF0000
' Permits collisions of some labels with the plot line.';
title3 h=0.50 f=none '';
title4 h=1.00 f='Georgia' j=C
'Sales (1000's of $) by Day During April 2003';
footnote;
axis1 label=none major=none minor=none style=0 value=(h=0.85);
axis2 label=none major=none minor=none style=0 value=none;
symbol1 v=dot h=0.50 i=join pointlabel=(h=0.85);
goptions vpos=17 vsize=2.55 IN ymax=2.55 IN ypixels=765;
goptions hpos=34 hsize=3.25 IN xmax=3.25 IN xpixels=975;
proc gplot data=DataForTrendPlot(where=(YearMonth='200304'));
plot Sales*DayOfMonth /
    haxis=axis1 vaxis=axis2 vzzero;
run; quit;
```

The problem in Figure 15 has been solved. The solution is outside the scope of this Beginning Tutorial, and uses a custom macro. If you are interested in the macro, send me an email. If you wish to build your own macro, consult Reference 2.

Figure 16 provides the visual guidance of a trend line for quick assessment, supplemented with the precision of an on-graph table built into tick mark labels. A graph is a visual aid to accelerate decision-making and inferences. A table of precise values (or annotation of them on the graph) is a necessity for reliable decision-making and inferences. Since all y-values are in the table, there is no need for a y-axis.

Here is the code used to create Figure 16:

```
%macro maketks;
%do i = 1 %to &ValCount;
tick=&i "&&x_value&i" j=C c=CXFF0000 "&&y_value&i"
%end;
%mend maketks;
```

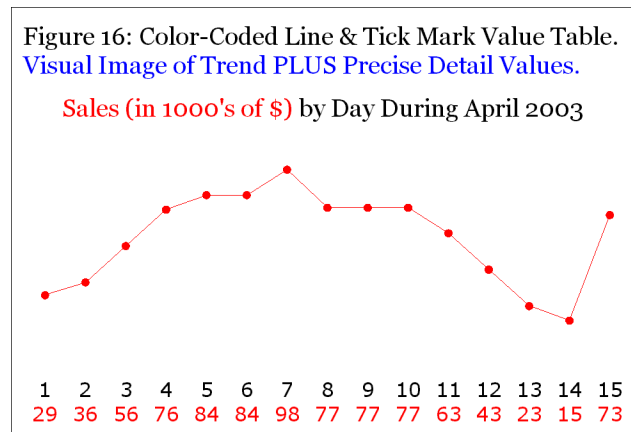
```
data _null_;
set DataForTrendPlot(where=(YearMonth='200304')) end=lastobs;
call symput('y_value'||trim(left(_N_)),trim(left(Sales)));
call symput('x_value'||trim(left(_N_)),trim(left(DayOfMonth)));
if lastobs;
call symput('ValCount',_N_);
run;
```

```
title2 h=1.00 f='Georgia' j=L
' Figure 16: Color-Coded Line & Tick Mark Value Table.' j=L
c=CX0000FF
' Visual Image of Trend PLUS Precise Detail Values.';
title3 h=0.50 f=none '';
title4 h=1.00 f='Georgia' j=C
c=CXFF0000 'Sales (in 1000's of $)'
c=CX000000 'by Day During April 2003';
footnote;
axis1 label=none major=none minor=none style=0
    value=(h=0.85 %maketks);
axis2 label=none major=none minor=none style=0 value=none;
symbol1 v=dot h=0.50 i=join c=CXFF0000;
goptions vpos=15 vsize=2.25 IN ymax=2.25 IN ypixels=675;
goptions hpos=34 hsize=3.25 IN xmax=3.25 IN xpixels=975;
proc gplot data=DataForTrendPlot(where=(YearMonth='200304'));
plot Sales*DayOfMonth /
    haxis=axis1 vaxis=axis2 vzzero;
run; quit;
```

The keys to this solution are: (a) the preprocessing to load Sales and DayOfMonth values as macro variables into the global symbol table; (b) the retrieval of those values as tick mark values in the AXIS1 statement with the %MAKETKS macro; and (c) color-coding the title, the plot dots and line, and the tick mark table entries. If you use OPTIONS MPRINT, and inspect the SAS log, you will find that the effect of the %MAKETKS macro is to produce, at run time, the following AXIS1 statement:

```
axis1 label=none major=none minor=none style=0
    value=(h=0.50
    tick=1 "1" j=C c=CXFF0000 "29"
    tick=2 "2" j=C c=CXFF0000 "36"
    tick=3 "3" j=C c=CXFF0000 "56"
    ...
    tick=15 "15" j=C "73");
```

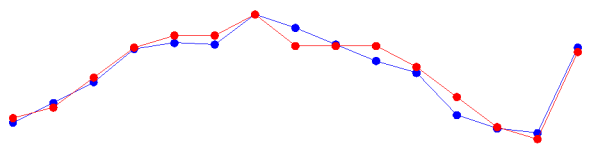
Use of the SAS Macro facility in coding this SAS/GRAPH application is an example of what I call using Software Intelligence. It dynamically creates an axis definition customized to the data found at run time.



Now we can take the method used for Figure 16 one step further. We can handle the problem of comparing the trend for the current reporting period trend with that for the prior reporting period. See Figure 17. This is an instance of the general multi-line multi-class plot, defined by code PLOT Y\*X=Z, where Z is the class variable. Here, Z is Year. Since all the y-values are listed in the table, there is no need for a y-axis.

Figure 17: Color-Coded Trends & Tick Mark Value Table.  
Multi-Line Multi-Class Plot.

Sales (in 1000's of \$) by Day During April 2003  
and Tabular Comparison with Previous Month



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
29	36	56	76	84	84	98	77	77	77	63	43	23	15	73
26	39	53	75	79	78	98	89	78	67	59	31	22	19	76

Day of Month  
Sales During Report Month  
Sales During Previous Month

Here is the code used to create Figure 17:

```
%macro maketks2;
%do i = 1 %to &ValCount;
tick=&I c=CX000000 "&&x_value&i" j=C'---' j=C c=CXFF0000
"&&y_value&i" j=C c=CX0000FF "&&y_prev&i"
%end;
%mend maketks2;

data _null_;
set DataForTrendPlot(where=(YearMonth eq '200304'))
end=lastobs;
call symput('y_value'||trim(left(_N_)),trim(left(Sales)));
call symput('x_value'||trim(left(_N_)),trim(left(DayOfMonth)));
if lastobs;
call symput('ValCount',_N_);
run;

data _null_;
set DataForTrendPlot(where=(YearMonth eq '200303'));
call symput('y_prev'||trim(left(_N_)),trim(left(Sales)));
run;

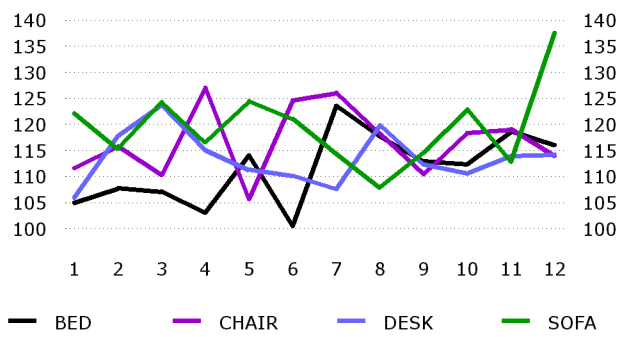
title2 h=1.00 f='Georgia' j=L
' Figure 17: Color-Coded Trends & Tick Mark Value Table.' j=L
c=CX0000FF
' Multi-Line Multi-Class Plot.';
title3 h=0.50 f=none '';
title4 h=1.00 f='Georgia' j=C
'Sales (in 1000"s of $) by Day During April 2003' j=C
'and Tabular Comparison with Previous Month';
footnote;
axis1 label=(h=0.80
j=L c=CX000000 'Day of Month'
j=L c=CXFF0000 'Sales During Report Month'
j=L c=CX0000FF 'Sales During Previous Month')
major=none minor=none style=0 value=(h=0.85 %maketks2);
axis2 label=none major=none minor=none style=0 value=none;
symbol1 v=dot h=0.50 i=join c=CX0000FF;
symbol2 v=dot h=0.50 i=join c=CXFF0000;
goptions vpos=20 vsize=3.00 IN ymax=3.00 IN ypixels=900;
goptions hpos=34 hsize=3.25 IN xmax=3.25 IN xpixels=975;
proc gplot data=DataForTrendPlot;
plot Sales*DayOfMonth=YearMonth /
haxis=axis1 vaxis=axis2 vzero
nolegend; /* suppress the automatic default legend */
run; quit;
```

SAS/GRAPH does not make it easy to get those descriptors for the lines of tick mark values at, say, the left end of their respective print lines. Hence, I used the less than ideal solution of a multi-line axis label. The SAS/GRAPH Annotate Facility is outside the scope of a Beginning Tutorial. My search for an Annotate-free solution is underway.

When you have too many lines in a trend plot, it is infeasible to create a table of tick mark values. You must resort to visual estimates and the reference lines of Figure 13. And you must add a legend. In this case, it is important to be able to spread out the lines as much as possible. The y-axis is not forced to start at zero, but instead usability of the vertical space is maximized to facilitate estimating the y-values.

Figure 18: Multi-Line Multi-Class Plot With Legend.  
Too many lines for tick mark value table to be feasible.

Sales by Product Category by Month During 1998



Here is the code used to create Figure 18:

```
title2 h=1.00 f='Georgia' j=L
' Figure 18: Multi-Line Multi-Class Plot With Legend.' j=L
' Too many lines for tick mark value table to be feasible.';
title3 h=0.50 f=none '';
title4 h=1.00 f='Georgia' j=C
'Sales by Product Category by Month During 1998';
footnote;
axis1 label=none major=none minor=none style=0 value=(h=0.75);
axis2 label=none major=(c=CXFFFFFF) /* make major ticks invisible */
minor=none style=0 value=(h=0.75) order=100 to 140 by 5;
axis3 label=none major=(c=CXFFFFFF) /* make major ticks invisible */
minor=none style=0 value=(h=0.75 j=L) order=100 to 140 by 5;
symbol1 v=none i=join w=6 c=CX000000; /* w=6 thickens the lines */
symbol2 v=none i=join w=6 c=CX9900CC;
symbol3 v=none i=join w=6 c=CX6666FF;
symbol4 v=none i=join w=6 c=CX009900;
symbol5 v=none i=none r=4; /* no symbols/lines for PLOT2 */
legend1 label=none value=(h=0.75) shape=line(1.5);
goptions vpos=17 vsize=2.55 IN ymax=2.55 IN ypixels=765;
goptions hpos=34 hsize=3.25 IN xmax=3.25 IN xpixels=975;
proc gplot data=DataForFourLineTrendPlot;
plot Sales*Month=Product /
haxis=axis1 vaxis=axis2
legend=legend1 /* legend1 statement defines the legend */
autovref lvref=33;
plot2 Sales*Month=Product /* PLOT2 to get right-hand vertical axis */
vaxis=axis3
nolegend;
run; quit;
```

The SYMBOL5 statement prevents drawing visible lines for the PLOT2 statement, for which the only function is production of the right-hand-side vertical axis. The other SYMBOL statements define the line colors. Because it can be difficult to distinguish colors of thinly drawn lines, they are thickened by use of W=6 in the SYMBOL statements. SYMBOL statements are always applied in ascending sort sequence order of the values of the CLASS variable, i.e., the values of Z in code PLOT Y\*X=Z. Here, Z is Product.

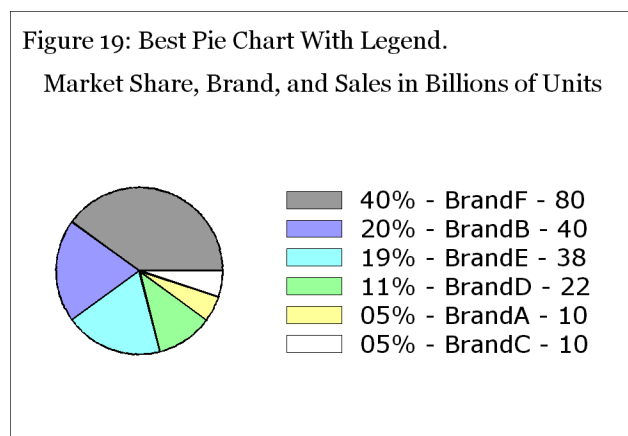
It is also possible to create an overlay plot of two different y variables, with different y-axes. However, code for the design that I recommend is too lengthy to print here. During the stand-up presentation, I display the result. To request the code, you may send me an email.

Figure 19 is the most problem-resistant pie chart you can build with SAS/GRAPH—if you wisely insist on code that can simply be pointed at the data, and that no post-processing or special manipulation or circumvention be required.

This legend is custom-built. The Software-Intelligent code dynamically orders the legend entries and the pie slices from largest to smallest. “Show them what’s important.” The color list is designed to go from darkest to lightest, so that the smaller pie slices are more visible. The legend normally obtainable from SAS/GRAPH only includes the pie slice name labels, not PERCENT and VALUE.

If you permit SAS/GRAPH to place the PERCENT and VALUE around the pie perimeter, there is no guarantee that overlaps will be avoided. The only way that the solution in Figure 19 can “fail” is if you have pie slices that are too small for their wedges and area fill to be visible. However, even in that situation, you have the all-inclusive ranked table which supplies name, percent, and value for every pie slice, including the invisible slices.

This solution is more complicated than I like to present in a Beginning Tutorial, but it is too important and too widely applicable to omit. Those pie label overlap problems occur too often, and have no better solution.



Here is the code used to create Figure 20:

```
%macro dopatrn;
%do I = 1 %to 6;
pattern&i v=psolid c=&&SliceColor&i r=1;
%end;
%mend dopatrn;

%macro getlegnd;
%do I = 1 %to 6;
"&&legentry&i"
%end;
%mend getlegnd;

data ColorList;
pctseq = 1; color='CX999999'; output;
pctseq = 2; color='CX9999FF'; output;
pctseq = 3; color='CX99FFFF'; output;
pctseq = 4; color='CX99FF99'; output;
pctseq = 5; color='CXFFFF99'; output;
pctseq = 6; color='CXFFFFFF'; output;
run;

proc sort data=SliceNameWithPercentAndValue;
by descending Percent;
run;
```

```
data ToSort;
set SliceNameWithPercentAndValue;
pctseq = _N_;
call symput('legentry'||trim(left(_N_)),
            trim(left(NameWithPercentAndValue)));
run;

proc sort data=ToSort; by pctseq; run;

data SliceWithColor(drop=pctseq);
merge ToSort ColorList;
by pctseq;
run;

proc sort data=SliceWithColor; by NameWithPercentAndValue; run;

data _null_;
set SliceWithColor;
call symput('SliceColor'||trim(left(_N_)),trim(left(color)));
run;

title2 h=1.00 f='Georgia' j=L
' Figure 19: Best Pie Chart With Legend.';
title3 h=0.50 f=none '';
title4 h=1.00 f='Georgia' j=C
'Market Share, Brand, and Sales in Billions of Units';
footnote a=+90 h=0.10 IN '';
%do patrn;
legend1 label=none
shape=bar(3,0.6)
order=(%getlegnd)
across=1
position=(middle right outside);
goptions vpos=15 vsize=2.25 IN ymax=2.25 IN ypixels=675;
goptions hpos=34 hsize=3.25 IN xmax=3.25 IN xpixels=975;
proc gchart data=SliceNameWithPercentAndValue;
pie NameWithPercentAndValue /
sumvar=Value noheading descending
outline=CX000000 woutline=2
legend=legend1
slice=none value=none percent=none; /* turn off all pie labels */
run; quit;
```

The same data set, SliceNameWithPercentAndValue, which was prepared for Figure 4, is used here. The FOOTNOTE statement with the strange parameter A=+90 (90 degrees counterclockwise) forces extra blank space at the right-hand margin, pushing the legend to the left.

#### Input Data

Below is the code used to create the input. The values for the observations not listed can be read off the graphs created with them.

```
data DataForSimpleCharts;
infile cards;
input @1 Name $6. @8 Value 2.;
cards;
BrandA 10
...
; run;
data UnitsSoldByBrandAndYear;
infile cards;
input @1 Year $4. @6 Brand $6. @13 Value 2.;
cards;
2001 BrandF 76
...
; run;
data UnitsSoldByBrandAndMarket;
infile cards;
input @1 Brand $6. @8 Market $8. @17 Value 2.;
cards;
BrandF Domestic 10
...
; run;
```



```

data DataForTrendPlot;
infile cards;
input @1 DayOfMonth 2. @5 Sales 2. @8 YearMonth $6.;
cards;
1 29 200304
...
; run;
data DataForFourLineTrendPlot(drop=Date Year Actual);
set sashelp.prdsal3
  (where=(Year=1998) keep=Year Product Actual Date);
Sales = Actual / 1000;
Month = month(Date);
run;
proc summary nway data=DataForFourLineTrendPlot;
class Product Month;
var Sales;
output out=DataForFourLineTrendPlot(drop=_type__freq_)
sum=Sales;
run;

```

### Common Preliminary Code for All the Graphs

```

proc catalog c=work.gseg kill;run;quit; /* clean out graph catalog */
goptions reset=all; /* it is best to do this reset before every graph */
goptions cback=CXFFFFFF; /* background color RGB white */
goptions htext=1.00 ftext='Verdana'; /* height and font used for parts
of graph for which you do not make an explicit assignment, or for which
no direct controls are available in SAS/GRAPH. With the exception of
the last-mentioned situation, you can override htext= and ftext= in
various graphic PROC Step statements with h= and f=. */
goptions border; /* put the graph in a box—to separate it from text
when being published in a document by import with MS Word */
title1 h=0.50 f=none ' '; /* empty space between border and TITLE2 */

```

### Using and Customizing Graphic Device Drivers

```

goptions device=PNG (or GIF); /* specifies the device driver */
goptions gsfname=YourFileRef; /* for output .png or .gif file */
filename YourFileRef 'c:\YourFileName.ext'; /* .ext is .png or .gif */

goptions HPOS=      number of columns in the graphic area.
goptions VPOS=      number of rows in the graphic area.
goptions HSIZE=     horizontal size of the graphic area.
goptions VSIZE=     vertical size of the graphic area.
goptions XMAX=      maximum value for HSIZE.
goptions YMAX=      maximum value for VSIZE.
goptions XPIXELS=   maximum number of pixels in the XMAX space.
goptions YPIXELS=   maximum number of pixels in the YMAX space.

```

#### PNG driver defaults are:

```

hpos: 76  xmax: 6.474 IN  hsize: 6.474 IN  xpixels: 615
vpos: 43  ymax: 3.631 IN  vsize: 3.631 IN  ypixels: 345

```

#### GIF driver defaults are:

```

hpos: 88  xmax: 8.420 IN  hsize: 8.420 IN  xpixels: 800
vpos: 43  ymax: 6.310 IN  vsize: 6.310 IN  ypixels: 600

```

The above default values are overridden for the graphs created here. PNG image files are bigger than GIF image files for the same graph, but can produce smoother contours and better text. For graphs to be imbedded in a document, you can, e.g., set maximum PNG pixel counts to 300 times the maximum size in inches—as was done in all the examples in this paper. However, if the image quality is adequate, you may prefer the GIF driver for web publishing, where quicker downloads (web page displays) result from smaller file sizes.

These drivers have default color lists. I have only used Browser-Safe RGB colors with codes of the form CXrrggbb, where rr, gg, and bb are from the list 00, 33, 66, 99, CC, FF. Browser-Safe colors are a set of 216 colors recommended for web use. For more about image files and color, please see Reference 1.

### Glossary of SAS/GRAPH Statements and Options

**TITLE and FOOTNOTE statements.** (Up to 10 of each per graph.)  
**F=** means FONT=. 'Georgia' and 'Verdana' are enclosed in single quotes because they are Windows fonts, rather than SAS/GRAPH fonts.  
**H=** means HEIGHT=. With nothing after the numeric assignment, the default unit is CELLS. It could instead be IN (inches), CM (centimeters), PT (points), or PCT (percent of graphic display area). The entire graph composition area is divided into cells, whose size is determined from default or custom-specified GOPTIONS parameters (see discussion above).  
**J=** means JUSTIFICATION=. Possible values are C (center), L (left), and R (right).  
**Text** must be enclosed in quotes. If the text contains any macro variables, it must be enclosed in double quotes. The text may be specified in multiple quoted strings, which will simply be concatenated. However, if you insert a J= assignment between two strings, the second string is displayed on a new line. This same thing can be done for the text strings in LABEL= assignments and in tick mark values. The statement "FOOTNOTE;" is used to turn off any FOOTNOTE statements that may have been defined for a prior graph during the same SAS session.

### AXIS statements.

**LABEL=** ('some text', with optional use of F=, H=, J=) to label the axis. For some examples, LABEL=NONE.  
**MAJOR=NONE** suppresses printing of major tick marks.  
**MINOR=NONE** suppresses printing of minor tick marks.  
**STYLE=0** suppresses printing of the axis line.  
**VALUE=**('tickmark1 text' 'tickmark2 text' . . . , with optional use of F=, H=, J=) to customize the labels of the major tick marks (even if the tick marks themselves are not displayed).  
For examples where vertical axis tick mark values are superfluous (e.g., on a bar chart with bar ends labeled with the exact values of the y variable), VALUE=NONE is used here.  
In some examples, the tick mark text is specified in parts, with J= separating them, to create stacked tick mark labels. In other examples, the values automatically supplied by SAS/GRAPH are controlled only as to height.  
**ORDER=**starting value to ending value by increment value, specifies the range bounds for the axis, and spacing of intermediate major tick marks.  
**OFFSET=** can be used to put some space between where the axis would naturally begin and where it actually begins. It is useful, e.g., to offset the y-axis if you have plot points closer to the x-axis than you prefer.

### SYMBOL statements.

**V=** specifies the plot symbol placed at each data point. The examples use either DOT or NONE.  
**I=** specifies interpolation between plot points; here we usually use JOIN or NONE. For I=NEEDLE, see the effect in Figure 14.  
**H=** specifies the size of the plot symbol.  
**C=** specifies the color of the plot symbol and, if I=JOIN, the color of the plot line.

### PATTERN statements.

**V=** specifies the type of area fill. The examples use either EMPTY (for empty pie slices) or SOLID (for solid bars).  
**C=** specifies the color of the area fill.  
**R=** specifies for how many areas the assigned V= and C= should be repeated.

### LEGEND statements.

**LABEL=** ('some text', with optional use of F=, H=, J=) to label the legend. For some examples, LABEL=NONE.  
**SHAPE=BAR(x,y)** or **LINE(x)** specifies the length and height of the rectangular area fill samples or the length of the plot line samples.  
**ORDER=** specifies the order of the legend entries.  
**ACROSS=** specifies the number of columns into which the legend entries are arranged.  
**POSITION=** specifies the location of the legend in the graph display area.  
**VALUE=** ('text1' 'text2' . . . , with optional use of F=, H=, J=) to customize the text of the legend entries.

**PLOT statements and options.**

PLOT Y\*X specifies that the vertical axis variable is Y and horizontal axis variable is X.

PLOT Y\*X=Z specifies that there is to be a separate plot drawn for every value of classification variable Z.

HAXIS=AXISp and VAXIS=AXISq specify that statements AXISp and AXISq define the horizontal and vertical axis characteristics.

VZERO specifies that the vertical axis must start with 0.

AUTOVREF specifies that a horizontal reference line be drawn at every major tick mark on the vertical axis, even if the tick mark is not drawn.

LVREF specifies the line type used to draw the VREFs.

LEGEND=LEGEND1 specifies that statement LEGEND1 is to be used for the multi-line plot.

NOLEGEND specifies that no LEGEND be created.

PLOT2 Y\*X=1 specifies that a second vertical axis must be drawn at the right-hand side, and that statement AXIS1 defines the right-hand axis characteristics.

**Options Common to HBAR and VBAR Statements.**

HBAR/VBAR Name specifies that the variable Name contains the values to label the bars.

SUMVAR=Value specifies that all values of Value for the same value of Name should be summed to determine the measurement (the so-called “response”) for that value of Name (the so-called “midpoint”). This is also used when each Name value occurs only once.

WIDTH= specifies the non-default width of the bars.

SPACE= specifies the non-default space between the bars.

MAXIS=AXISp specifies that the AXISp statement is to be used for the axis for the Name values (the so-called “midpoints”).

RAXIS=AXISq specifies that the AXISq statement is to be used for the axis for the Value values (the so-called “responses”).

DESCENDING specifies that the bars be arranged in descending order of length (HBAR) or height (VBAR).

**Options used here with HBAR Statements.**

FREQ=Value specifies that all values of Value for the same value of Name should be summed to determine the charted measurement for that value of Name. This can be also used when each Name value occurs only once. This sum is regarded as a “frequency of response”, but it really can be a magnitude, like SUMVAR=.

FREQ specifies that a column of FREQ values be listed at the right margin of the bar chart.

FREQLABEL= specifies the heading for that column of FREQ values.

SUMLABEL= specifies the heading for the column of SUMVAR values at the right margin of the bar chart.

**Options used here with VBAR Statements.**

(These CAN also be used with HBAR.)

GROUP= specifies that this third variable (besides what is assigned as Name and Value) will group sets of midpoint values (i.e., sets of what I call the Name variable).

GAXIS=AXIS3 specifies that the AXIS3 statement defines the characteristics of the “axis” for GROUP variable. This “axis” is below (or to left of) the midpoint axis for VBAR (or HBAR).

AXIS3 . . . NOBRACKETS removes unneeded brackets to span each group of bars.

PATTERNID=MIDPOINT specifies that the PATTERN statements are to be used to distinguish bars *within* a group.

**PIE statements and options.**

PIE Name specifies that the variable Name contains the strings to label the slices.

SUMVAR=Value specifies that all values of Value for the same value of Name should be summed to determine the measurement (or “response”) for that value of Name. Also used if each Name value occurs only once.

NOHEADING specifies that the default pie chart heading be omitted (it’s neither elegant, nor necessary—every pie chart deserves a TITLE).

COUTLINE=CX000000 specifies that pie slices be outlined in Black.

WOUTLINE= specifies the width of pie slice outlines.

PERCENT=OUTSIDE specifies that each SAS/GRAPH-supplied Percent of Whole be displayed outside its respective slice.

VALUE=NONE specifies that values of Value be omitted.

SLICE=ARROW specifies that each value of Name be displayed outside its respective slice, with an “arrow” connecting it to the slice.

**Web-Publishing Your Graphs** (See also References 1-4)

To put your graph in a web page, use the GIF (or, if download time is not a concern, the PNG) driver, and wrap your graph code as follows:

```
ods listing close;
ods noresults;
ods html path='c:\YourFolderName' (url=None)
      gtitle gfootnote
      body='YourWebPageName.html'
      style=styles.YourWellDesignedStyle;
YOUR GRAPH CODE GOES HERE
ods html close;
ods listing;
```

In this case, you omit the gsfname goption, and the filename for the GIF or PNG file. To control the name of the GIF or PNG file that is stored in the PATH= folder, put NAME='XXXXXXXX' after the / in the PIE, HBAR, VBAR, or PLOT statement (or CHORO statement, in the case of a map), where XXXXXXXX is a unique 8-character name.

For web design and methods, see References 1 and 3.

For the best compendium of well-designed web-enabled graphs, maps, and tables built with ODS, SAS, and SAS/GRAPH, see Reference 4.

If contemplating using an ODS Table of Contents, code for a well-designed TOC can be found in Reference 2 or 4.

**Acknowledgement**

My thanks to Dawn Schrader for educating me as to the benefit and use of PNG image files for MS Word publication. This is my first paper to use PNG images for the illustrations, and I am pleased with the results.

**Related Work by the Author**

1. “The Power of Pictures and Paint: Using Image Files and Color with ODS, SAS, and SAS/GRAPH”, elsewhere in these SUGI 28 Proceedings.
2. With Francesca Pierri, “%TREND: A Macro to Produce Maximally Informative Trend Charts with SAS/GRAPH, SAS, and ODS for the Web or Hardcopy”, *Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference*, SAS Institute (Cary, NC., 2002.
3. “Web Communication Effectiveness: Design and Methods to Get the Best Out of ODS, SAS, and SAS/GRAPH”, in SUGI 28 Proceedings.
4. With Francesca Pierri, “Show Your Graphs and Tables at Their Best on the Web with ODS”, *Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference*, SAS Institute (Cary, NC), 2002.

**Notices**

SAS/GRAPH and SAS are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other product and brand names are trademarks or registered trademarks of their respective owners.

**Author Information**

LeRoy Bessler PhD  
Bessler Consulting and Research  
PO Box 96, Milwaukee, WI 53201-0096, USA  
Phone: 1 414 351 6748  
Email: bessler@execpc.com

LeRoy Bessler does mentoring, general SAS application development, and communication-effective design and construction of reports, tables, graphs, and maps for the web and other media. He has special expertise in Software-Intelligent application development, which yields SAS solutions that are reliable, reusable, maintainable, and extendable. Dr. Bessler has 25 years of experience using SAS for various industries, on MVS, Windows, Unix, and VM.

**The Power to Show**