Paper HW-05-2013

Building a Better Bar Chart with SAS® Graph Template Language

Perry Watts, Stakana Analytics, Elkins Park, PA

ABSTRACT

This workshop combines instructions for chart building with principles defined by the statistical graphics experts: Edward Tufte, William Cleveland, and Naomi Robbins. Step-by-step instructions are provided for building basic and group charts that display frequencies, sums, percents, and means with associated confidence intervals. Group charts are further subdivided into repeating and non-repeating categories. For repeating group charts varying bar displays such as *stacked*, *cluster*, and *nested* will also be reviewed.

Exercises make use of the BARCHART statement included in the SAS® 9.3 Graph Template Language Reference manual. Along the way, comparisons are made between GTL's BARCHART statement and the GCHART procedure in SAS/GRAPH software. The data used in the exercises come either from the SASHELP library or from pre-summarized data read into WORK data sets with a DATALINES statement.

While there may not be time in the workshop to cover advanced topics, two enhanced bar charts will be covered in the paper. Intermediate to advanced SAS programmers with experience using any graphics software package will get the most out of this presentation.

THE BAR CHART AS A GRAPHIC CONSTRUCT

DEFINITION:

From Wikipedia:

> A **bar chart** or **bar graph** is a chart with rectangular bars with lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally. A vertical bar chart is sometimes called a column bar chart.
>
> **What it is:**
> A bar graph is a chart that uses either horizontal or vertical bars to show **comparisons** among categories (emphasis added).

Note that the categories being compared consist of discrete nominal or ordinal data. Histograms are better suited for displaying ranges of continuous data.

It can also be deduced from the Wikipedia definition that information in a bar chart is conveyed solely by bar length. As Figure 1 demonstrates, bar widths can be changed without altering the graph's message.

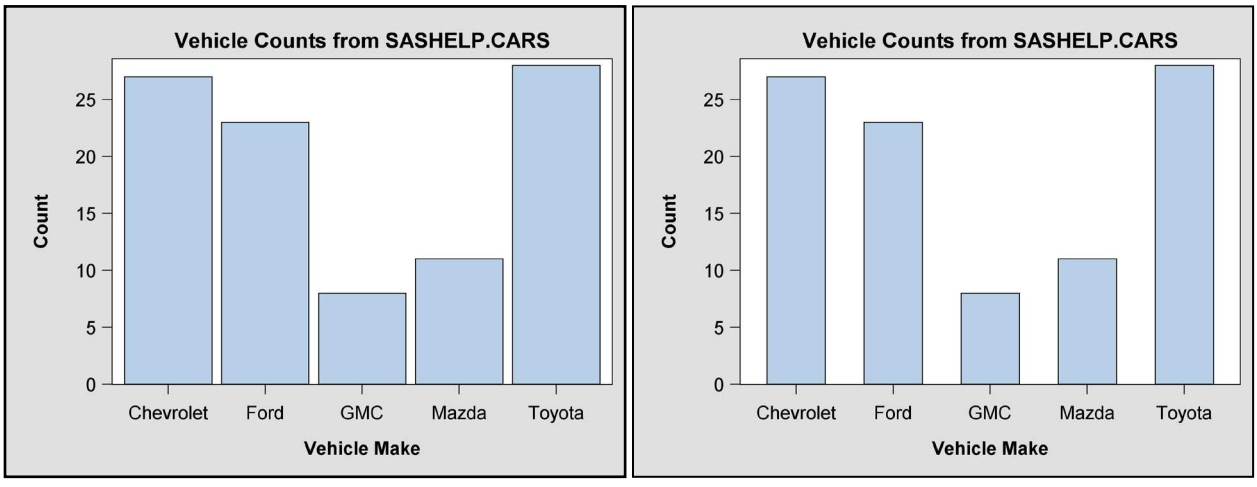

Figure 1. Two bar charts convey exactly the same information, because bar width has no intrinsic meaning.

HISTORY:

Displayed in Figure 2 is a hazy reproduction of one of the earliest bar charts created by William Playfair. Playfair's chart is entitled *CHART, Shewing at One View The Price of The Quarter of Wheat, & Wages of Labour by the Week --from-- The Year 1565 to 1824*. A more readable version of this graph appears in *The Visual Display of Quantitative Information* by Edward Tufte **[Tufte 2001, p. 34].**

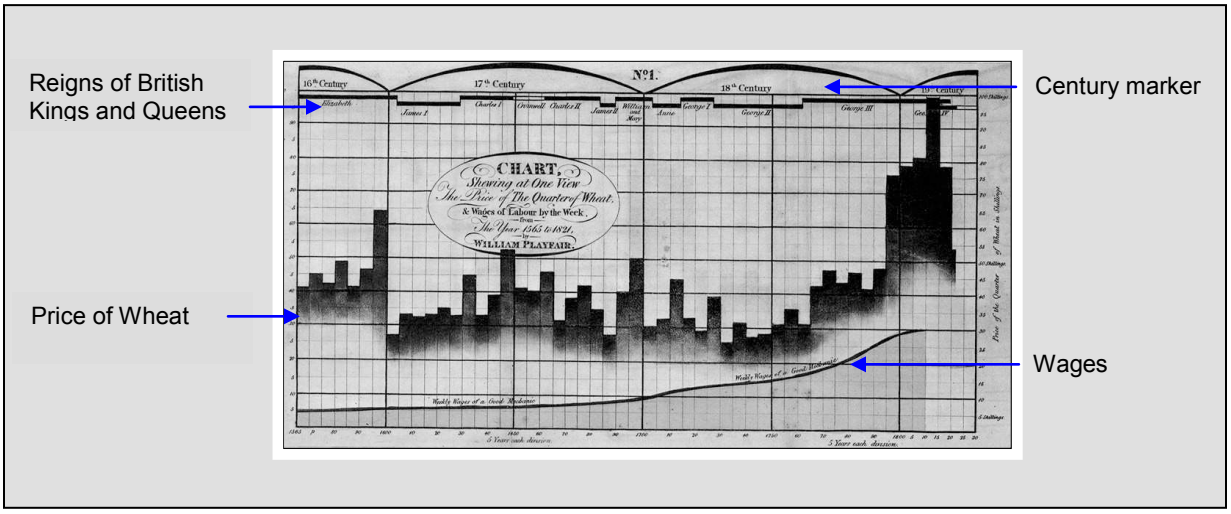

Figure 2. An early bar chart by William Playfair compares the price of wheat to the wages for skilled laborers.

Playfair enhanced his chart by adding series plots for wages, reigns of monarchs and centuries. With these enhancements he anticipated the arrival of LAYOUT OVERLAY in GTL that makes it possible to enhance a bar chart with the addition of points, lines and annotation. Nevertheless, the stand-alone bar chart with its simple structure has managed to endure over the centuries. Possibly its longevity can be attributed to its transparency. Bar charts use summary data to convey brief messages quickly!

THE BAR CHART GETS BAD PRESS

Despite their popularity, graphics experts do not have a high regard for bar charts. Cleveland makes no reference to them in *The Elements of Graphing Data* **[Cleveland, 1994]**, and Tufte points to a low *data-ink ratio* defined below as a reason for not taking them seriously.

$$\text{Data-ink ratio} = \frac{\text{data-ink}}{\text{total ink used to print the graphic}}$$

$$= \text{proportion of a graphic's ink devoted to the non-redundant display of data-information [Tufte, 2001, p. 93]}$$

For Tufte, the only data-ink in a bar chart is the height or altitude of the bars. Furthermore, he shows that redundancy increases when a single labeled shaded bar conveys the same "altitude in six separate ways (any five of the six can be erased and the sixth will still indicate the height)" **[Tufte, 2001, p. 96].** However, the data-ink ratio is not an appropriate metric to apply when the stated goal of a graphic is to *compare* results (see Wikipedia definition). In Figure 3, for example, the scatter plot portrays a confusing relationship between HITS and SALARY even though the data-ink score is much higher. The confusion can be attributed to the presence of overlapping plotting symbols in the scatter plot.

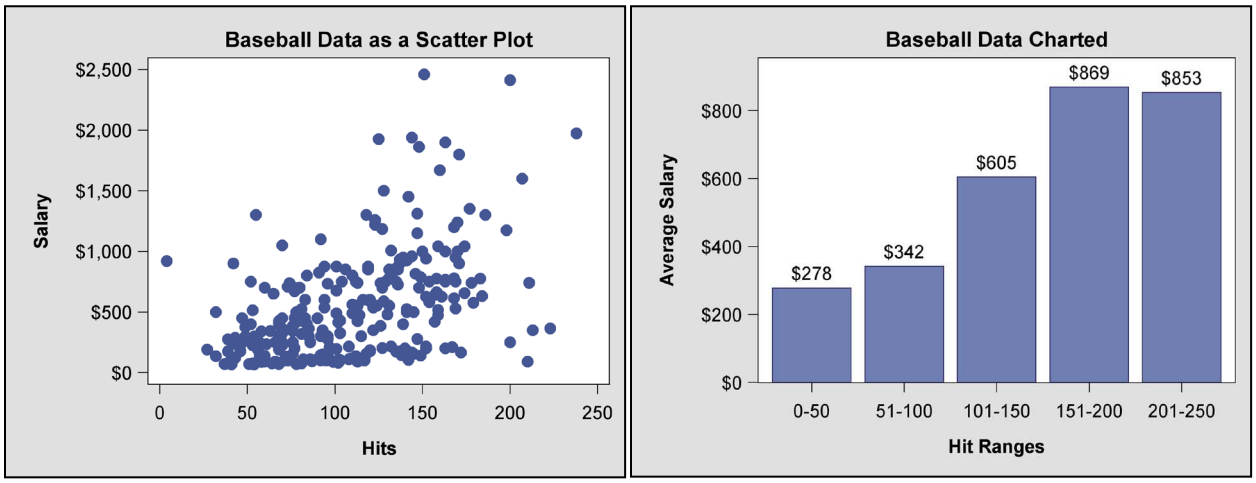

Figure3. Tufte's data ink ratio collides with Cleveland's maxim that "overlapping plotting symbols must be visually distinguishable" **[Tufte, 2001, p. 50]**. Overlap is not an issue with a bar chart, because the data are summarized. Data source in **[StatLib, 1988]**.

ODS SETUP FOR BAR CHARTS IN GTL

THE STYLE TEMPLATE REPLACES SAS/GRAPH'S GOPTIONS STATEMENT

The overall appearance of a graph is controlled by the STYLE template in ODS. The STYLE template pretty much replaces GOPTIONS in SAS/GRAPH software. Both packages contain in-line options for further refinements that take precedence over general settings. However, ODS STYLE shines over GOPTIONS by using inheritance to eliminate the need for coding defaults. This way the programmer can hone in on defining customized settings for a subset of attributes. The interaction between inheritance and customization is best illustrated by example:

```
ODS PATH work.templat(update) sashelp.tmplmst(read);❶
PROC TEMPLATE;
   DEFINE STYLE bigText;          ❷
     PARENT = styles.default;     ❸
     CLASS graphfonts             ❹
      "Fonts used in graph styles" /
      'GraphTitleFont' = (", ",16pt,bold)
      'GraphLabelFont' = (", ",14pt,bold)
      'GraphValueFont' = (", ",14pt);
     CLASS graphBorderLines /
       linethickness = 0px;       ❺
    END;
 RUN;
 PROC TEMPLATE;
   DEFINE STYLE myColors;         ❻
     PARENT = bigText;
     CLASS graphDataDefault /
       contrastcolor=CX445694     ❼
       color=CX6F7EB3;            ❽
     CLASS graphOutlines /
       linethickness = 2px
       contrastcolor = CX292561; ❾
   END;
 RUN;
```

❶ The source template store for STYLES.DEFAULT is SASHELP.TMPLMST. To prevent any changes from being made, it is opened in READ mode. On the other hand, WORK.TEMPLAT is where the newly created templates will be stored. WORK.TEMPLAT accepts changes but goes away at the end of the session. Creating a temporary template makes iterative graphics development possible in GTL.

❷ A new style template BIGTEXT is being created. It will be stored in WORK.TEMPLAT.

❸ BIGTEXT inherits approximately 400 attribute settings from STYLES.DEFAULT.

❹ The style element GRAPHFONTS is changed with a CLASS statement. The term CLASS is short-hand for STYLE GRAPHFONTS FROM GRAPHFONTS. What CLASS is telling SAS to do is to take the defaults from the GRAPHFONTS in STYLES.DEFAULT and store them together with any changes to GRAPHFONTS in BIGTEXT. CLASS can always be used as a keyword when graphics style elements are being modified. On the other hand, inheritance for tabular style elements is much more versatile. For example style element HEADER can inherit attributes and their settings from HEADERSANDFOOTERS.

❺ No border is drawn around the graph. Instead the border is created in Microsoft WORD.

❻ A new style MYCOLORS is created. It inherits from BIGTEXT that continues to inherit from STYLES.DEFAULT.

❼-❽ Default color assignments are changed in MYCOLORS. The symbol color in the scatter plot comes from CONTRASTCOLOR in GRAPHDATADEFAULT, and the fill color in the bar chart comes from the COLOR attribute.

❾ The color for the bar outlines is changed from black to dark blue by changing CONTRASTCOLOR in GRAPHOUTLINES.

For more information about STYLE templates and style elements see Chapter 3 in *Statistical Graphics in SAS: An Introduction to the Graph Template Language and the Statistical Graphics Procedures* **[Kuhfeld, 2010, pp. 115-152].**

THE ODS DESTINATION STATEMENT

The following destination statement indicates where the graph will reside (GPATH=), the name of the STYLE template that will define its appearance (STYLE=), and the number of dots per inch for graphics resolution (IMAGE_DPI):

```
ODS listing
   GPATH = "C:\_MyGraphDir"
   STYLE = myColors
   IMAGE_DPI = 300;
```

The LISTING destination is chosen, because graphs can also be viewed in the RESULTS window. In this instance, STYLE points to the style template just created, and IMAGE_DPI is being set to a high value to improve the resolution of the output.

THE ODS GRAPHICS STATEMENT

The ODS GRAPHICS statement makes it possible for ODS to create graphics:

```
ODS GRAPHICS on / RESET IMAGENAME="Fig3a" IMAGEFMT=png ANTIALIASMAX=10000;
```

With this statement the graphics output that appears in the RESULTS window will also be written out to FIG3A.PNG. Antialiasing with a high number makes a graph less pixilated, smoother and all-around better looking.

THE STATGRAPH TEMPLATE AND PROC SGRENDER

Graphs produced in GTL are defined in a STATGRAPH template and generated in PROC SGRENDER where STATGRAPH template meets input data set. Two types of templates are used in GTL: STYLE and STATGRAPH. You have already seen an example of STYLE. STATGRAPH is addressed in this section.

GTL works hierarchically and top-down in code blocks to produce a graph. SAS code for the bar chart shown in Figure 3 is shown below. GTL hierarchy is outlined in the source code with rectangle overlays.

```
PROC TEMPLATE;
❶  DEFINE STATGRAPH myBChart;
❷    BEGINGRAPH;
       ENTRYTITLE "Baseball Data Charted";
❸      LAYOUT OVERLAY / xaxisopts=(label="Hit Ranges")
                        yaxisopts=(label="Average Salary"
                                   linearopts=
❹                                    (tickvalueformat=DOLLAR6.));
❺        BARCHART  X=hitGrp Y=salary / stat=MEAN barlabel=TRUE;
❻                                      barlabelattrs=(size=14pt)
❼                                      barlabelformat=DOLLAR6.;
       ENDLAYOUT; /* OVERLAY */
     ENDGRAPH; /* END GRAPH BLOCK */
   END; /* END DEFINE BLOCK */
RUN; /* END PROC TEMPLATE */
...
PROC SGRENDER DATA=work.sHits TEMPLATE=myBchart;
RUN;
```

❶ The outermost DEFINE block names the STATGRAPH template **myBchart**.
❷ The GRAPH block is where the graph title is inserted.
❸ LAYOUT OVERLAY contains the single plotting statement for this particular graph. Notice that axes are defined as options for the OVERLAY block. Since a bar chart is being plotted, SAS automatically assigns TYPE=DISCRETE to X axis and TYPE=LINEAR tor the Y axis. (TYPE can also be set by the user).
❹ One inline option is assigned to YAXISOPTS. TICKVALUEFORMAT= does as the name implies: it names a format that is applied to tick values along a LINEAR axis. TICKVALUEFORMAT= is not supported when TYPE=DISCRETE.
❺ LAYOUT OVERLAY typically supports multiple plotting statements; hence the name OVERLAY. In this particular example, only the BARCHART statement with X and Y parameters is entered. (The Y parameter is required, because stat=MEAN).
❻ BARLABELATTRS=(SIZE=14PT) takes precedence over the default setting of 7PT, found by looking at the GRAPHDATATEXT style element in the STYLES.DEFAULT template. Defaults are identified by style element affiliation for each option in the BARCHART statement **[SAS Institute, 2011a, pp.161-177].**
❼ Bar labels can be conveniently formatted with the BARLABELFORMAT option.

THE BASIC BAR CHART IN PROC GCHART AND GTL

While PROC GCHART in SAS/GRAPH software has significantly influenced the development of the BARCHART statement in GTL, there are basic structural differences between the two packages that should be mentioned. First are the axes configurations. Unique to PROC GCHART is the ***midpoint*** axis. This is the axis immediately below the axis line. No tick marks are displayed along a midpoint axis. A second ***group*** axis below the midpoint axis is also available when group bar charts are plotted in PROC GCHART. In addition there is a unique RAXIS or response-axis in GCHART, and separate HBAR and VBAR statements are used to generate horizontal and vertical bar charts.

In Graph Template Language there is only the X-axis for discrete categorical data and the Y-axis for continuous linear data. A ***group*** axis simply does not exist. Later on we see how to circumvent the requirement for one when group charts are created in GTL. Also, there are no HBAR or VBAR options in GTL. If you want to create a horizontal bar chart you do not change the X or Y parameter assignments. Instead you create a vertical bar chart and then add the option ORIENT=HORIZONTAL to the code.

With such a unique axis configuration in PROC GCHART, you have to count on the availability of an option in the procedure to get the graph you want. In GTL, on the other hand, charts can be enhanced by option **or** by adding different plotting statements to LAYOUT OVERLAY. For example, in PROC GCHART you get confidence intervals with the ERRORBARS= option whereas in GTL you need to issue a separate SCATTERPLOT statement to add them to your graph. Specific instructions for generating a bar chart with confidence intervals are provided later in the paper. Below in Figure 4 you will see a comparison between GTL and GCHART output for the basic bar chart.

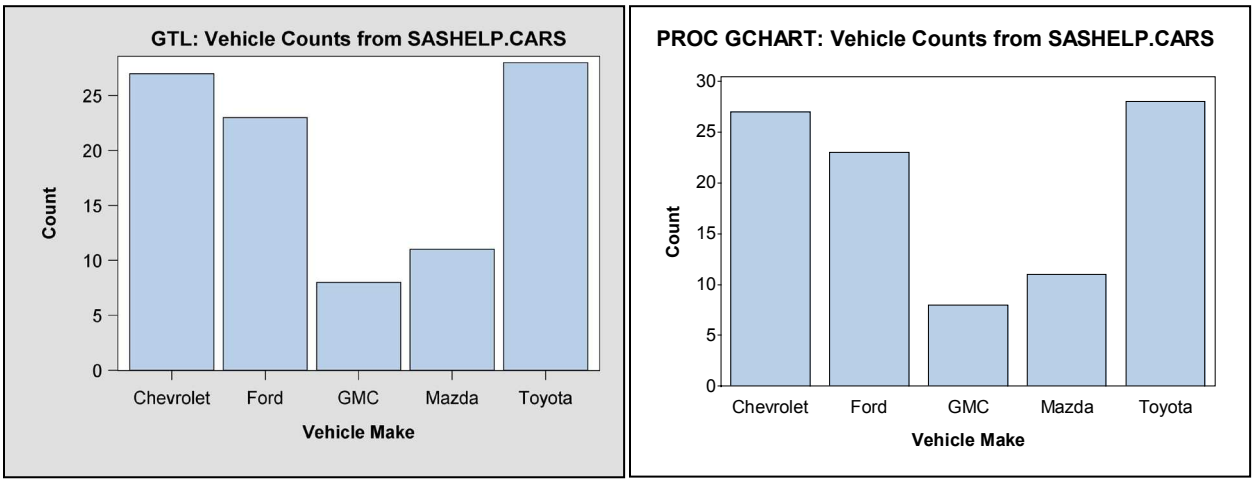

Figure 4. If appearances are downplayed, GTL and PROC GCHART produce similar bar charts. Where the charts differ is in their axes configurations. Ticks are found along the discrete X-axis in GTL whereas they are missing in the GCHART graph. The Y-axis is also different. In the GTL version, bar heights exceed the maximum tick value of 25 whereas the maximum tick value must always be greater than the tallest bar in PROC GCHART.

VARIATIONS ON THE BASIC BAR CHART IN GTL

In this section, bar charts that support frequencies, means, sums and percents are presented. Since code for a no-frills bar chart has been listed for in Figure 3, enhancements will be featured in this section. Also described here are difficulties encountered when charting percents, ordering bars, and inserting confidence limits into a graph created from raw data. Workarounds involve pre-summarizing the data.

CHARTING FREQUENCIES AND SUMS WITH RAW DATA

FREQUENCIES

Since we have already encountered a bar chart for means in Figure 3, let's start off by charting frequencies and sums. In Figure5, a more glitzy frequency bar chart with a non-repeating group affiliation is displayed. If the bar chart in Figure 5 were supporting a repeating group, then ***each*** MAKE (Chevrolet ...Toyota) would have two bars representing the two car manufacturer origins for a total of 10 bars in the graph. Repeating group bar charts are discussed later in the paper.

To make a chart with a similar format in PROC GCHART, use the NOZERO option that suppresses bars when the chart statistic is zero. For example, when 'Chevrolet' is processed, the Asia count would be zero and no room would be made for the bar in PROC GCHART. However, in repeating group bar charts zero counts are informative, so the NOZERO option should be suppressed.

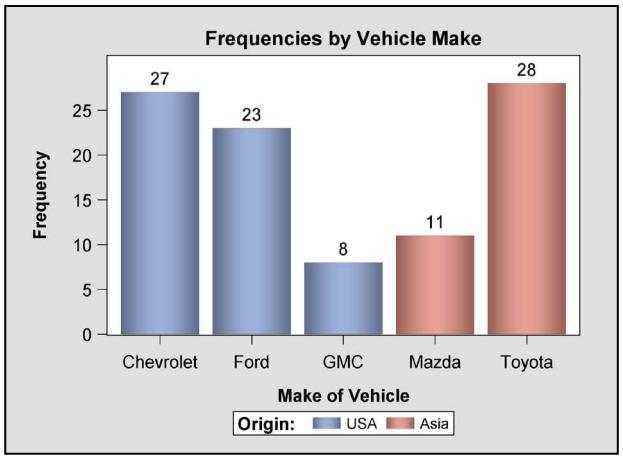

Figure 5. The frequency bar chart in GTL is enhanced by adding a *pressed* data skin. The two colors assigned to the bars come from GRAPHDATA1 (blue) and GRAPHDATA2 (red) style elements in the STYLES.DEFAULT template. There are 12 such style elements containing contrasting colors. Group affiliations can easily be identified with contrasting colors.

Code for the STATGRAPH template used for generating the bar chart in Figure 5 is listed below:

```
proc template;
   define statgraph MYBCHART;
   begingraph;
     entrytitle "Frequencies by Vehicle Make";
      LAYOUT OVERLAY / XAXISOPTS=(LABEL="Make of Vehicle"); ❶
         BARCHART X=MAKE /                                  ❷
            group=ORIGIN name="OGROUP"                      ❸
            dataskin=PRESSED                                ❹
            barlabel=TRUE
            barlabelattrs=(size=12PT);
         DISCRETELEGEND "OGROUP" /                          ❺
            across=2
            titleattrs=(size=10PT)
            valueattrs=(size=9PT)
            border=TRUE
            borderattrs=(color=BLACK)
            title="Origin:";
       ENDLAYOUT;
     endgraph;
   end;
 run;
```

❶ Default settings are used for the Y-axis. Therefore, YAXISOPTS are not listed.

❷ A Y parameter assignment is not required when a frequency bar chart is being made. Also the STAT= option does not have to be specified, since FREQ is the default.

❸ The only way to create bars with different colors is with a GROUP= option. By assigning the variable ORIGIN to the GROUP= option, color can be used as a third dimension. The NAME= option provides a link to the legend that explains the newly added third dimension.

❹ Here is where the bar chart is jazzed up. Other choices available for DATASKIN include GLOSS, SHEEN, CRISP, and MATTE.

❺ Here is where the discrete legend "oGroup" is created. For more details, see the manual **[SAS Institute, 2011a, p.661-676]**.

SUMS

Any technique that increases the size of the plotting region in a graph should be used. Unfortunately, sums quickly add up to very large numbers that stretch out horizontally along the Y-Axis. In Figure 6, before and after graphs show one way to compress the width of the Y-axis region.

The right-side graph in Figure 6 also takes up the issue of bar width with the BARWIDTH= option. In GTL BARWIDTH is defined with a range of zero (narrowest) to 1 (widest where the adjacent bars are touching). The default 0.85 is used for the left-side graph in Figure 6 whereas the bar width is reduced to 0.75 in the chart on the right.

BARWIDTH in GTL is far more effective than the WIDTH-SPACE-GSPACE combination in PROC GCHART. WIDTH is GCHART'S BARWIDTH, SPACE is the space between the bars, and GSPACE is the space between groups. GTL has no need for SPACE and GSPACE whereas all three width options can be adjusted in PROC GCHART. Unfortunately though, the units of measurement are "character cells" that are difficult to estimate. Furthermore, if you enter numbers that are too high, the default chart that you are trying to change is reproduced along with a warning that tells you nothing about the largest values you can enter.

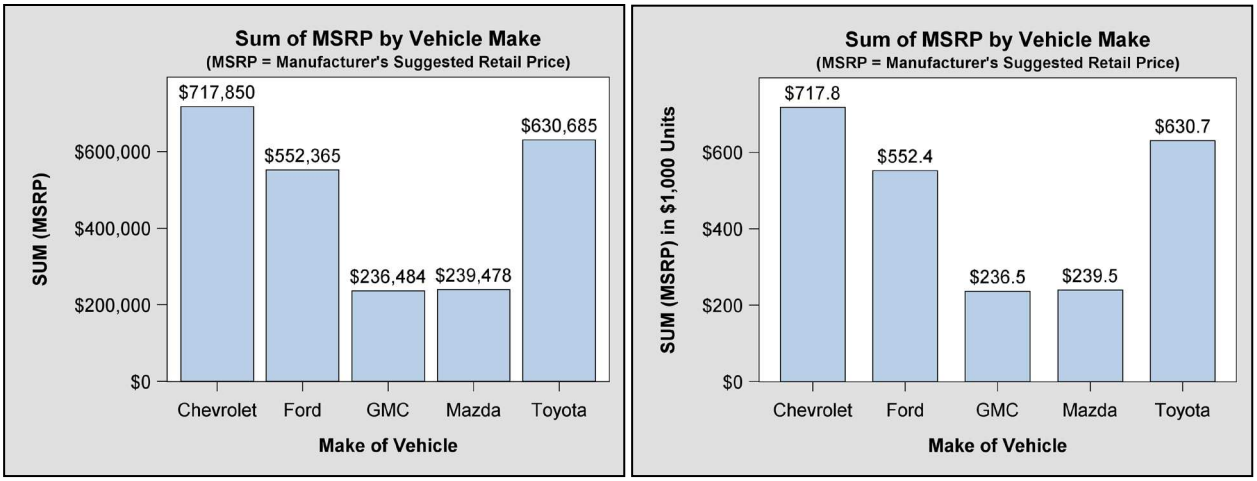

Figure 6. In the right-side graph, the width of the Y-axis region is reduced by setting Y= EVAL(MSRP/1000). Bar width can also be reduced in the right-side graph, because the number of digits in the bar labels has been reduced to four. Numbers won't spill over.

Code for the STATGRAPH template used for generating the bar chart on the right-side in Figure 6 is listed below:

```
proc template;
  define statgraph MYBCHART;
    BEGINGRAPH;
      ENTRYTITLE "Sum of MSRP by Vehicle Make";
      ENTRYTITLE "(MSRP = Manufacturer's Suggested Retail Price)" /
                  textattrs=(size=12PT);                          ❶
      LAYOUT OVERLAY /
        xaxisopts=(label="Make of Vehicle")
        yaxisopts=(label="SUM (MSRP) in Thousands of Dollars"
                  offsetmin=0
                  linearopts=(tickvalueformat=DOLLAR4.));         ❷

        BARCHART X=Make Y=EVAL(MSRP/1000) /                       ❸
                  barlabel=TRUE
                  barlabelattrs=(size=10PT)
                  barlabelformat=DOLLAR6.1                        ❷
                  barwidth=0.7;                                   ❹
      ENDLAYOUT;
    ENDGRAPH;
  end;
run;
```

❶ There is no TITLE2 in GTL. The only way to get smaller text for it is with a TEXTATTRS option.
❷ The Y-Axis and bar label formats are constructed the same way as they were in the code for Figure 3.
❸ Typically, GTL summary statistic functions are used with **EVAL** such that **NUMBER = EVAL(function-name(numeric-column)).** There are 32 summary statistic functions available in GTL: pretty much a subset of the statistics that are provided by **PROC UNIVARIATE**. In this instance, however, a simple arithmetic operation (division) is performed on the raw data.
❹ Here is where the bar widths are made smaller.

EXTENDING THE BARCHART STATEMENT IN GTL WITH PRESUMMARIZED DATA

PERCENT BAR CHARTS

The percent bar chart does not work as advertised in GTL, because fractions, not percents, appear in the output. This is unfortunate, since it is easier to compare bar heights when they are labeled with percents rather than frequencies. Recall from the definition that *comparison* is the main reason why bar charts are created.

The left-side graph in Figure 7 shows what happens when the following is executed:

```
BARCHART X=make / STAT=PCT ...;
```

With STAT=PCT, fractions are created in version 9.3 SAS. The only reliable solution to this problem is to summarize the data and alter the SAS code so that STAT=SUM (not PCT). When you work with summary data, the value itself is what appears in the graph. STAT=SUM works, because the sum of a single entity is itself. In this instance, bar labels match data set values for the Variable PCTVEHICLES in the Figure 7 right-side graph.

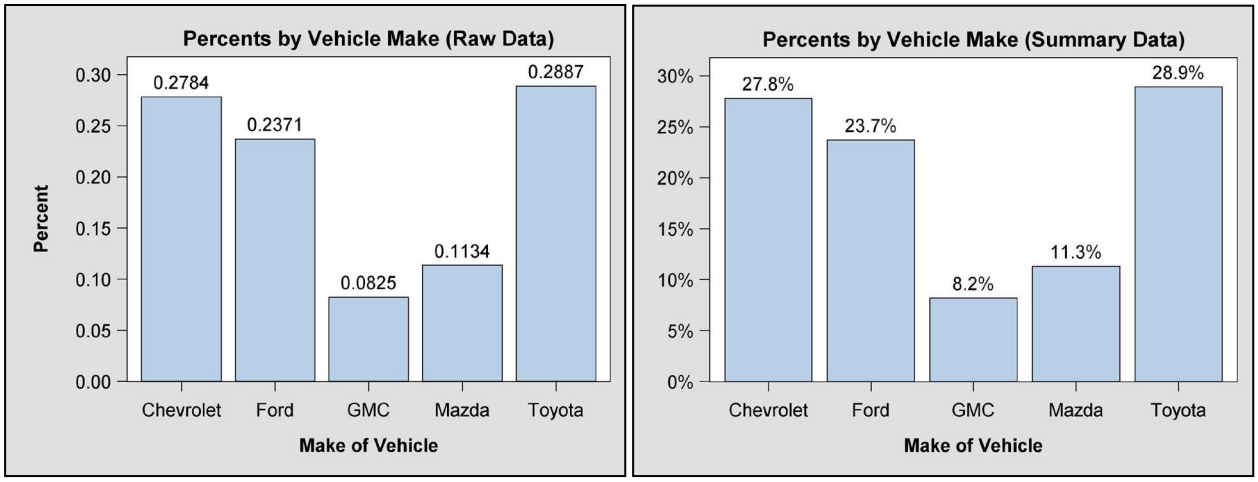

Figure 7. In the right-side graph, percents are correctly coded along the Y-axis and on top of the bars. Note that the word "Percent" has been removed as a Y-axis label. It is not needed, since all the tick values are shown with percent signs affixed to them. Its removal expands the data display region for better viewing.

What follows is a printout of the summary data for the right-side graph along with a review of source code for the relevant STATGRAPH template:

make	Pct Vehicles
Chevrolet	27.8
Ford	23.7
GMC	8.2
Mazda	11.3
Toyota	28.9

```
PROC FORMAT;                                                          ❶
  PICTURE pctYaxFm LOW-HIGH = '009%';
  PICTURE pctBarLF LOW-HIGH = '009.9%';
RUN;

proc template;
   define statgraph MYBCHART;
   begingraph;
     entrytitle "Percents by Vehicle Make (Summary Data)";
     LAYOUT overlay / xaxisopts=(label="Make of Vehicle")
                      yaxisopts=(DISPLAY=(LINE TICKS TICKVALUES)      ❷
                         linearopts=(tickvalueformat=PCTYAXFM.));

         BARCHART X=Make Y=PCTVEHICLES / stat=SUM                     ❸
                                         barlabel=TRUE
                                         barlabelattrs=(size=12PT)
                                         barlabelformat=PCTBARLF.;
       ENDLAYOUT;
     endgraph;
   end;
run;
```

❶ Picture formats are used to create the two percent annotations used in this graph. Format names are highlighted in blue. The BARCHART statement is a better choice than the BARCHARTPARM statement here, because the BARLABELFORMAT option is not a listed option for BARCHARTPARM.

❷ To totally remove an axis label from a graph, the DISPLAY= option has to be specified without the mention of the word "LABEL".

❸ PCTVEHICLES containing just five observations is assigned to the Y parameter and STAT=SUM is declared.

ORDERING BARS

Unlike PROC GCHART, the BARCHART statement in GTL has no such options as ASCENDING=, DESCENDING= or MIDPOINTS= for reordering bars in a graph. Bar order in GTL is determined exclusively by the order in which categorical variables appear in the input data set. Up to now you have been looking at graphs where vehicle makes are

listed in alphabetic order along the X-axis. This only happens because SASHELP.CARS is already sorted alphabetically by MAKE.

Let's say though that you want a graph of vehicle make averages ordered by MPG_HIGHWAY. You won't get them in the right order if you sort ***raw*** data by MPG_HIGHWAY. Instead you get the left-side graph shown in Figure 8. From this graph, you can determine that the vehicle make for the first record in the data set is "Ford" and that "Mazda" appears after all other makes in the data set have at least one entry. On the other hand when means are calculated in a summary data set that is then sorted, the bars come out in the ascending order that appears in the right-side graph of Figure 8.

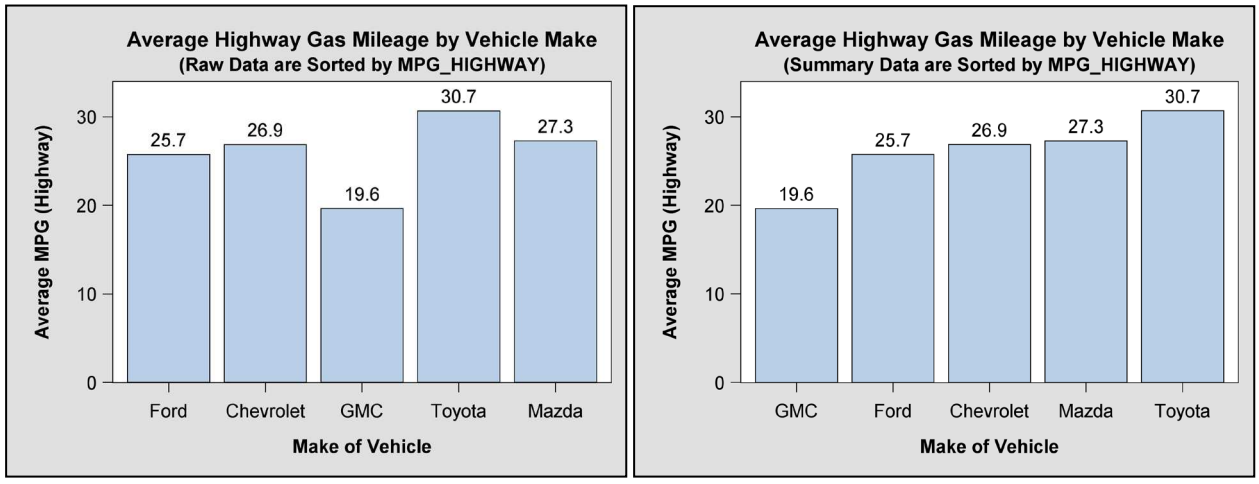

Figure 8. In the right-side graph, averages come out in ascending order only because the input summary data set has been sorted.

What follows is a print out of the sorted summary data for the right-side graph along with an abbreviated review of source code for the BARCHART statements applied to each chart:

make	Highway MPG
GMC	19.6250
Ford	25.7391
Chevrolet	26.8519
Mazda	27.2727
Toyota	30.6786

```
proc template; /* FOR LEFT-SIDE GRAPH - RAW DATA */
❶DEFINE STATGRAPH MYBCHART;
   begingraph;
     layout overlay / ... ;
❷      BARCHART X=Make Y=mpg_highway /
❷                    stat=MEAN BarLabel=TRUE ... ;
     endlayout;
   endgraph;
 END;
run;

proc template; /* FOR RIGHT-SIDE GRAPH - SUMMARY DATA*/
❶DEFINE STATGRAPH MYBCHART;
   begingraph;
     layout overlay / ... ;
❷      BARCHART X=Make Y=mean_HighwayMPG /
❷                    stat=SUM BarLabel=TRUE ... ;
     endlayout;
   endgraph;
 END;
run;
```

❶ MYBCHART is the name repeatedly given to the STATGRAPH template that resides in the WORK library. MYBCHART is overwritten each time PROC TEMPLATE is compiled and goes away at the end of the session.

❷ Note that the different values are assigned to the Y parameter and to the STAT= option in the BARCHART statements.

ADDING CONFIDENCE LIMITS TO MEANS

Confidence limits cannot be directly added to a bar chart, because there is no option for them in GTL's BARCHART statement. There are the LIMITS= and LIMITSTAT= options for the VBAR statement in the SGPLOT procedure **[SAS Institute, 2011b, p. 388]** and the ERRORBARS= option in PROC GCHART **[SAS Institute, 2004, p. 803].** For the workaround in GTL, a summary data set containing the interval of interest must first be created. Then a SCATTERPLOT statement is added in GTL to superimpose confidence limits over the bar chart. In Figure 9, confidence limits replace bar labels in the ascending bar chart from Figure 8.

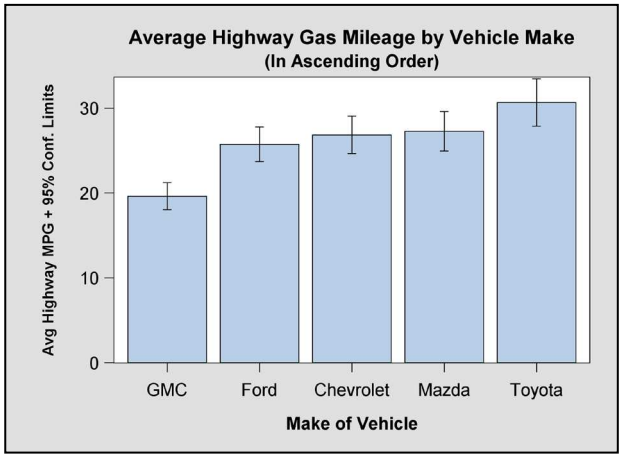

Figure 9. Confidence Limits replace bar labels in the ascending bar chart in Figure 8. When the summary data are created, care must be taken to define the ALPHA= option correctly in PROC SUMMARY.

Below is a printout of the input data set along with an abbreviated listing of the revised STATGRAPH template:

make	mean_ Highway MPG	L95Clm	U95Clm
GMC	19.6250	18.0263	21.2237
Ford	25.7391	23.7045	27.7738
Chevrolet	26.8519	24.6411	29.0626
Mazda	27.2727	24.9530	29.5924
Toyota	30.6786	27.8915	33.4656

```
proc template;
  define statgraph MYBCHART;
    begingraph;...;
❶      LAYOUT OVERLAY / xaxisopts=(type=DISCRETE ...)
                        yaxisopts=(type=LINEAR);

❷        BARCHART    X=make Y=mean_highwayMPG / stat=SUM;
         SCATTERPLOT X=make Y=mean_highwayMPG /
❸          markerattrs=(size=0)
❹          yerrorlower= L95CLM yerrorupper=U95CLM;
       ENDLAYOUT;
     endgraph;
   end;
run;
```

❶ The X-axis is discrete and the Y-axis is linear for both the BARCHART and SCATTERPLOT statements.
❷ The X and Y parameters in the BARCHART and SCATTERPLOT statements have identical assignments.
❸ While the SCATTERPLOT statement generates a graph with points, the points are hidden here. This is a fairly common "trick".
❹ All that is left from the SCATTERPLOT statement then are the YERRORLOWER= and YERRORUPPER= options; just what is needed.

THE REPEATING GROUP BAR CHART IN PROC GCHART AND GTL

In this section, we take up the repeating group bar chart where bars are "clustered in groups of more than one" **[wikipedia, Bar Chart]**. Repeating group comparisons are more complex, since bar categories and groups can be displayed in different orders. The total number of bars in a repeating group bar chart increases from ***m*** (number of bar

categories) to ***m*** X ***g*** (***m*** X #groups). A variation on the repeating group bar chart is the stacked bar chart that has the same axis configuration as the basic bar chart. Group affiliation in this type of chart is denoted by identifiable bar *segments*.

While PROC GCHART and GTL work with both group and stacked bar charts, structural differences exist between the two packages that are reviewed pictorially below. To find out more about constructing bar charts using the GCHART procedure see **[Watts, 2007a]**, **[Watts, 2007b]**, and **[Watts, 2008]**.

REPRESENTING A GROUP BAR CHART IN PROC GCHART

Unlike GTL, PROC GCHART supports two categorical axes: the midpoint axis that is contiguous with the bars and the group axis that appears immediately below the midpoint axis. Their relationship can be easily identified by looking at Figure 10 that shows the relationship between a categorized Body Mass Index (BMI) and the incidence of Chronic Heart Disease (CHD) for subjects who participated in the Framingham Heart Study (SASHELP.HEART).

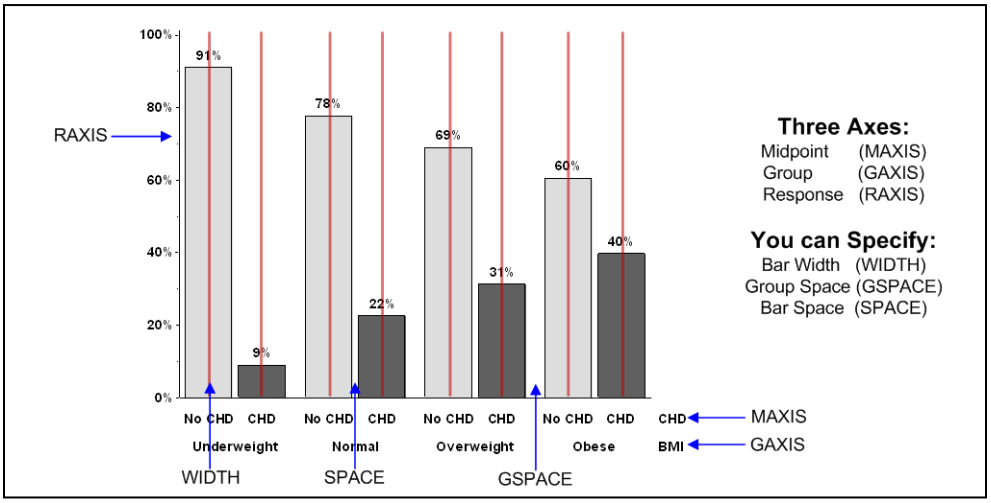

Figure 10. In Proc GCHART the X-axis is replaced by two axes: the midpoint and group axes (MAXIS, GAXIS). Note that a single group member references multiple midpoints (in this case "No CHD" and "CHD"). Also, programmers only have access to coordinates along the red lines. Everything else, including the coordinates for the GAXIS labels, is out of bounds.

DISPLAYING A GROUP BAR CHART WITH GTL'S BARCHART STATEMENT

In GTL, the midpoint axis is replaced by a generic discrete axis that can be used by many separate plotting statements. This flexibility is what made it possible to position confidence limits with a SCATTERPLOT statement over a bar chart of means in Figure 9. Also there is no group axis in GTL. Instead, group affiliation is indicated by legend. The graph from Figure10 is replicated in Figure 11 using a BARCHART statement from GTL.

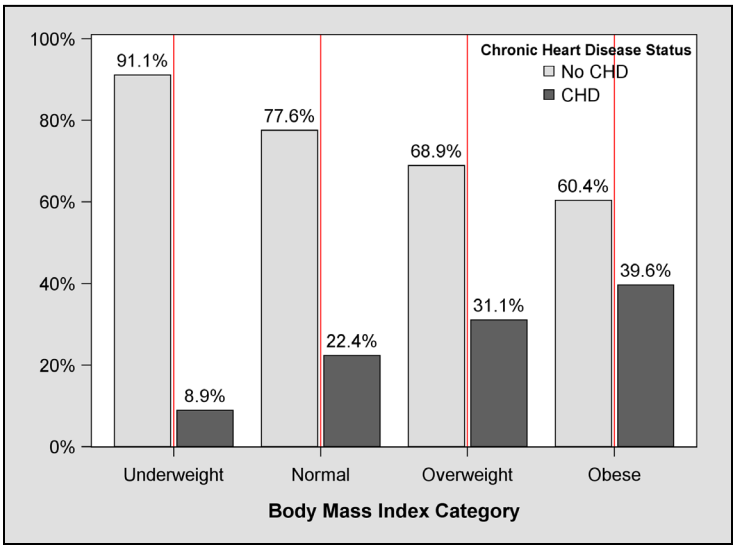

Figure 11. In GTL there is only a single X-axis, and in this graph it references BMI. CHD is relegated to the legend. While vertical red grid lines are associated with the discrete X-axis tick marks, they are not restrictive. Instead, in 9.3 SAS the entire data display area is available to the programmer. Now the off-gridline bar labels can be assigned with ease. GTL acknowledges its GCHART origins by calling the X-axis tick marks "category ***midpoints***" **[SAS Institute, 2011a, p. 171]** (emphasis added).

Let's enlarge upon comments in the figure captions from Figures 10 and 11. In PROC GCHART a single group member (BMI) references multiple midpoints (CHD) whereas this *one-to-many* relationship is reversed in GTL's BARCHART statement. In GTL the role of the midpoint is assigned to what was the group in GCHART, and the new "midpoint" (BMI) references many group members (CHD). So now we are looking at a *many-to-one* relationship. To drive the point home CHD, not BMI, is assigned to the GROUP= option in the BARCHART statement. This reversal can be very confusing to programmers coming to GTL from SAS/GRAPH software. Just remember: the ***group*** always gets the ***legend*** in GTL.

Clearly the four red lines in Figure 11 are not restrictive, because you see two bars spread out from each of them. In Figure 10, on the other hand, each bar represents a midpoint category even though tick marks are not drawn. The splaying in Figure 11 is made possible by the GROUPDISPLAY= option. When GROUPDISPLAY is set to CLUSTER, bars are spread out, and when the option is set to STACK a stacked (subgroup) bar chart is created. The programmer can control the width of the splayed bars with the BARWIDTH= option that combines GCHART'S WIDTH and corresponding SPACE options. GWIDTH along with GSPACE can be altered in GTL with the CLUSTERWIDTH= option. How BARWIDTH actually works is illustrated by example in the sections that follow.

WORKING WITH REPEATING GROUP BAR CHARTS IN GTL

Since comparisons are more complex in repeating group bar charts, percent bar charts are pretty much the norm in the examples from this section. Later, though, you will see an example that combines both frequency and percent displays plus another example with confidence bounds added to a repeating group bar chart of means.

With percents comes the need for working with summary data. The summary data, derived from SASHELP.HEART, is enhanced with the addition of Body Mass Index (BMI) from WEIGHT and HEIGHT variables in the data set:

$$BMI = 703 \times \frac{Weight}{Height^2}$$

What follows is a listing of a PROC FREQ command, its output, and how that output translates into a summary data set. Data have been pre-formatted for PROC FREQ input so that BMI and CHD (chronic heart disease) categories are printed out in the right order. For this run, CHD is assigned to the GROUP and BMI supplies the midpoints along the X-Axis. You can find the bar label percents in Figure 11 by scanning the highlighted rows the PROC FREQ output and the G100PCT column in the summary data set.

```
proc freq data=fmtedBMI;
  tables CHDordered*BMICatOrdered;
run;
```

CHD Frequency Percent Row Pct Col Pct	1:UnderW	2:Normal	3:OverWt	4:Obese	Total
0:No CHD	72	1905	1357	419	3753
OAPCT	1.38	36.64	26.10	8.06	72.19
MP100PCT	1.92	50.76	36.16	11.16	
G100PCT	91.14	77.57	68.88	60.37	
1:CHD	7	551	613	275	1446
OAPCT	0.13	10.60	11.79	5.29	27.81
MP100PCT	0.48	38.11	42.39	19.02	
G100PCT	8.86	22.43	31.12	39.63	
Total	79	2456	1970	694	5199
	1.52	47.24	37.89	13.35	100.00

SUMMARY DATA SET: MPbmi_GRchd MIDPOINT=BMI GROUP=CHD

BMI_ MPOrd	BMI_MPDesc	CHD_ GrpOrd	CHD_ GrpDesc	MPFreq	Grp Freq	OAPct	mp100pct	g100pct
1	Underweight	1	No CHD	79	72	1.4	1.9	91.1
1	Underweight	2	CHD	79	7	0.1	0.5	8.9
2	Normal	1	No CHD	2456	1905	36.6	50.8	77.6
2	Normal	2	CHD	2456	551	10.6	38.1	22.4
3	Overweight	1	No CHD	1970	1357	26.1	36.2	68.9
3	Overweight	2	CHD	1970	613	11.8	42.4	31.1
4	Obese	1	No CHD	694	419	8.1	11.2	60.4
4	Obese	2	CHD	694	275	5.3	19.0	39.6

Already you can see that there are three different percent bar charts that can be plotted: OAPCT for overall percent, MP100PCT where BMI categories sum to 100% for each CHD outcome, and G100PCT where CHD outcomes sum to 100 percent for each BMI outcome. But wait, there's more! If you reverse roles in PROC FREQ so that BMI is now assigned to the GROUP and CHD supplies the two midpoints along the X-Axis you will end up with a table that has the same percents occupying very different positions:

SUMMARY DATA SET: MPchd_GRbmi MIDPOINT=CHD GROUP=BMI

CHD_ MPOrd	CHD_ MPDesc	BMI_ GrpOrd	BMI_GrpDesc	MPFreq	Grp Freq	OAPct	mp100pct	g100pct
0	No CHD	1	Underweight	3753	72	1.4	91.1	1.9
0	No CHD	2	Normal	3753	1905	36.6	77.6	50.8
0	No CHD	3	Overweight	3753	1357	26.1	68.9	36.2
0	No CHD	4	Obese	3753	419	8.1	60.4	11.2
1	CHD	1	Underweight	1446	7	0.1	8.9	0.5
1	CHD	2	Normal	1446	551	10.6	22.4	38.1
1	CHD	3	Overweight	1446	613	11.8	31.1	42.4
1	CHD	4	Obese	1446	275	5.3	39.6	19.0

For example, values from G100PCT from the previous table now populate MP100PCT in the current table, but their order is different. So now the question is, "which data column should be charted?" The answer undoubtedly comes from the questions that are being asked of the data, but Naomi Robbins observes in her book *Creating More Effective Graphs* that "the closer together objects are, the easier it is to judge attributes that compare them" **[Robbins, 2005, p. 63]**. She then goes on to add "it is certainly easier to judge the difference in lengths of two bars if they are next to one another than if they are pages apart" **[Robbins, 2005, p. 63].** In Figure 12 the two highlighted data columns from the summary data sets are plotted in separate charts.

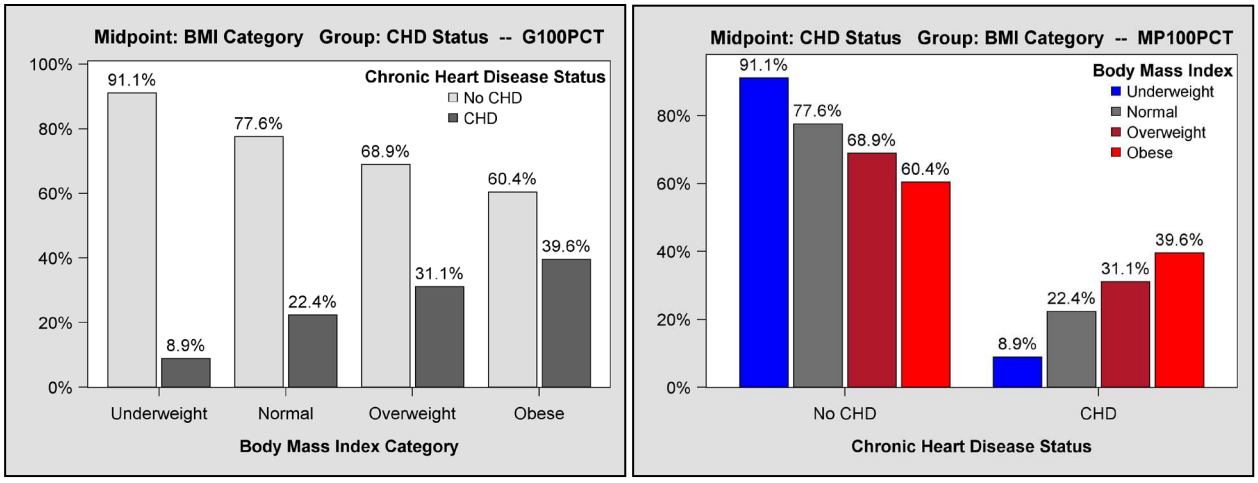

Figure 12. Bar heights are the same in the two graphs. Only the bars themselves have been rearranged. Robbins' observation is confirmed. It is easier to compare by *midpoint* in both graphs, because midpoint values are next to each other. For the left-side graph that would be BMI and for the right-side graph CHD. This is an instance, however, where G100PCT and MP100PCT could be perceived as misleading. Relatively few subjects in the Framingham heart study are underweight, but the height of the 91.1% bar overshadows the remaining bars in the graph. Bear in mind, though, *what* the graphs are saying. Underweight people typically do not have chronic heart disease.

CODING CLUSTER, STACKED AND NESTED GROUP BAR CHARTS IN GTL

CLUSTER CHARTS

In this section we take a look at how to code cluster, stacked and nested bar charts in GTL. Up to now you have been looking at the *cluster* format for repeating group bar charts. In Figure 13 we stick to the cluster format but switch **from** the G100PCT column in data set **MPbmi_GRchd** and MP100PCT from **MPchd_GRbmi** **to** MP100PCT in **MPbmi_GRchd** and G100PCT in **MPchd_GRbmi**. (See highlighted blue columns in the data listings above). Code for the right-side graph is reviewed in the discussion that follows.

Calculations for the G100PCT graphs in Figures 12 and 13 are easier to track visually. Bars over each midpoint category add up to 100 percent. For the MP100PCT graphs, calculations are a bit more challenging. Members of individual groups sum to 100 percent *across* all midpoints. To get to 100% in a MP100PCT chart just sum by bar color.

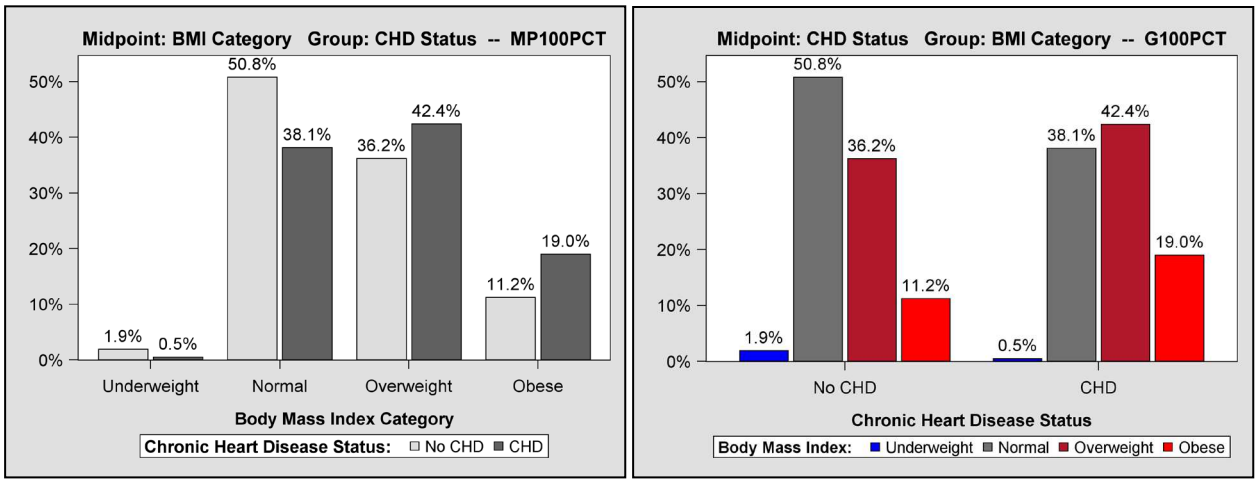

Figure 13. From this pair of graphs you see how few subjects are underweight in the Framingham Heart Study. That leaves the three remaining categories of the BMI to consider. From the left-side graph the percentage of obese people nearly doubles for those with CHD whereas the increase is far less for overweight individuals who acquire CHD. To look at the right side graph draw an imaginary line up from the tick marks at No CHD and CHD. Bars to the left of the line decrease in height when the transition from No CHD is made to CHD, and those to the right do the opposite. The connection between weight and chronic heart disease is made clear with the right-side graph.

Code for the STYLE and STATGRAPH templates used for generating the right-side bar chart in Figure 13 is listed below:

```
proc template;
❶   define style BMISTYLE;
     parent = BIGTEXT;
      class GRAPHDATA1 /
        contrastcolor=BLACK
        color=CX0000FF;       /* Under Weight (blue)*/
      class GRAPHDATA2 /
        contrastcolor=BLACK
        color=CX707070;       /* Normal Weight (gray)*/
      class GRAPHDATA3 /
        contrastcolor=BLACK
        color=CXB2182B;       /* Overweight (dark red)*/
      class GRAPHDATA4 /
        contrastcolor=BLACK
        color=CXFF0000;       /* Obese (bright red) */
    END;
RUN;

proc template;
   define statgraph MYBCHART;
   begingraph;
     entrytitle ...;
     layout overlay / xaxisopts=( ... )
                      yaxisopts=( ... );
❷      BARCHART Y=g100pct X=CHD_MPDesc /
❸        stat=SUM BARLABEL=TRUE barlabelattrs=(size=12PT)
         barlabelformat=PCTFMT.
❹        group=BMI_GrpDesc name="barGroup"
❺        groupdisplay=CLUSTER
❻        barwidth=0.90;
       DISCRETELEGEND "barGroup"  / ...;
     endlayout;
   endgraph;
   end;
run;

❼ ODS listing style=BMISTYLE image_dpi=300;
```

❶ The new BMISTYLE template changes defaults for the COLOR (for bar fill) and CONTRASTCOLOR (for bar outline). Blue (cold) is for light weight. Gray (neutral) is for normal weight. Dark Red (getting hot) is for overweight, and bright red is for obese.
❷ See data set listing for MPCHD_GRBMI on page 13 for columns assigned to the X and Y parameters.
❸ stat=SUM by default for a summary data bar chart. However, it is being spelled out here.
❹ The GROUP variable is identified with a pointer to a legend statement.
❺ Here is the command required for making a cluster bar chart.
❻ Bar width is increased from the default 0.85.
❼ BMISTYLE with the four colors is linked to the output.

STACKED BAR CHARTS

In a *stacked* bar chart, bars forming a group cluster are placed on top of each other in a right to left order. This action results in a bar chart where the number of bars is equal to the number of midpoint categories; exactly what you would have when a basic bar chart is plotted.

While simpler in structure, the stacked bar chart has a couple of drawbacks that are illustrated in Figure 14. With all four bars at the same height in the left-side graph of percents, the main task of comparison is severely compromised. Furthermore it is just about impossible to estimate the percent for each of the upper bar segments.

Stacked bar charts work better for frequencies. At least bar heights are different in the right-side graph in Figure 14. However, it is still hard to estimate frequencies for the upper bar segments even with the added bar labels.

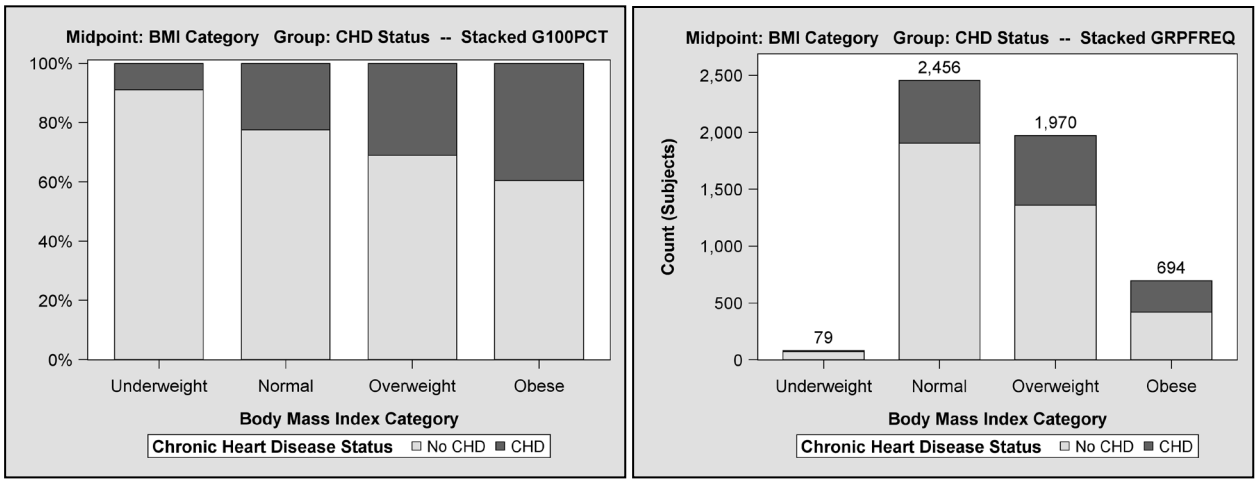

Figure 14. Conventional stacked bar charts such as those you see here do not work as well as cluster charts. However, for an alternative that combines percents from the left-side graph with frequencies from the right-side graph into a fully functional stacked bar chart see the combined frequency-percent bar chart in Figure 16.

An abbreviated code listing for the STATGRAPH template used for generating the left-side bar chart in Figure 14 is listed below. This time, the data set MPBMI_GRCHD on page 12 provides values for columns assigned to the X and Y parameters.

```
proc template;
  define statgraph MYBCHART;
  begingraph / ... ;
      layout overlay / xaxisopts=( ... )
                       yaxisopts=( ... );
        BARCHART Y=G100pct X=BMI_MPDesc /
                 stat=SUM barlabel=FALSE
                 group=CHD_GrpDesc name="barGroup"
               ❶ GROUPDISPLAY=STACK
               ❷ BARWIDTH=0.7;
        DISCRETELEGEND "barGroup"  / ...;
      endlayout;
    endgraph;
  end;
run;
```

❶ GROUPDISPLAY is set to STACK, even though STACK is the default (meaning that in this instance the GROUP= option is sufficient for generating a stacked bar chart).
❷ Bar width is narrowed, because there are so few bars. More white space is desired.

NESTED BAR CHARTS

The *nested* bar chart is similar to the stacked bar chart except that bars from a group cluster share a midpoint category and extend upwards from the base of the Y-axis. Matange and Heath state that such graphs have a "bar-bar" overlay, because multiple BARCHART statements need to be executed in order to create them **[Mantange and Heath, 2011, p. 49]**.

As seen from Figure 15, nested bar charts solve the two problems associated with stacked bar charts: bar heights for percents are no longer fixed at 100, and values for the nested bars can easily be estimated. Nevertheless bar heights in a nested bar chart are not guaranteed to be as consistent as they are in the left-side graph from Figure 15 where percentages for those with chronic heart disease are always lower than those who are disease free. In the right-side graph, a higher percentage of females fall into the normal weight category whereas their percentage is lower than that for males in the overweight category. Also all bars, with the exception of the underweight category, are fully visible in the right-side graph. Visibility is enhanced by an application of transparency available in ODS statistical graphics.

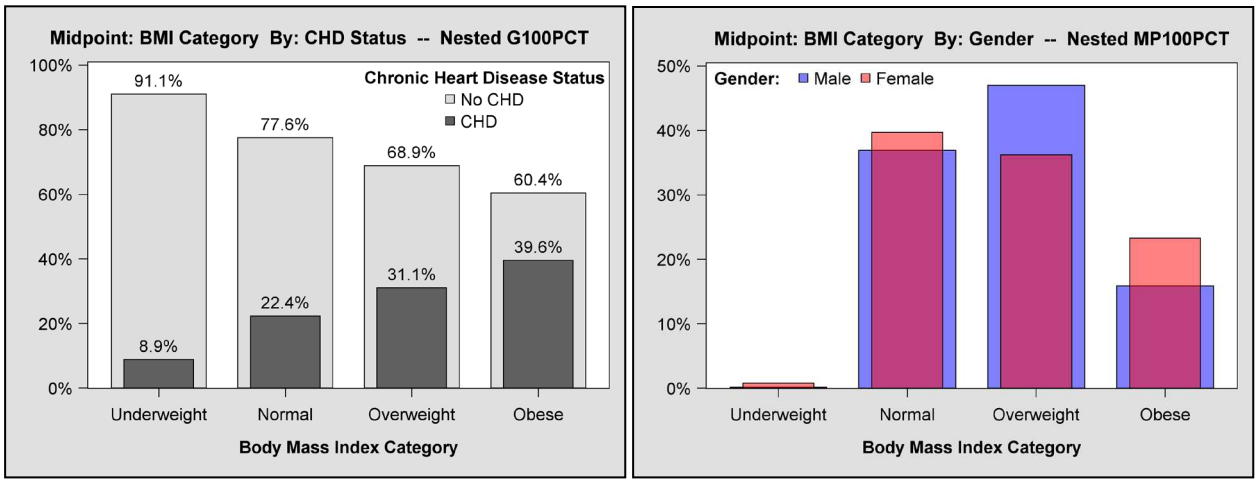

Figure 15. The nested bar chart solves problems commonly associated with stacked bar charts. In addition, transparency is used to increase visibility in the right-side chart where bar height relationships are inconsistent. Nested bar charts with their shortened distances increase the viewer's ability to make both ***between-category*** and ***within-group*** comparisons.

An abbreviated code listing for the STATGRAPH template used for generating the right-side bar chart in Figure 15 is listed below:

```
proc template;
    define statgraph MYBCHART;
      begingraph;
      layout overlay /
      ❶   CYCLEATTRS=TRUE
          xaxisopts=( ... )
          yaxisopts=( ... );
      ❷   BARCHART Y=M_mp100pct X=BMI_MPDesc /
      ❸      stat=SUM NAME="Males" LEGENDLABEL="Males"
             barwidth=0.75 barlabel=FALSE
      ❹      fillattrs=(TRANSPARENCY=0.5);
      ❷   BARCHART Y=F_mp100pct X=BMI_MPDesc /
      ❸      stat=sum NAME="Females" LEGENDLABEL="Females"
             barwidth=0.55 barlabel=FALSE
      ❹      fillattrs=(TRANSPARENCY=0.5);
      ❺   DISCRETELEGEND "Males" "Females" / ...;
      endlayout;
    endgraph;
  end;
run;
```

❶ Bar fill colors are assigned by statement iteration rather than by group affiliation.
❷ Two BARCHART statements replace the GROUP= option: one for males and a second for females.
❸ Separate NAME= options are used in the two BARCHART statements: again one for males and the other for females.
❹ Transparency is set at the half-way mark (0.5) in both BARCHART statements.
❺ Here is where the DISCRETELEGEND statement picks up the values in ❸ for the two NAME= options.

CREATING ENHANCED GROUP BAR CHARTS IN GTL

The enhanced bar charts in this section are created by adding separate graphics statements to LAYOUT OVERLAY in a STATGRAPH template. We saw an example of this type of enhancement when error bars were added to the basic bar chart in Figure 9 with a SCATTERPLOT statement. In Figure 16 the inside percent labels are again added via scatter plot, and the Figure 17 graph is enhanced with the addition of three SCATTERPLOT statements plus DRAWTEXT and DRAWRECTANGLE statements that work like SAS/GRAPH's ANNOTATE in version 9.3 GTL.

COMBINING FREQUENCIES AND PERCENTS IN A STACKED BAR CHART

In Figure 16 below the outside bar labels that reflect Y-Axis values are assigned conventionally in a BARCHART statement. Inside bar labels are added via a SCATTERPLOT statement with Y-coordinates that are adjusted with an EVAL function. Underweight and normal BMI categories have been combined into "Under_Normal" in Figure 16 so that all inside bar labels will be fully visible.

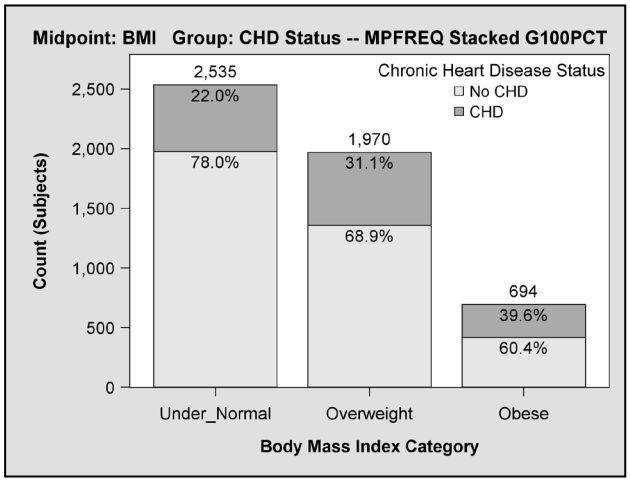

Figure 16. This stacked bar chart works, because bar heights determined by COUNT vary and segment labels sum to 100%.

Below is a printout of the input data set along with an abbreviated listing of the STATGRAPH template for Figure 16:

BMI3_MPDesc	CHD_ GrpDesc	MPfreq	Grp❶ Freq	Adj Grp❷ Freq	g100Pct❸
Under_Normal	No CHD	2535	1977	1977	78.0%
Under_Normal	CHD	2535	558	2535	22.0%
Overweight	No CHD	1970	1357	1357	68.9%
Overweight	CHD	1970	613	1970	31.1%
Obese	No CHD	694	419	419	60.4%
Obese	CHD	694	275	694	39.6%

```
proc template;
  define statgraph MYBCHART;
    begingraph;
     layout overlay / xaxisopts=( ... )
                      yaxisopts=( ...
❶                       linearopts=(tickvalueformat=COMMA5.));
❶      BARCHART X=BMI3_MPDesc Y=GRPFREQ /
                      stat=SUM
                      BarLabel=TRUE barlabelattrs=(size=12pt)
❶                     barlabelformat=COMMA5.
                      group=CHD_GrpDesc name="barGroup"
❹                     groupdisplay=STACK
                      barwidth=0.80;
       DISCRETELEGEND "barGroup"  /...;
❷      SCATTERPLOT X=BMI3_MPDesc Y=eval(AdjGrpFreq - 90) /
❸                     markercharacter=G100PCT
❸                     markercharacterattrs=(size=12pt color=BLACK)
                      group=CHD_GrpDesc
❹                     groupdisplay=OVERLAY;
     endlayout;
    endgraph;
  end;
run;
```

❶ The Y-coordinate in the BARCHART statement is GRPFREQ. Therefore, Y-axis values and OUTSIDE bar labels are formatted with COMMA5.
❷ ADJGRPFREQ is the starting point when Y coordinates for INSIDE percents are calculated. They are reduced a little with an application of the EVAL option.
❸ A format is applied to G100PCT at the data step level, since MARKERCHARACTERATTRS= does not support formatting.
❹ GROUPDISPLAY=stack in the BARCHART statement is equivalent to GROUPDISPLAY=overlay in the SCATTERPLOT statement.

COMBINING FREQUENCIES, MEANS AND ERROR BARS IN A CLUSTER BAR CHART

The motivation for the graph in Figure 17 comes from an entry in the SAS Sample Library that shows how to make a grouped bar chart of percentages with overlaid error bars of subjects complaining of eye irritation **[Sample 39166, 2010].** While the appearance and major tasks performed in the two programs are similar, implementation is very different. A single LAYOUT OVERLAY statement is used to create the bar chart in Figure 17 whereas the graph generated from SAS Sample code uses a multiple panel format: LAYOUT DATALATTICE nested within a LAYOUT GRIDDED statement. In Figure 17 the row of frequencies (preceded by ***n:***) is created with a SCATTERPLOT statement whereas the corresponding variable ***N*** becomes X-axis tick values in the code from the SAS Sample library.

Genuine bar labels are also used in the code from the SAS Sample Library. In fact the BARLABEL option is advertized as a major feature in the source code header. For Figure 17, BARLABEL is set to FALSE, and means are written to bar INSIDEs with a second SCATTERPLOT statement. INSIDE is chosen over OUTSIDE so that bar labels and error bars don't compete for the same space. Both programs use a SCATTERPLOT statement to create the error bars. However, full error bars that block out some of the BARLABELs are created in the SAS Sample Library chart, whereas Figure 17 uses half-error bars to complete the same task.

You are encouraged to take a look at the source code in both listings. The entire program for Figure 17 is listed with comments in the appendix. See if you can figure out what is going on. If you need assistance, contact the author.

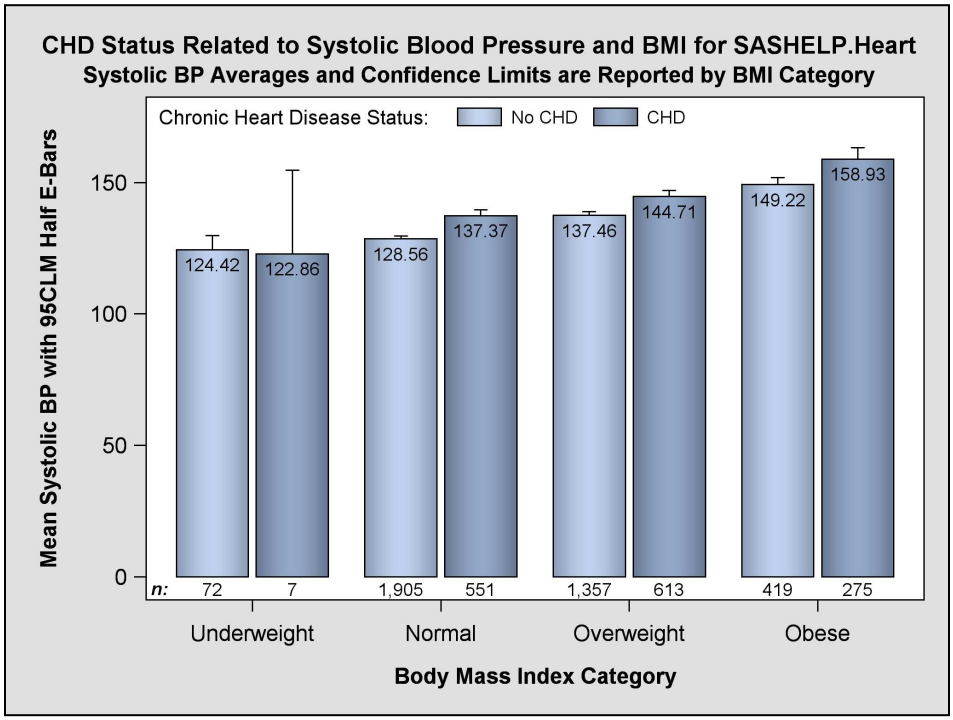

Figure 17. An annotated repeating group bar chart created with LAYOUT OVERLAY, BARCHART statements plus three SCATTERPLOT statements for individual bar counts and means. Now the viewer can see the connection between bar count and the length of the corresponding half error bar.

SUMMARY AND CONCLUSIONS

Step-by-step instructions have been provided for building basic and group bar charts that display frequencies, sums, percents and means with associated confidence intervals. As you saw from the beginning, percent bar charts do not work as intended. A switch from raw to summary data where picture formats for percents can be applied provided a solution to the problem.

Group charts were further subdivided into repeating and non-repeating categories. For the repeating category, summary data was again the format of choice. In addition to working satisfactorily with percents, summary data provides

a ready check for what goes on in the more complex repeating group bar chart. Examples of *stacked*, *cluster*, and *nested* repeating group charts were also presented.

Besides learning what impact the BARCHART statement and its associated options have on graphics output, an effort has been made to show how the STYLE template, ODS Destination and Graphics statements, axis options, legends and even other GTL graphics statements are incorporated into chart building.

Probably the greatest strength of the BARCHART statement over SAS/GRAPH's GCHART procedure is that it works with a generic *discrete* X-axis and not GCHART's specialized *midpoint* and *group* axes. With both the discrete axis and the GROUPDISPLAY option, new in 9.3 SAS, the programmer is in an excellent position to create intricate graphs with relative ease. As you will see in the complete listing for the Figure 17, repeating group bar charts and scatter plots can now be combined to produce a single graph, simply because the SCATTERPLOT statement **also** supports a GROUPDISPLAY option. Also while code is not presented for Figure 18 below, you can see that bar charts can be aligned with box plots and vertical strip plots with the addition of the DISCRETEOFFSET option so that now the programmer has access to the entire data display region while working within the confines of a discrete axis. For additional information about this graph, see **[Watts and Derby, 2012].**

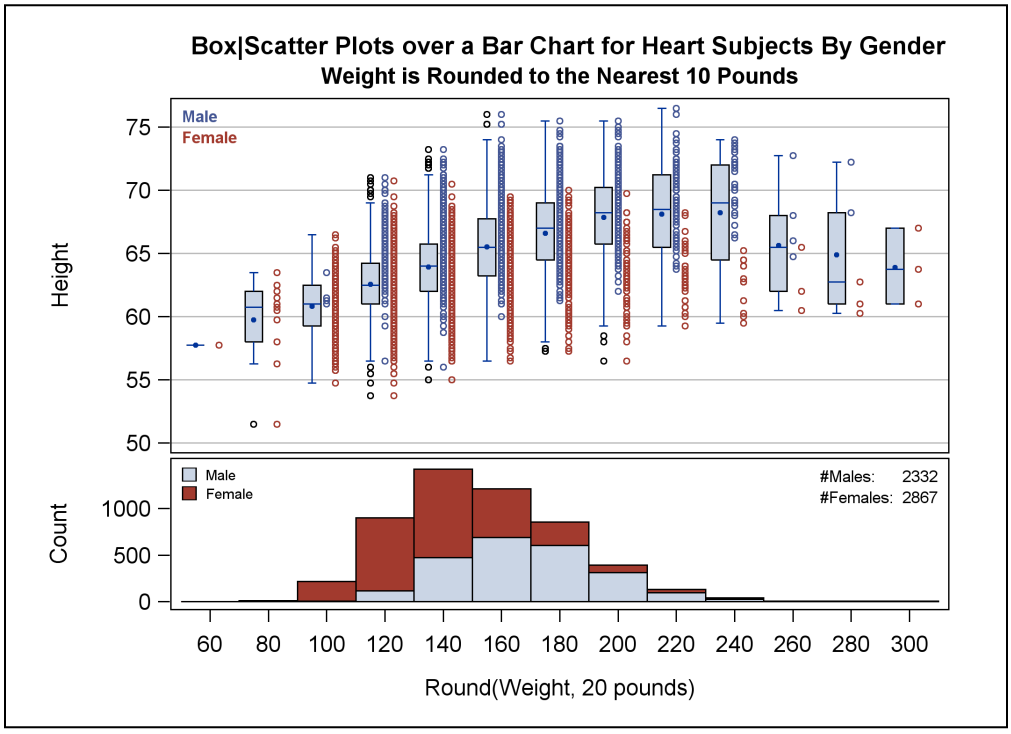

Figure 18. The common X axis in the two-panel display that uses a LATTICE layout is discrete. That means all unique values for ROUNDWEIGHT are fully enumerated along the horizontal axis. Regions between the major ticks for the strip plots and box plots can be accessed by setting the DISCRETEOFFSET option to a non-zero number ranging from -0.5 to +0.5 "where 0.5 represents half the distance between axis ticks" **[SAS Institute, 2011a, p. 447**]. In other words, -0.5 and +0.5 are located at the bar boundaries.

REFERENCES

Cleveland, W. S. (1994). *The Elements of Graphing Data, revised edn*. Summit, NJ: Hobart Press,

Bar chart: From Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Bar_chart. A definition and history of the bar chart is provided. Access Date: June 26, 2013.

Kuhfeld, W. F. (2010). *Statistical Graphics in SAS®: An Introduction to the Graph Template Language and the Statistical Graphics Procedures*. Cary, NC: SAS Institute Inc.

Mantange, S. and D. Heath. (2011). *Statistical Graphics Procedures by Example: Effective Graphs Using SAS®*. Cary, NC: SAS Institute Inc.

Robbins, N. B. (2005). *Creating More Effective Graphs*. Hoboken, NJ: John Wiley & Sons, Inc.

Sample 39166: Distribution of eye irritation. (2010). http://support.sas.com/kb/39/166.html. This sample uses the Graph Template Language (GTL) to produce a grouped bar chart with overlaid error bars. Access Date: June 26, 2013.

SAS Institute. (2004). *SAS/GRAPH® Reference, Volume 2*. Cary NC: SAS Institute Inc.

SAS Institute. (2011a). *SAS® 9.3 Graph Template Language Reference*. Cary, NC: SAS Institute, Inc.

SAS Institute. (2011b). *SAS® 9.3 ODS Graphics Procedures Guide*. Cary, NC: SAS Institute, Inc.

StatLib. *Baseball Data from Datasets Archive*, http://lib.stat.cmu.edu/datasets/baseball.data. This was the 1988 ASA Graphics Section Poster Session data set. The section organizer was Lorraine Denby. Access Date: June 26, 2013.

Tufte, E. R. (2001). *The Visual Display of Quantitative Information: Second Edition*. Cheshire, CT: Graphics Press.

Watts, P. (2007a). *Building a Better Bar Chart with SAS/Graph® Software.* Proceedings of the 20th Annual Northeast SAS Users Group Conference. Baltimore, MD, paper #NP16. http://www.nesug.org/proceedings/nesug07/np/np16.pdf

Watts, P. (2007b). *Charting the Basics with PROC GCHART*. Proceedings of the 20th Annual Northeast SAS Users Group Conference. Baltimore, MD, paper #FF17. http://www.nesug.org/proceedings/nesug07/ff/ff17.pdf

Watts, P. (2008). *Sensitivity Training for Building Better Bar Charts with SAS/GRAPH® Software.* Proceedings of the 21st Annual Northeast Sas Users Group Conference. Pittsburgh, PA, paper #HW07. http://www.nesug.org/proceedings/nesug08/hw/hw07.pdf

Watts, P. and N. Derby. (2012). *Using SAS® GTL with 9.3 Updates to Visualize Data When there is Too Much of It to Visualize.* Proceedings of the 20th Annual Northeast SAS Users Group Conference. Baltimore, MD, paper #PO14. http://www.nesug.org/proceedings/nesug12/po/po14.pdf

ACKNOWLEDGEMENTS

I would like to thank Arthur Li for encouraging me to make my first presentation at MWSUG and Nate Derby, President of Stakana Analytics, for his steadfast support of my efforts in the field of statistical graphics. I am deeply honored to be part of Nate's team at Stakana Analytics. I also want to thank my husband, Samuel Litwin, who literally has gone the extra mile to make it possible for me to attend MWSUG.

TRADEMARK CITATION

CONTACT INFORMATION

Comments and questions are valued and encouraged. Workshop source code and handouts will be made available at http://www.PerryWatts.org.

Perry Watts
Stakana Analytics
pwatts@stakana.com

APPENDIX: COMPLETE SOURCE CODE FOR FIGURE 17 – WITH QUESTIONS

```
/*  ------------------------------------------------------------------------
    Program  :  Figure17.sas

    Author   :  Perry Watts
    Purpose  :  Create a MEAN repeating group bar chart with confidence limits.
    Output   :  Fig17.png

    Questions:
    1) what does fillattrs=(transparency=0.3) do in the FIRST BARCHART statement?
    2) Why was this option set?
    3) Also in the first BARCHART statement, "DATASKIN=PRESSED" replaces the
       SAS Sample version, "SKIN=MODERN".  The replacement was made, because
       the term "SKIN" does not appear anywhere in 9.2 or 9.3 GTL manuals.
       The same is true for "MODERN". However when the change to is made to
       "DATASKIN=PRESSED" outlines cannot not drawn around chart bars or in
       the legend. (See manual). Your task is to find the statements that
       create the outlines and to figure out how they work.
    ------------------------------------------------------------------------  */
   %let mypath=C:\MWSUG2013;

   ods path work.templat(update) sashelp.tmplmst(read);
   ods listing gpath="&myPath";

   data heartSysDat;
    length BMI_MPDesc $12;
    length CHD_GrpDesc $6;
    length grpFreq $5;
    input BMI_MPDesc CHD_GrpDesc & GrpFreq YforN meanRvar_Sys lclm uclm;
   cards;
    Underweight    No CHD          72     -5      124.42     119.00    129.83
    Underweight    CHD              7     -5      122.86      91.10    154.62
    Normal         No CHD       1,905     -5      128.56     127.58    129.54
    Normal         CHD            551     -5      137.37     135.17    139.58
    Overweight     No CHD       1,357     -5      137.46     136.12    138.81
    Overweight     CHD            613     -5      144.71     142.55    146.86
    Obese          No CHD         419     -5      149.22     146.50    151.93
    Obese          CHD            275     -5      158.93     154.62    163.25
  run;

  /* STYLE TEMPLATE */
   proc template;
     define Style myStyle;
      parent = styles.default;
       CLASS GraphFonts
         "Fonts used in graph styles" /
         'GraphTitleFont' = (", ",12pt,bold)
         'GraphLabelFont' = (", ",11pt)
         'GraphValueFont' = (", ",11pt);
       CLASS graphBorderLines /
         linethickness = 0px;
       CLASS graphdata1 /
         contrastColor=black
         color=cx8CaEdF;     /* No CHD   -- LIGHT BLUE */
       CLASS graphdata2 /
         contrastcolor=black
         color=cx42659C;     /* With CHD -- DARKER BLUE */
       CLASS graphAxisLines /
         contrastcolor=black;
       CLASS graphwalls /
         contrastcolor=black;
     end;
    run;
    ods listing style=myStyle image_dpi=300;

  /* STATGRAPH TEMPLATE */
    proc template;
      define statgraph myBChart;
      begingraph;
        entrytitle "CHD Status Related to Systolic Blood Pressure and BMI for SASHELP.Heart";
        entrytitle "Systolic BP Averages and Confidence Limits are Reported by BMI Category"
                        / textattrs=(size=11pt);
          layout overlay / xaxisopts=( label="Body Mass Index Category"
                           labelAttrs=(weight=bold))
                         yaxisopts=( label="Mean Systolic BP with 95CLM Half E-Bars"
```

```
                        labelattrs=(weight=bold)
                        offsetmin=0.02 offsetmax=0.1);
/* FOR FILLED BARS IN THE BAR CHART */
          barchart X=BMI_MPDesc Y=MeanRvar_Sys / stat=sum BarLabel=FALSE
                        group=CHD_GrpDesc name="barGroup"
                        groupdisplay=cluster dataskin=pressed
                        display=(fill)
                        barwidth=0.9 fillattrs=(transparency=0.3);
/* FOR BAR OUTLINES*/
          barchart X=BMI_MPDesc Y=MeanRvar_Sys / stat=sum BarLabel=FALSE
                        group=CHD_GrpDesc
                        groupdisplay=CLUSTER
                        display=(OUTLINE)
                        outlineattrs=(color=BLACK thickness=1PX)
                        barwidth=0.9;
          discreteLegend "barGroup"  / across=2 autoalign=(TOPLEFT) titleattrs=(size=10PT)
                        valueattrs=(size=9PT)
                        border=FALSE location=INSIDE
                        title="Chronic Heart Disease Status:";
/* OUTSIDE LABELS ARE MOVED INSIDE */
          scatterplot X=BMI_MPDesc Y=EVAL(MeanRvar_Sys - 6)/
                        markercharacterattrs=(size=9PT color=BLACK)
                        markercharacter=MeanRvar_Sys
                        group=CHD_GrpDesc groupdisplay=CLUSTER
                        clusterwidth=0.85;
/* ERRORBAR: WHY IS SIZE=0? */
          scatterplot X=BMI_MPDesc Y=MeanRvar_Sys / markerattrs=(size=0)
                        errorbarattrs=(thickness=1)
                        yerrorlower=meanRvar_Sys yerrorupper=uclm
                        group=CHD_GrpDesc groupdisplay=CLUSTER
                        clusterwidth=0.85;
/* *****
   DRAW THE LETTER 'N' WITH DRAWTEXT, AN ANNOTATE STATEMENT IN GTL.
   XSPACE IS SET TO WALLPERCENT IN DRAWTEXT, SINCE THE X-AXIS IS DISCRETE.
   THE X-COORD HERE IS 2% TO THE RIGHT OF THE WALL. (IT COULD BE 2.0791 - A REAL NUMBER).
   YSPACE IS SET TO DATAVALUE, BECAUSE THE Y AXIS IS LINEAR AND CAN SUPPORT FRACTIONS.
***** */
          drawtext textattrs=(size=9pt style=ITALIC weight=BOLD) "n:" /
                        width=5 widthunit=PERCENT
                        xspace=WALLPERCENT yspace=DATAVALUE
                        x=2 y=-4.5 justify=CENTER;
/* ****
   PLOT ALL THE VALUES FOR 'N' WITH A SCATTERPLOT STATEMENT.
   TO GET COMMA FORMATTED OUTPUT FOR GRPFREQ, YOU MUST FORMAT AT THE DATA STEP LEVEL.
   (SEE PAPER). THE TRICK FOR THIS SCATTERPLOT CONCERNS THE Y PARAMETER WHERE Y=YFORN=-5.
   POINTS WITH VALUES EQUAL TO MINUS 5 BECOME VISIBLE BY EXTENDING OFFSET MIN FROM
   0 TO 0.02. IN OTHER WORDS, DATA CAN BE PLOTTED BY VALUE IN THE OFFSET REGION.
***** */
          scatterplot y=YforN  x=BMI_MPDesc /
                        markercharacterattrs=(size=9PT color=BLACK)
                        markercharacter=GRPFreq
                        group=CHD_GrpDesc groupdisplay=CLUSTER
                        clusterwidth=0.85;
/* *****
   DRAWRECTANGLE PLACES AN OUTLINE AROUND THE COLOR SAMPLES IN THE LEGEND.
   WHY WOULD ANYONE DO THIS?
   THE ANCHOR=OPTION PLAYS AN IMPORTANT ROLE. WHAT IS IT? (SEE MANUAL)
***** */
          drawrectangle x=40 y=94.25 width=5.6 height=3.5 /
                        anchor=BOTTOMLEFT drawspace=WALLPERCENT
                        display=(OUTLINE) outlineattrs=(color=BLACK thickness=1PX);
          drawrectangle x=57.5 y=94.25 width=5.6 height=3.5 /
                        anchor=BOTTOMLEFT drawspace=WALLPERCENT
                        display=(OUTLINE) outlineattrs=(color=BLACK thickness=1PX);
        endlayout;
      endgraph;
    end;
  run;

ods graphics on / reset=index imageName="Fig17" antialiasmax=10000 imagefmt=png;
 proc sgrender data=work.heartSysDat template=myBchart;
 run;
 quit;
ods graphics off;
```