Paper 220-2009

Show Them What's Important with Communication-Effective SAS® Graphs: Solutions for a Finite Work Day in an Era of Information Overload

LeRoy Bessler, Assurant Health, Milwaukee, Wisconsin, USA

Abstract

In this era of information overload and incessant instant information update, what are the best ways to visually and memorably communicate, and to meet the six crucial criteria of concise, clear, complete, correct, convincing, and current? This paper uses illustrations, design guidelines, and code to answer that question. Communication-effective SAS graphs visually say the most with the least, to focus on what matters. Solutions are provided for production applications, i.e., for overnight or on-demand batch jobs or for real-time SAS/IntrNet® response, where the graph has to be right the first time every time, with no opportunity for iterative refinement that might be tolerable for ad hoc presentation preparation. Ways that Version 9.2 makes results more communication-effective are emphasized.

Introduction

For over twenty years, I have advocated for minimalist graphic design to focus on the message, not the medium. I have demonstrated the power of simplicity by stripping out traditional graphic presentation paraphernalia to make communication-effective use of SAS and SAS/GRAPH®. This paper shows you how to provide not only visual indication of size or trend for quick and easy inference, but also readable precise numbers for reliable inference. Also, I have promoted the use of graphic creation techniques that are robust enough to not require hands-on intervention from run to run. A production application does require iterative modification during its development, but once built for the "worst" case thereafter it must perform successfully without run-specific intervention.

In Reference 1 you can find communication-effective design for the small screen with image and sufficient detail, and the SAS/GRAPH code needed, for popular management reporting formats. Many of the design principles presented in that paper are applicable here, and the design principles presented here are applicable to the small screen. For a broader and deeper discussion of graphic design than I will provide now, and not restricted to the small screen, please see Reference 2.

This paper is about management graphics, not statistical, analytical, or scientific graphics. It presents illustrations, design, and code, and relies on three principles:

1. Show them what's important
2. Where it will not impair communication, let part stand for the whole
3. Assure that all text and numbers are readable

When applicable, one way to implement Principle 1 is with ordering (if bigger is better, ordering from largest to smallest). Principle 2 supports Principle 1. Principle 2 might sound as if it defeats the objective of delivering information that is complete. However, if everything NEEDED is delivered, then the information IS complete. In a situation where you need a comprehensive reference/lookup tool, then every relevant element must be included. For such a resource, a table is eminently reliable to use and trivially simple to create. But when you are trying to easily and quickly draw a business inference, it is not necessarily the case that every piece of information is significant. If you want to facilitate and accelerate the inferential process, presenting everything runs the risk of not only slowing down interpretation and inference, but also potentially obscuring, or diminishing the visual impact of, the important. Ways shown here to implement Principle 3 are: (a) avoiding the common regrettably common practice of using vertical labels to get around horizontal space limitations; (b) avoiding the annotation collisions that are possible for line charts that use the POINTLABEL option on the SYMBOL statement; and (c) avoiding the label overlap problems common for pie charts when you display slice name, value, and percent of whole.

NOTE: All figures are shrunken to meet the Proceedings page count limit for contributed papers. Full-size figures and all code (most of which is not given here) are available in a zip file via email from the author. This paper uses Version 9.2, but the zip file also includes Version 9.1.3 figures and code. Figures 1, 3, 5, 17, and 18 require code changes based on SAS/GRAPH version, but the changes are minor. The PNG driver was vastly improved for 9.2, and is used here for Figures 1-16, but the ZGIF driver is required for Figures 17 & 18. In 9.1.3, the GIF driver must be used throughout. In 9.1.3, the PNG driver has an awkward aspect ratio and does not support transparency. In 9.2, fonts are better rendered than in 9.1.3. For more on 9.2 vs. 9.1.3, please see Reference 3.

Bar Charts

Vertical bar charts, though widely used, suffer space constraints for the horizontal axis tick mark value labels. Horizontal bar charts allow you to present a table with a visual aid. Figures 1-3 use ranking and subsetting.

Figure 1: Top 10

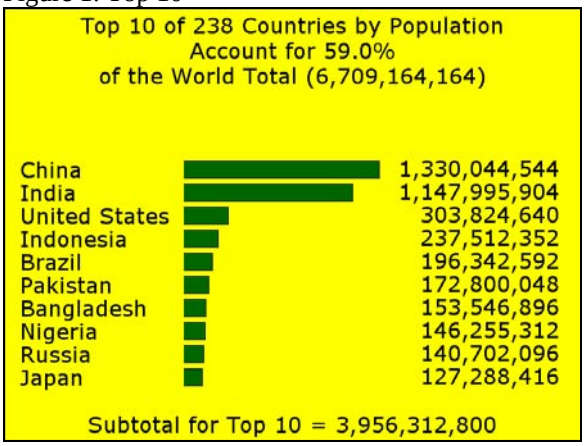


Figure 2: Top 20

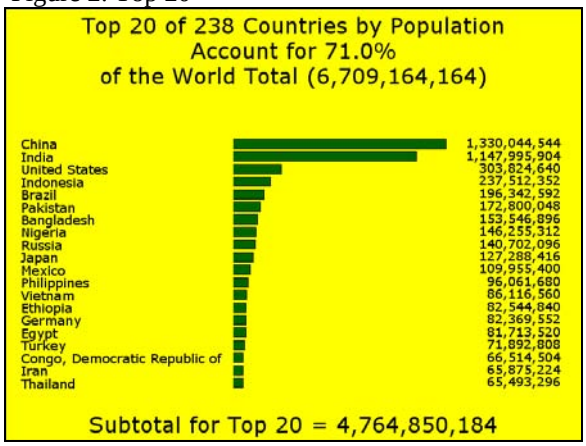


Figure 3 uses a different method of subsetting. Without subsetting, the design can be a pie chart alternative for the case where there would be too many slices. All three figures notify the viewer as to how much has been left out.

Figure 3: Top 80%

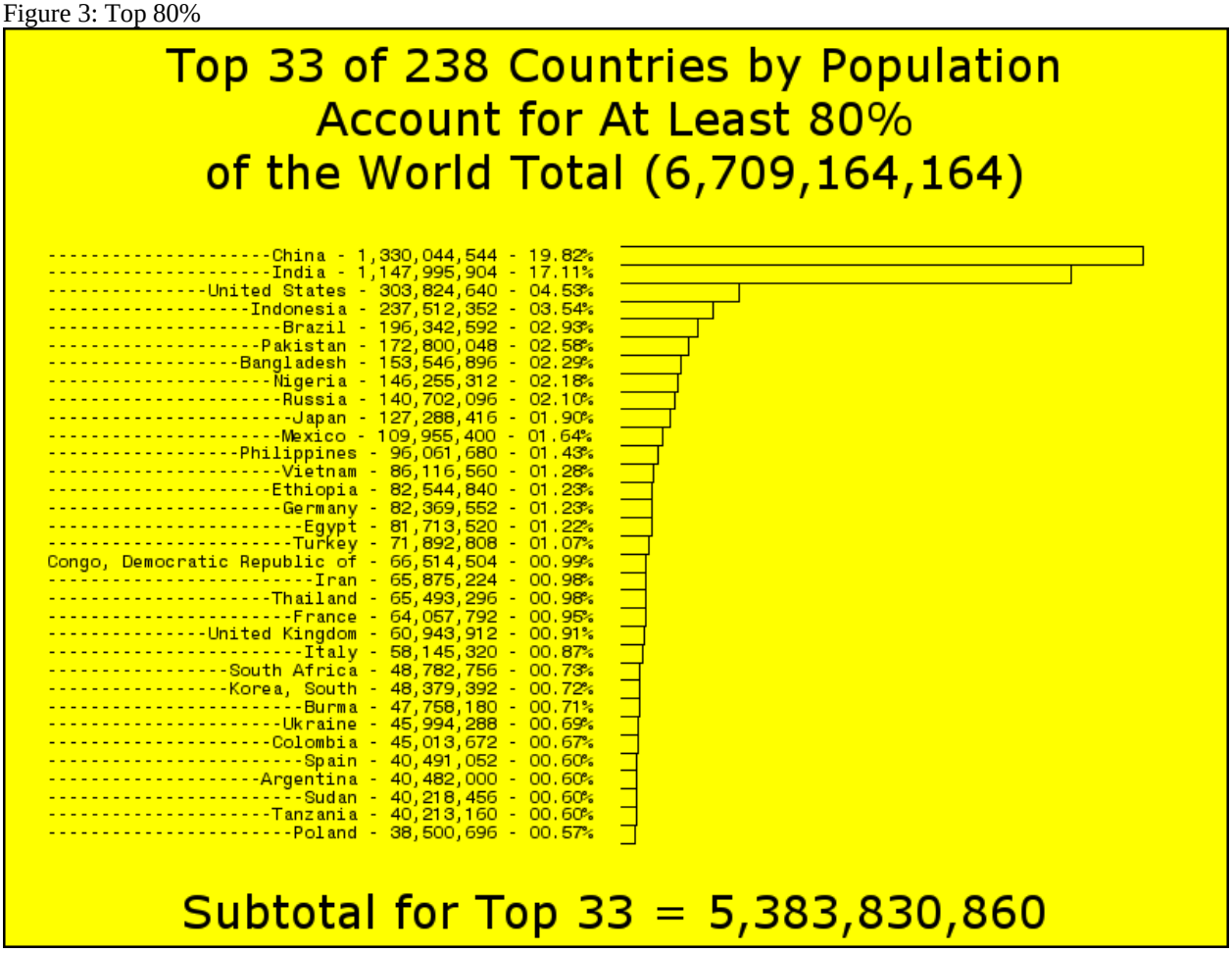


Figure 4: 3D with Needless Complexity

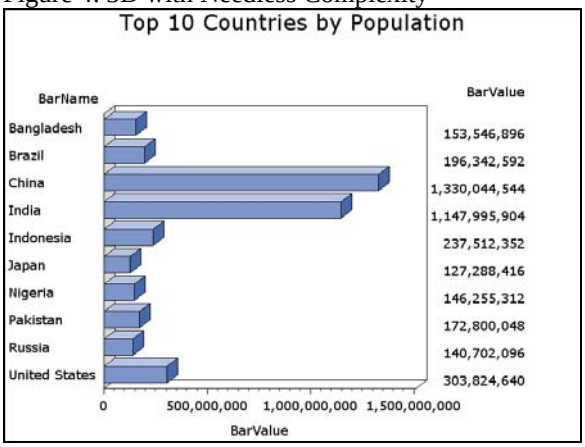
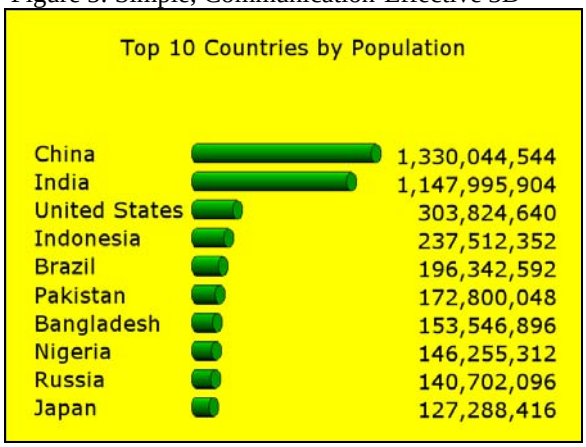


Figure 5: Simple, Communication-Effective 3D



If your manager, your client, or you require 3D, the cylinders without a frame in Figure 5 should suffice.

Version 9.2 Notes:
Bar label text now can be as long as 256 characters. The old limit of 32 characters would have been too small for the bar chart of the Top 80 Percent.
In Version 9.1.3, the COUTLINE option had no effect for SHAPE=CYLINDER. In Version 9.2, the COUTLINE option now works as expected. However, since the 3D effect for cylinders is achieved by painting the wrapping wall of the cylinder with stripes of the chosen color, progressing from the true color to its darkest shade, which is actually black, if you use a COUTLINE (Outline Color) other than black, the result is an undesirable visual effect.
In Version 9.1.3, the walls of a 3D bar chart can be removed only by putting the NOPLANE option on the vertical and horizontal AXIS statements, whereas in Version 9.2 it can be put on the HBAR3D statement directly.

Here is the code to create Figure 1:

```
data work.Input;
infile datalines; input @1 BarName $13. @15 BarValue 10.;
datalines;
Bangladesh    153546896
Brazil        196342592
China         1330044544
India         1147995904
Indonesia     237512352
Japan         127288416
Nigeria       146255312
Pakistan      172800048
United States 303824640
Russia        140702096
etc.
; run;

proc means data=work.Input noprint sum n;
var BarValue;
output out=work.Summary sum=Total n=Count;
run;

data _null_;
set work.Summary;
call symput('Total',Total);
call symput('TotalForDisplay',trim(left(put(Total,comma13.))));
call symput('CountForDisplay',trim(left(put(Count,3.))));
run;

proc sort data=work.Input; by descending BarValue; run;

%let MaxN = 10; /* Select the Top 10 */
%let PercentOfPicHeightForBars = 65;
```

```
data work.Top10(keep=BarName BarValue);
retain TotToRpt 0;
set work.Input;
output;
TotToRpt = TotToRpt + BarValue;
if _N_ = &MaxN;
PctOfTotal = (TotToRpt / &Total) * 100;
call symput('N_TotToRptForDisplay',trim(left(put(TotToRpt,comma13.))));
call symput('MaxNpctOfTotal',trim(left(put(PctOfTotal,5.1))));
BarLabelFontSizeAsPct = min(5,floor(&PercentOfPicHeightForBars / _N_));
call symput('BarLabelFontSizeAsPct',BarLabelFontSizeAsPct);
stop; run;

%let TopBottomSpace = 1PCT;
%let FontSize = 5PCT;
%let Font = Verdana;
goptions reset=all;
goptions device=PNG;
goptions border; /* Put the graph in a box. */
goptions cback=CXFFFF00; /* Use RGB Yellow for background. */
goptions htext=&FontSize ftext="&Font";
 /* Height & font for those parts of the graph which you do not explicitly assign,
    or for which no controls are available */
goptions gsfmode=replace gsfname=anyname;
footnote1 angle=+90 font=none height=1 ' ';
footnote2 angle=-90 font=none height=1 ' ';
 /* blank "side-notes" for space on each side of the image. */
pattern1 v=solid color= CX006600; /* RGB Medium Dark Green for bars */
 /* Remove all axis paraphernalia: */
axis1 label=none major=none minor=none style=0;
axis2 label=none major=none minor=none style=0 value=none offset=(1 PCT,1 PCT);
                      /* Offset to insert space bar ends and bar-related text */

title1 font="&Font" color=CX000000 height=&TopBottomSpace ' ' height=&FontSize
justify=CENTER
  "Top &MaxN of &CountForDisplay Countries by Population"
  justify=CENTER "Account for &MaxNpctOfTotal.%"
  justify=CENTER "of the World Total (&TotalForDisplay)";
footnote3 font="&Font" color=CX000000 height=&FontSize
  justify=CENTER "Subtotal for Top &MaxN = &N_TotToRptForDisplay";
footnote4 font="&Font" height=&TopBottomSpace ' ';

options symbolgen;
goptions htext=&BarLabelFontSizeAsPct.PCT ftext="&Font";
options nosymbolgen;
filename anyname "C:\9.2\Figure1.PNG";
proc gchart data=work.Top10;
hbar BarName /
  sumvar=BarValue discrete sumlabel=none descending maxis=axis1 raxis=axis2
  /* space=1 width=2 needed in 9.1.3 */
  ;
format BarValue comma13.;
run; quit;
```

To replace 2D bars with cylinders as in Figure 5, replace the HBAR statement with:

```
hbar3d BarName /
  sumvar=BarValue discrete sumlabel=none descending
  space=1 width=2 maxis=axis1 raxis=axis2 shape=cylinder noframe
  noplane /* NOPLANE must be used in AXIS statements instead for 9.1.3 */
  coutline=CX000000; /* COUTLINE not available for 9.1.3 */
```

Pie Charts

The problem with pie charts, if you want to provide full information (slice name, value, and percent share of whole), is that they are prone to label overlap. The custom legend in Figure 7, not the optional SAS/GRAPH legend, which is for slice name only, solves the problem. In Figures 8 and 9, suppressing details by use of an "Other" slice assists communication, rather than obstructing it. I created Figure 9 (and an innovative table for the details) while serving as elected Village Trustee to deflect complaints from residents. The Village Treasurer's tax bill also collects for other government units. I hope that comparison of Figures 10 and 11 will convince anyone who sees them that 3D pie charts, though popular, are an anti-communication tool, even if cute. The only 3D pie charts that do not distort relative size of shares are ones with either two or four slices of equal size. In such cases, who needs a pie chart?

Figure 6: Pie Chart with Label Overlap Problem

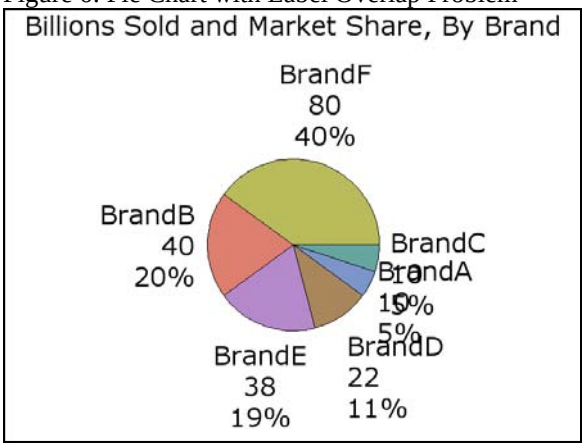


Figure 7: Pie Chart with Custom Legend

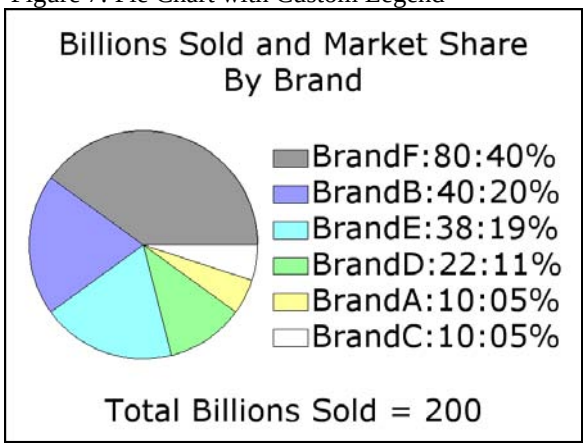


Figure 8: Insignificant Content Detail in "Other"

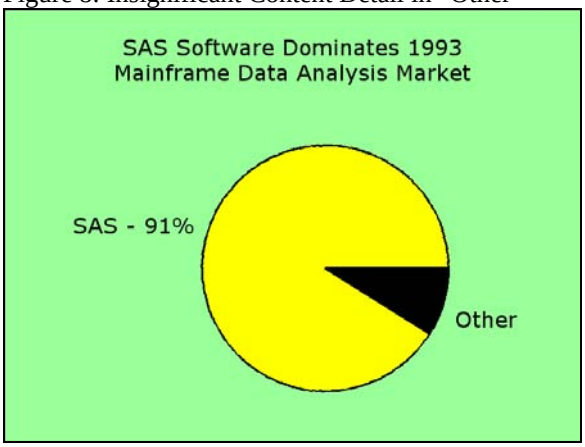


Figure 9: Emphasizing Smallness of "Non-Other"

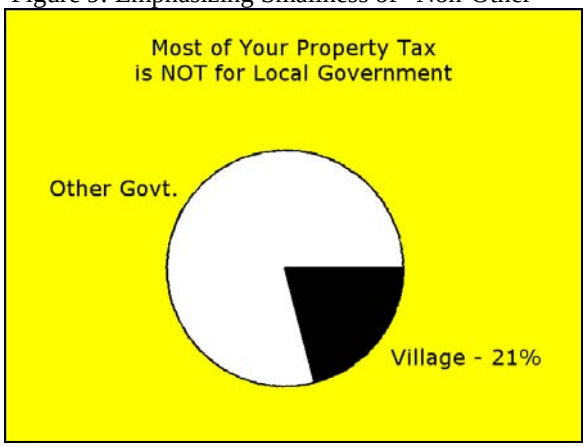


Figure 10: Communication-Defective 3D Pie Chart

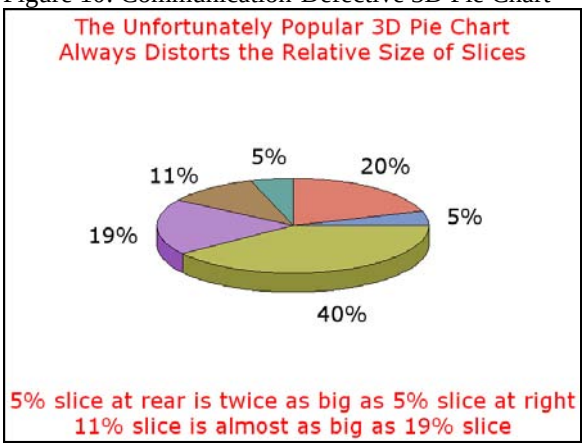
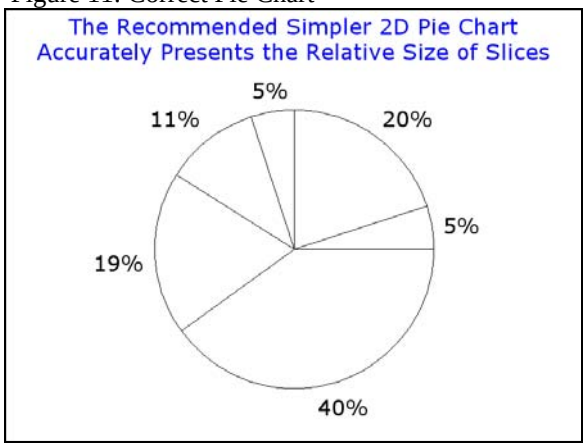


Figure 11: Correct Pie Chart



Here is the code to create Figure 7:

```
options mprint mprintnest;

data work.DataForPieChart;
infile datalines; input @1 SliceName $6. @8 Value 2.;
datalines;
BrandA 10
BrandB 40
BrandC 10
BrandD 22
BrandE 38
BrandF 80
; run;

%let TopBottomSpace = 3PCT;
%let FontSize = 7PCT;
%let Font = Verdana;
goptions reset=all;
goptions device=PNG;
goptions border; /* Put the graph in a box. */
goptions cback=CXFFFFFF; /* Use RGB White for background with multi-color pie. */
goptions htext=&FontSize ftext="&Font";
 /* Height & font for those parts of the graph which you do not explicitly assign,
    or for which no controls are available */
goptions gsfmode=replace gsfname=anyname;

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

 /* concatenate Percent and Value with the original Slice Name */
data SliceNameWithPercentAndValue;
length SliceNameWithPercentAndValue $ 15;
set DataForPieChart;
if _N_ eq 1 then do;
  set PieTotal;
  call symput('PieTotalForDisplay',trim(left(put(TotalValue,3.))));
end;
Percent = (Value / TotalValue) * 100;
SliceNameWithPercentAndValue =
  trim(left(SliceName)) || ':' || trim(left(put(Value,2.))) || ':' ||
  trim(left(put(Percent,Z2.))) || '%';
run;

proc sort data=SliceNameWithPercentAndValue;
by descending Percent; /* sequence the slices from largest to smallest */
run;

data SliceNameWithPercentAndValue;
set SliceNameWithPercentAndValue;
pctseq = _N_; /* add a key to merge this data with the ColorList    */
call symput('LegendEntry'||trim(left(_N_)),
            trim(left(SliceNameWithPercentAndValue)));
              /* Store the legend text entries in the symbol table. */
run;

data ColorList; /* Sequence the colors from darkest to lightest,
                   if you want small slices to stand out. */
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;

data SliceWithColor;
merge SliceNameWithPercentAndValue ColorList;
by pctseq;
run;

proc sort data=SliceWithColor;
by SliceNameWithPercentAndValue; /* because SAS/GRAPH will always apply
  your PATTERN statements in the sort order of the SLICE name text */
run;

data _null_;
set SliceWithColor;
call symput('SliceColor'||trim(left(_N_)),trim(left(color)));
            /* Store the slice colors in the symbol table. */
run;

%macro LegendEntries;
%do i = 1 %to 6;
  "&&LegendEntry&i"
%end;
%mend  LegendEntries;

legend1 order=(%LegendEntries) label=none shape=bar(6 PCT,4 PCT)
        across=1 position=(middle right outside) offset=(-2 PCT,0);

%macro PatternStatements;
%do i = 1 %to 6;
pattern&i v=psolid color=&&SliceColor&i repeat=1;
%end;
%mend  PatternStatements;

%PatternStatements;

title1 font="&Font" color=CX000000 height=&TopBottomSpace ' ' height=&FontSize
justify=CENTER
  "Billions Sold and Market Share"
  justify=CENTER "By Brand";
footnote1 angle=-90 font="&Font" height=1 ' '; /* push pie to the right */
footnote2 font="&Font" color=CX000000 height=&FontSize justify=CENTER
  "Total Billions Sold = &PieTotalForDisplay";
footnote3 font="&Font" height=&TopBottomSpace ' ';

filename anyname "C:\9.2\Figure7.PNG";
proc gchart data=SliceNameWithPercentAndValue;
pie SliceNameWithPercentAndValue / sumvar=Value
    noheading coutline=CX000000 descending
    legend=legend1
    slice=none value=none percent=none; /* turn off all pie labels */
run; quit;
```

Trend Charts

Sparse trend charts focus on what's important. Precise numbers are most easily provided in a companion table, or, for a web-enabled graph, in a pop-up box of text. However, there are always four precise numbers of interest to a typical viewer of a trend chart: start, end, minimum and maximum. You can use PROC MEANS to preprocess data to find min and max, and pass them to the graph creation step as macro variables. There can be other points of interest on a trend chart (as, e.g., in Figure 13), but they are not amenable to automated detection. For sparse ways to present a year of weekly data or a month of daily data, please see Reference 1. Figures 15-18 deliver complete detailed information. The PNG driver does not work correctly for Figures 17 & 18, and the ZPNG driver does not support transparency, which is used in web-enabled Figure 18. So, the ZGIF driver is used for Figures 17 & 18.

Figure 12: Annotate Start, End, Min, and Max

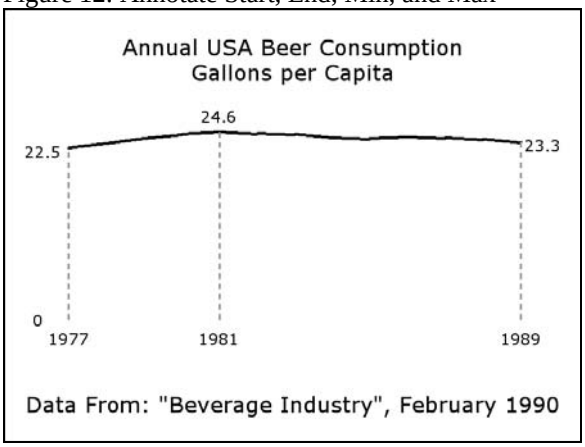


Figure 13: Annotate Start, End, and Change Point

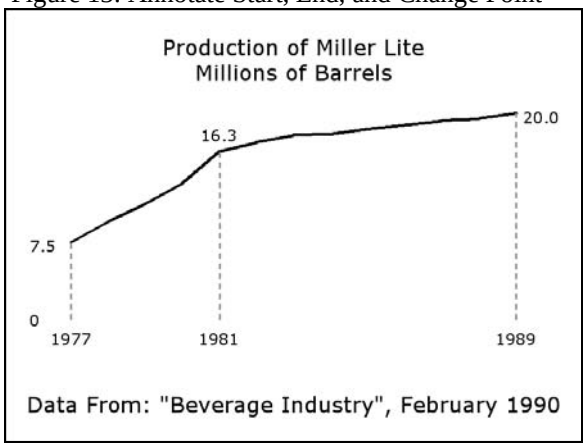


Figure 14: Identify Start/End Value, Min/Max Date

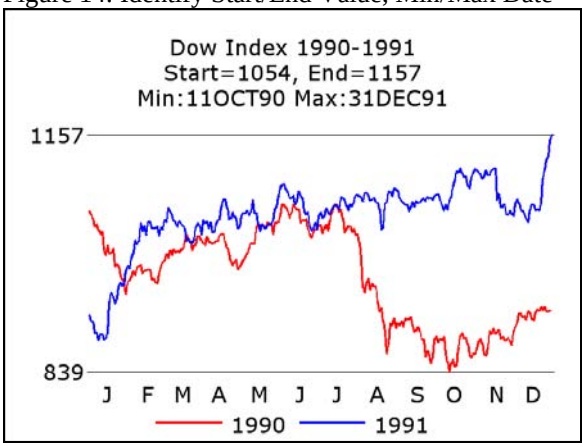


Figure 15: Trend with Tick Mark Value Table

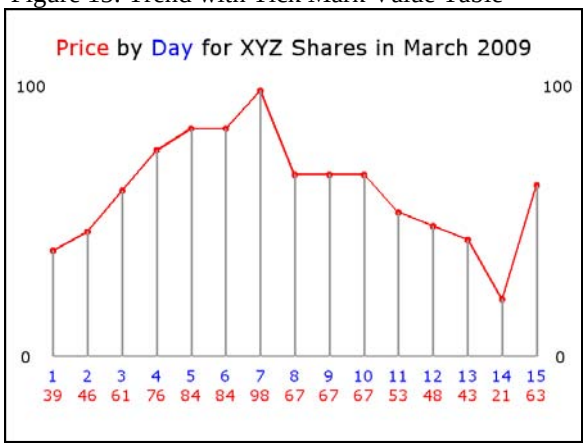


Figure 16: Not the Collision-Prone POINTLABEL Option

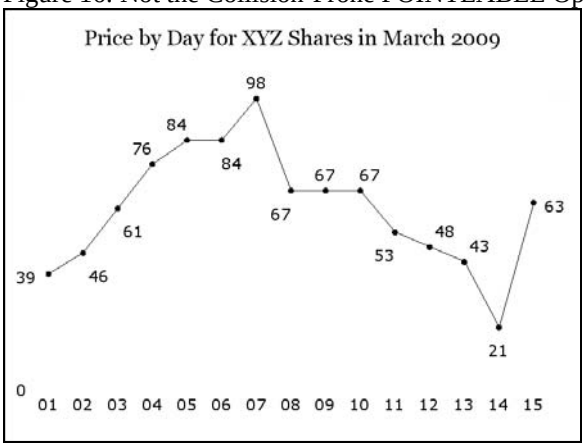


Figure 17: Expandable to 25 months with small font

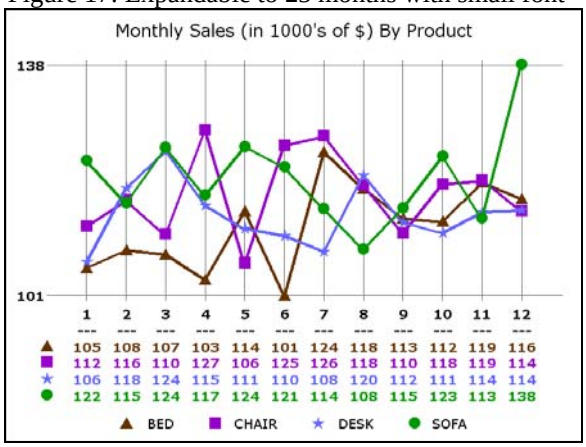
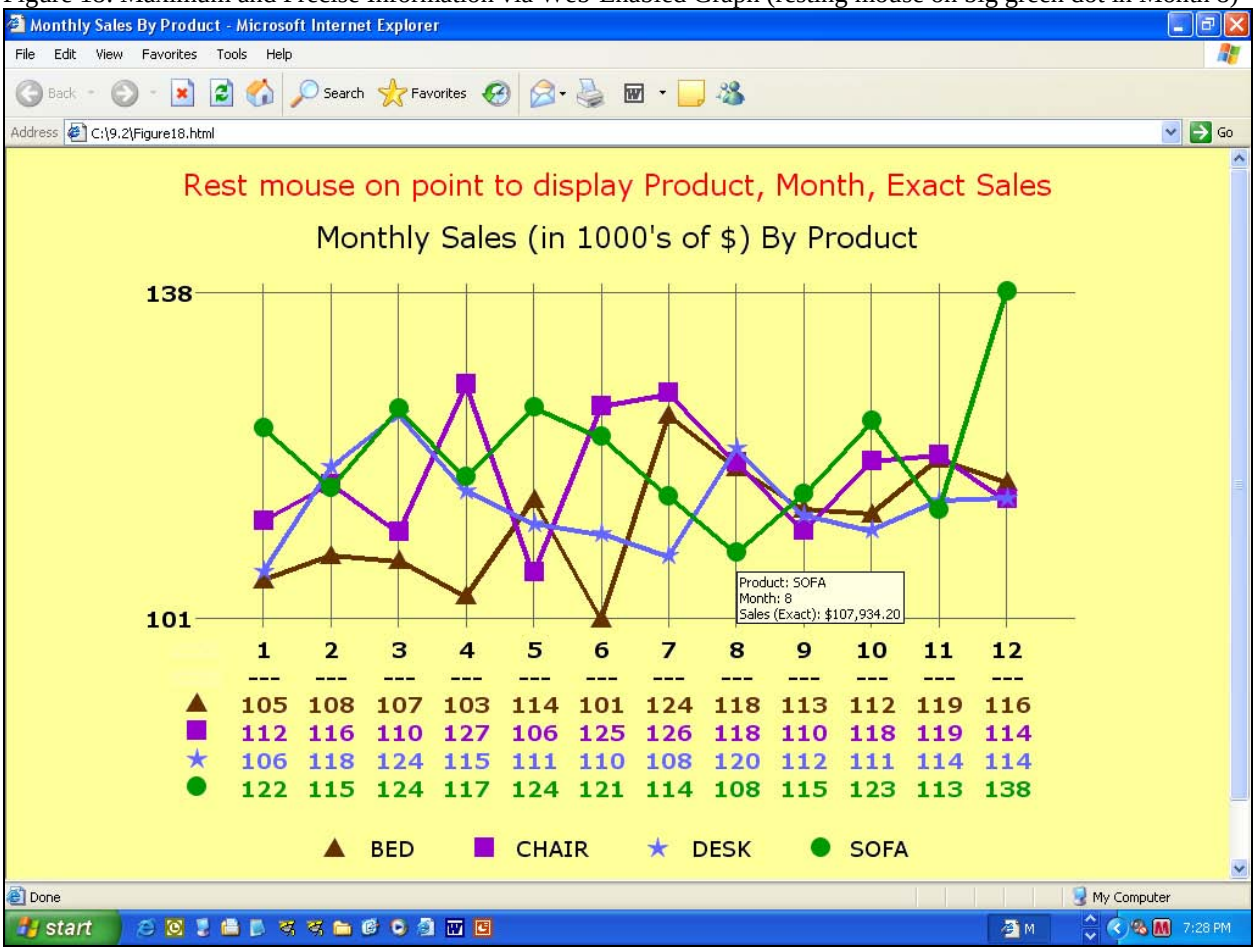


Figure 18: Maximum and Precise Information via Web-Enabled Graph (resting mouse on big green dot in Month 8)



Here is the code to create Figure 17:

```
options mprint mprintnest;

data DataForFourLineTrendPlot(drop=Date Year);
set sashelp.prdsal3(where=(Year=1998) keep=Year Product Actual Date);
Month = month(Date);
run;

proc summary nway data=DataForFourLineTrendPlot;
class Product Month;
var Actual;
output out=DataForFourLineTrendPlot(drop=_type_ _freq_) sum=Actual;
run;

data ToPlot;
drop Actual;
set DataForFourLineTrendPlot;
Sales = Actual / 1000;
run;

%macro TickMarkValueTable(FontSize=,BackgroundColor=CXFFFFFF);
tick=1 color=&BackgroundColor height=&FontSize font='Verdana/Bold' "XXX"
justify=CENTER height=&FontSize font='Verdana/Bold' "XXX"
  %do j = 1 %to &Zcount;
justify=CENTER height=&FontSize color=&&color&j
font='Monotype Sorts' "&&winMarker&j"X
  %end;
%do i = 2 %to %eval(&Xcount + 1);
```

```
tick=&I color=CX000000 height=&FontSize font='Verdana/Bold' "&&Xvalue&i"
justify=CENTER height=&FontSize font='Verdana/Bold' '---'
  %do j = 1 %to &Zcount;
justify=CENTER height=&FontSize font='Verdana/Bold' color=&&color&j
"&&Y_value_X&i._Z&j"
  %end;
%end;
%mend  TickMarkValueTable;

%macro SymbolStatements(NumberOfThemNeeded=,SymbolSize=2 PCT,LineWidth=2);
%do i = 1 %to &NumberOfThemNeeded;
symbol&i interpol=join width=&LineWidth color=&&color&i height=&SymbolSize
         font='Monotype Sorts' value="&&winMarker&i"X;
%end;
%global TotalLineCount;
%let TotalLineCount = %eval(&NumberOfThemNeeded + 1);
symbol&TotalLineCount interpol=none color=CXFFFFFF;
%mend  SymbolStatements;

%macro Legend(NumberOfYvalueLines=,SymbolSize=2.50 PCT,TextSize=3.75 PCT);
%do i = 1 %to &NumberOfYvalueLines;
      height=&SymbolSize color=&&color&I font='Monotype Sorts' "&&winMarker&i"X
      height=&TextSize   color=CX000000  font="&Font" "  &&ZvalueDescription&i"
  %if &i ne &NumberOfYvalueLines %then %do;
      "       "
  %end;
%end;
%mend  Legend;

%let winMarker1 = 73; /* triangle */
%let winMarker2 = 6E; /* square   */
%let winMarker3 = 48; /* star     */
%let winMarker4 = 6C; /* circle   */

%let color1 = CX663300; /* brown  */
%let color2 = CX9900CC; /* violet */
%let color3 = CX6666FF; /* blue   */
%let color4 = CX009900; /* green  */

data ForPlot; drop Month Product;
set DataForFourLineTrendPlot;
xvalue=Month; yvalue=Actual / 1000; zvalue=Product;
run;

proc means data=ForPlot min max range noprint;
var yvalue;
output out=MinMaxRangeY min=miny max=maxy range=rangey;
run;

data _null_;
set MinMaxRangeY;
call symput('minY',miny);
call symput('maxY',maxy);
call symput('rangeY',rangey);
call symput('DisplayMinY',trim(left(put(miny,3.))));
call symput('DisplayMaxY',trim(left(put(maxy,3.))));
run;

proc sort data=ForPlot out=DistinctX nodupkey; by xvalue; run;

data _null_;
set DistinctX end=LastOne;
call symput('Xvalue'||trim(left(_N_ + 1)),trim(left(xvalue)));
if LastOne;
call symput('Xcount',trim(left(_N_)));
run;
```

```
proc sort data=ForPlot out=DistinctZ nodupkey; by zvalue; run;

data _null_;
set DistinctZ end=LastOne;
call symput('ZvalueDescription'||trim(left(_N_)),trim(left(Zvalue)));
if LastOne;
call symput('Zcount',trim(left(_N_)));
run;

proc sort data=ForPlot; by xvalue zvalue; run;

data VisiblePlotLines(keep=yvalue Actual xvalue XforPlot zvalue)
     AnyYvalueWouldSuffice(keep=yvalue);
retain WhichZ 0 WhichX 1;
set ForPlot end=LastOne;
by xvalue zvalue;
if first.xvalue then WhichZ = 1;
call
symput('Y_value_X'||trim(left(put(WhichX+1,2.)))||'_Z'||trim(left(put(WhichZ,2.))),
           trim(left(put(yvalue,3.))));
XforPlot = WhichX;
output VisiblePlotLines;
WhichZ = WhichZ + 1;
if last.xvalue;
WhichX = WhichX + 1;
if LastOne;
output AnyYvalueWouldSuffice;
run;

data HiddenPlotLine(keep=XforPlot yvalue zvalue);
retain zvalue 'Z';
set DistinctX(keep=yvalue) end=LastOne;
if _N_ eq 1 then set AnyYvalueWouldSuffice;
if _N_ eq 1 then do;
  XforPlot = 0;
  output;
end;
XforPlot = _N_;
output;
if LastOne;
call symput('LastXforPlot',XforPlot);
run;

data VisibleAndHidden; set VisiblePlotLines HiddenPlotLine; run;

%macro HrefValues;
%do i = 1 %to %eval(&Xcount);
  &i
%end;
%mend  HrefValues;

%let Space = 2.5PCT;
%let FontSize = 5PCT;
%let GraphFontSize = 3.75 PCT;
%let Font = Verdana;
goptions reset=all;
goptions device=ZGIF;
goptions htext=&GraphFontSize ftext="&Font";
goptions border;

title1 font="&Font" height=&Space ' '
       justify=CENTER height=&FontSize 'Monthly Sales (in 1000''s of $) By Product';
footnote1 %Legend(NumberOfYvalueLines=&Zcount,SymbolSize=&GraphFontSize);
footnote2 height=&Space ' '; /* This puts space below the legend. */
footnote3 angle=+90 height=&Space ' '; /* prevent VREF lines from extending
```

```
                                        to right-hand edge of page */
axis1 label=none major=none minor=none style=0 order=(0 to &LastXforPlot by 1)
      value=(%TickMarkValueTable(FontSize=&GraphFontSize)) offset=(0,7 PCT);
axis2 label=none major=none minor=none style=0 order=&miny to &maxy by &rangeY
      value=(height=&GraphFontSize font="Verdana/Bold" "&DisplayMinY" "&DisplayMaxY");
%SymbolStatements(NumberOfThemNeeded=&Zcount,LineWidth=3,SymbolSize=&GraphFontSize);

goptions gsfmode=replace gsfname=anything;
filename anything "&Path.\Figure17.GIF";
proc gplot data=VisibleAndHidden;
plot yvalue*XforPlot=zvalue / haxis=axis1 vaxis=axis2
     lhref=1 chref=CX666666 href=(%HrefValues) /* could use whref= in 9.2 */
     lvref=1 cvref=CX666666 vref=(&miny &maxy) /* could use wvref= in 9.2 */
     nolegend; /* suppress the automatic default legend */
run; quit;
```

Conclusions and Closing

Designs presented here provide both visuals for easy, quick inference, and detail numbers for reliable inference. Figures 8, 9, 12, 13, and 14 demonstrate the communication power of maximally simple graphics.

Though not explicitly compared here, it is an important fact that fonts are better rendered in 9.2 than in 9.1.3. Besides improved font rendering, the highest value 9.2 enhancement is that the bar labels in bar charts are no longer limited to 32 characters. The new limit of 256 more than accommodates any practical management graphic need.

The PNG driver in 9.2 supports transparency, always supported by GIF, and supports over 16 million colors, while GIF is still limited to 256. However, it is unlikely that a management graphic will need even 256 colors. This paper uses PNG for Figures 1-16, but has to rely on ZGIF (a backward-compatible version of GIF) for Figures 17 and 18.

For more about the effects of changes in SAS/GRAPH 9.2, please see Reference 3.

Pie label overlap vulnerabilities will always be with us in any future release, but the vulnerability to label overlap when using the POINTLABEL option for GPLOT SYMBOL statements could have been reduced by adoption of the logic in my previously published algorithm used to produce Figure 16. The POINTLABEL overlap problem has been repeatedly reported by the author (see, e.g., Reference 2).

The challenge of relying on SAS/GRAPH annotation is avoided by the “tick mark value table” examples in Figures 15, 17, and 18, which, by using a small enough font, have been successfully extended by the author to reporting as many as 25 months (latest report month back to corresponding month two years ago).

References

1. SAS Graphs for a BlackBerry, iPhone, or Other Small Email Screen: Extreme SAS/GRAPH and the Necessity and Power of Simplicity, *Proceedings of the SAS Global Forum 2008 Conference*. Cary, NC: SAS Institute Inc. This tutorial by the author is at http://www2.sas.com/proceedings/forum2008/034-2008.pdf.

2. Get the Best out of SAS/GRAPH and ODS, *Proceedings of the SAS Global Forum 2007 Conference*. Cary, NC: SAS Institute Inc. This tutorial by the author is at http://www2.sas.com/proceedings/forum2007/228-2007.pdf.

3. Changes in SAS 9.2 SAS/GRAPH Software. This excellent resource by Robert Allison is at http://support.sas.com/kb/31/addl/fusion_31476_1_changesinsas92robertallison.pdf

Contact Information

Your comments, questions, and suggestions are welcome. All code and full-size figures, for both Version 9.1.3 and Version 9.2, are available in a zip file upon email request to the author.

LeRoy Bessler PhD
Email: Le_Roy_Bessler@wi.rr.com