

Building a Better Bar Chart with SAS/GRAPH® Software

Perry Watts, Independent Consultant

INTRODUCTION

The GCHART procedure in SAS/GRAPH software works very well for building a basic bar chart with a limited number of bars, brief text annotations, standard statistics, and hierarchical groupings. But what if you can't use the defaults to specify the widths of the bars in your chart? Or how do you plot uneven error bars on a bar chart? Or get a bar chart with both error bars and a legend? Or if you want to look at OUTSIDE or INSIDE annotation on the bars, how can you guarantee it won't disappear when bar width boundaries are exceeded? Also how do you make a non-hierarchical bar chart where subordinate membership is not repeated across groups?

Answers to all these questions and more are to be found in the pages that follow. An easy WIDTH= fix plus a more involved solution by formula are provided to optimize the size of bar widths in a bar chart. Using summary rather than raw data also enables any type of statistic available in SAS® software to be charted. Finally an application of ANNOTATE for OUTSIDE and INSIDE labels overcomes bar boundary issues with special commands built just for PROC GCHART.

This paper is geared to intermediate or advanced SAS/GRAPH users who are familiar with GOPTIONS and ANNOTATE plus the AXIS, PATTERN and LEGEND statements. Familiarity with the SAS macro language is also helpful but not essential. For review, see listings [1] - [4] in the references section. The companion NESUG paper, *Charting the Basics with PROC GCHART* [2] also provides examples of midpoint, subgroup and group charts along with percent, mean, sum and default frequency charts. Given a possible review of the basics, you should come away from this tutorial better equipped to construct customized bar charts with ease.

THE SAS ENVIRONMENT

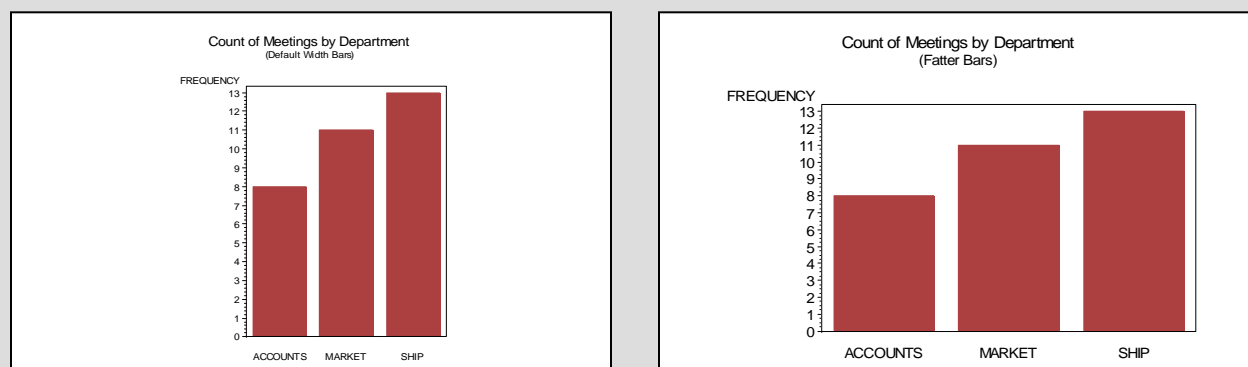
The code that supports the concepts described in this paper has been fully tested in Version 9.1.3 SAS software. Therefore, an accurate translation to earlier versions of the software is not guaranteed. In addition, all graphics output has been written to EMF files. For additional information about the EMF (Enhanced Windows Metafile) device driver, see *TS-DOC:TS-674* fully cited in the reference section [9].

BAR WIDTHS IN PROC GCHART

A bar chart unlike a scatter plot is a one-dimensional structure. Information is conveyed exclusively by the length of the bars along a response axis. This means that bar widths can be changed without altering the message conveyed by the graph. Bar widths in Figure 1 are changed with the WIDTH option:

```
proc gchart data=N07.meetings;  
  title2 "(Fatter Bars)";  
  vbar dept/width=15;  
run;
```

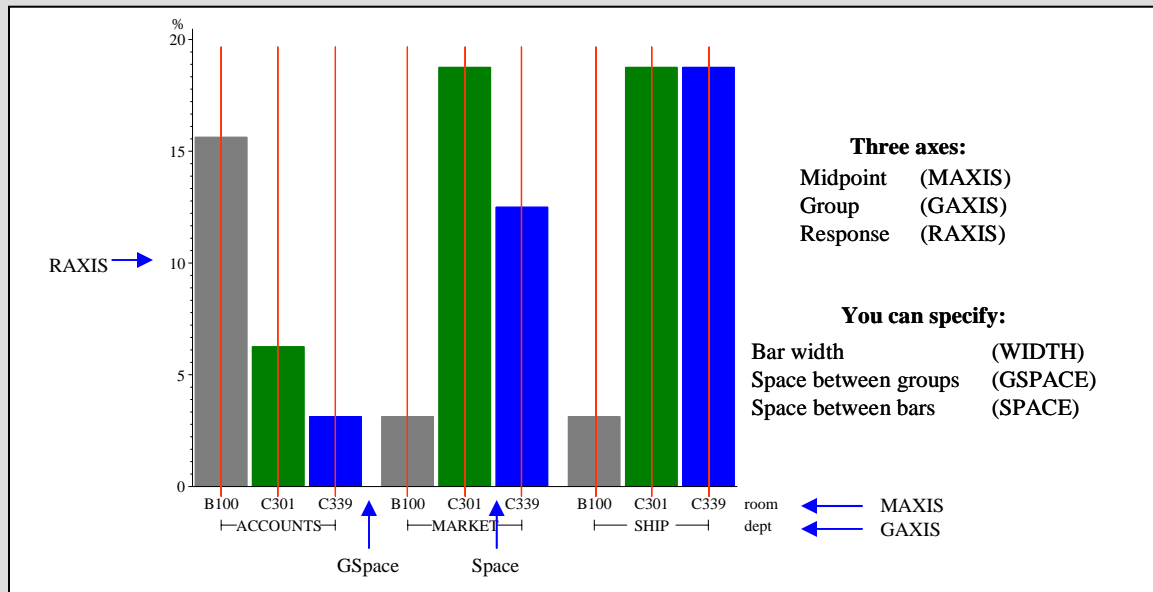
Figure 1. Two bar charts convey exactly the same information, because the bar widths have no intrinsic meaning.



Even though bar width is not a significant attribute in a bar chart, the WIDTH option provides insight into how PROC GCHART actually works. A midpoint (MAXIS) axis for categorical data replaces the horizontal axis found in

the GPLOT procedure. Unlike GPLOT there are no x-coordinates in GCHART to accommodate continuous data. Instead, the only addressable coordinates in PROC GCHART lie along the red vertical lines shown in Figure 2. Even the coordinates for the optional group axis (GAXIS) are not accessible to the programmer. This arrangement limits access to the graphic structure, but it makes it possible to change bar widths without having to know their exact location.

Figure 2. GCHART replaces the horizontal axis in GPLOT with a midpoint axis. Default locations for the joint midpoint and group axes labels, *room* and *dept*, are also highlighted. Programmers have access only to coordinates along the red lines. The data come from *The How-To Book for SAS/GRAPH® Software* by Thomas Miron [3]. Copyright 1995, SAS Institute Inc., Cary, NC, USA. All Rights Reserved. Reproduced with permission of SAS Institute Inc., Cary, NC



Difficulties with Default Widths:

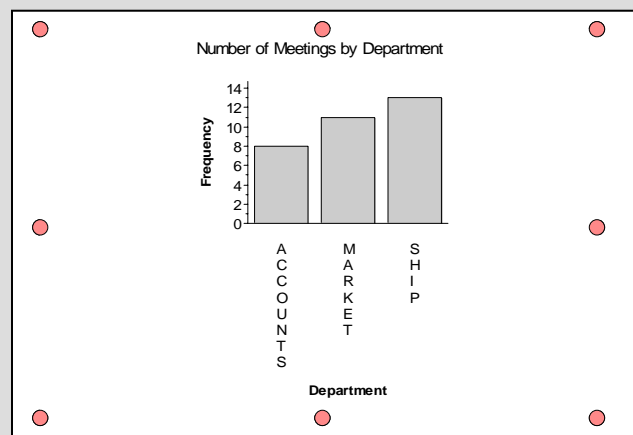
It is the programmer's responsibility to coordinate font size and bar width in PROC GCHART. If the bar widths are too narrow or the text size too large, letters are arranged vertically in the display. The result is an unappealing space-wasting graph. Even bar charts with few bars are vulnerable to this outcome as demonstrated in Figure 3.

Figure 3. Bar widths are set by default. Highlighted selection handles show that larger text could easily be accommodated.

```
pattern1 color=grayCC;
axis1 order=(0 to 14 by 2) minor=(n=1);
label=(a=90 f="Arial/Bold" "Frequency");
axis2 label=(f="Arial/Bold" "Department");
title1 'Number of Meetings by Department';

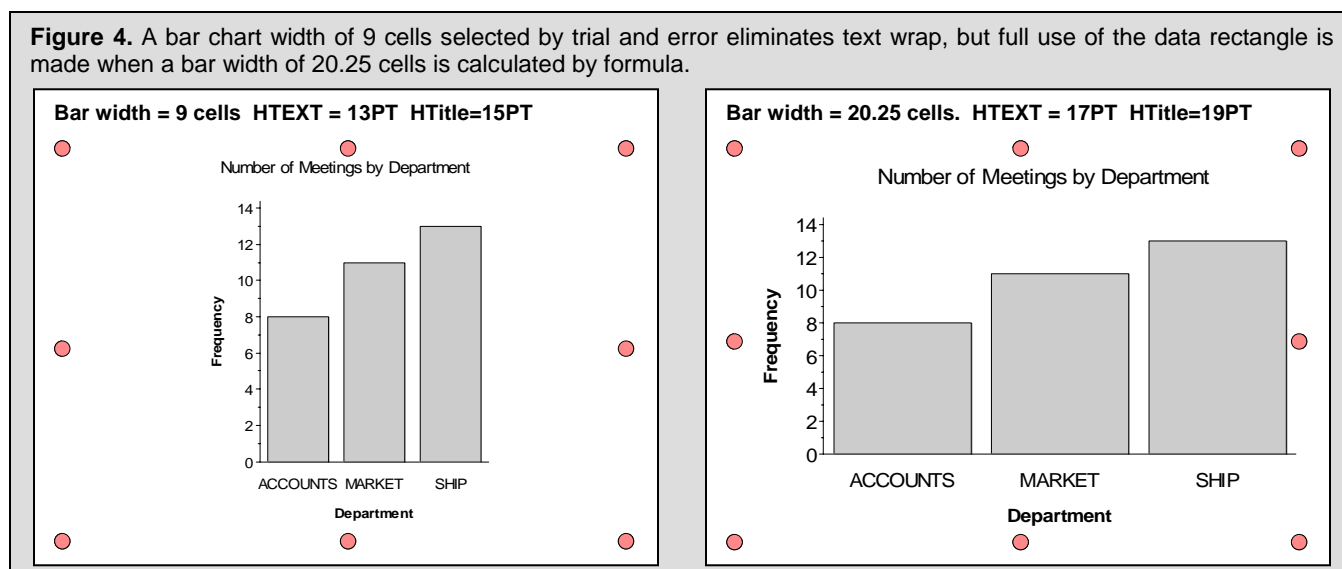
filename chrt 'c:\N07\BetterBarChart\Paper\EMF\chrt3.emf';
goptions htext=13pt htitle=15pt gsfname=chrt;

/* NO WIDTH= OPTION STATED BELOW */
proc gchart data=N07.meetings;
vbar dept /
outside=freq raxis=axis1 maxis=axis2
outline=black woutline=1;
run;
quit;
```



Bars have to be widened or font size has to be reduced in order to eliminate text-wrap from the graph in Figure 4. Either alternative requires trial and error, but legibility is improved if bar widths are increased to accommodate larger text. Setting the width to 9 cells solves the problem for the first graph in Figure 4. However, the graph still needs to be manually cropped to optimize display. In the graph to the right, a width of 20.25 cells has been calculated by formula. Now the graph takes up the entire data rectangle and there is no need for cropping.

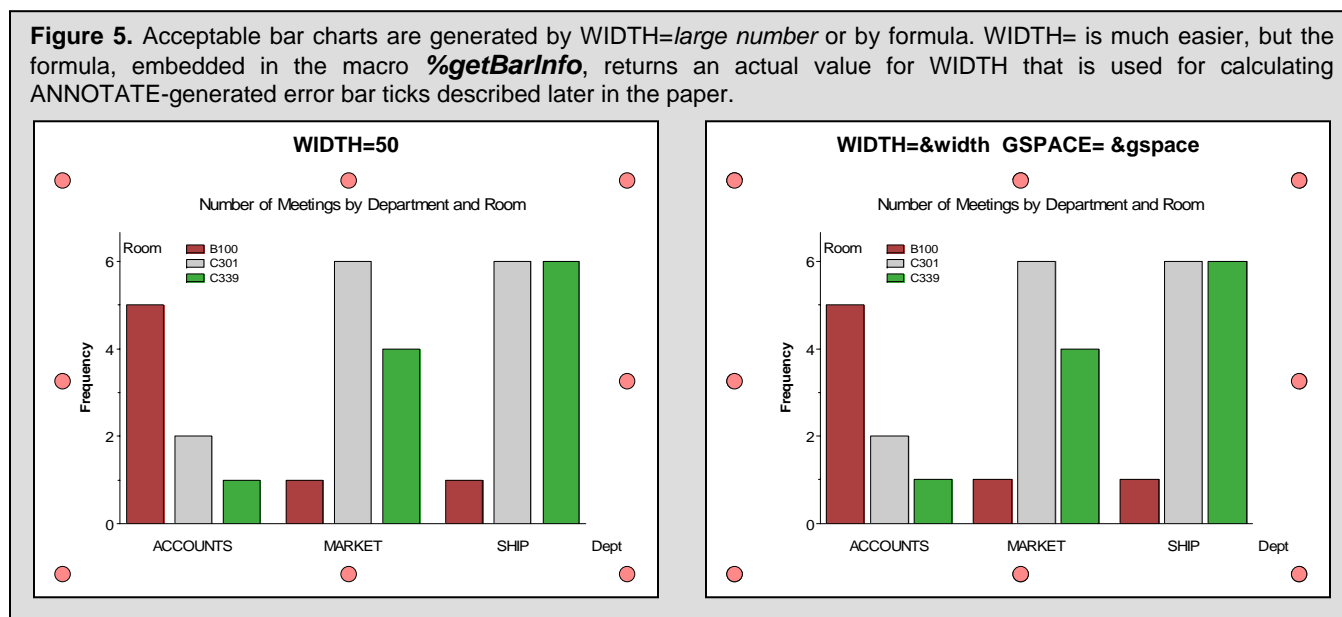
Figure 4. A bar chart width of 9 cells selected by trial and error eliminates text wrap, but full use of the data rectangle is made when a bar width of 20.25 cells is calculated by formula.



Use WIDTH=*large number* to generate a Bar Chart that Fills the Entire Data Rectangle:

Even when nine bars are in a chart, reliance on the default width short-changes the graphics display. If customized error bars described later in the paper are *not* needed, then setting WIDTH= to an arbitrarily high number will generate an acceptable bar chart. Two methods are used to create the charts displayed in Figure 5.

Figure 5. Acceptable bar charts are generated by WIDTH=*large number* or by formula. WIDTH= is much easier, but the formula, embedded in the macro **%getBarInfo**, returns an actual value for WIDTH that is used for calculating ANNOTATE-generated error bar ticks described later in the paper.



%getBarInfo Calculates Optimal Bar Widths in GCHART:

The **%getBarInfo** macro function is included in the appendix and on the NESUG CD. For a complete description of user-defined macro functions, see Carpenter [2], pp. 178-182. Parameters to the macro include:

Xorigin	in Percent
SpaceSz	space between bars (in percent)
GspaceSz	space between groups (in percent)
NGrps	Number of groups (If midpoint chart, enter 1)
NBars	n groups X m midpoints for a midpoint, subgroup or hierarchical bar chart
LCOLFrac	Multiply LCOLS from PROC GDEVICE by 0.01 (e.g. EMF: 75 X 0.01 = 0.75)

Since bar widths and spaces only come in cells, a conversion has to be made from input percent for SPACESZ and GSPACESZ to output cell for WIDTH and GSPACE used in GCHART. Unfortunately, the size of a cell is de-

vice dependent, so to get the conversion right, the value for LCOLS has to be supplied by the GDEVICE procedure. Here is how to get LCOLS for the EMF driver:

```
proc gdevice nofs;
  list EMF;
run; quit;
```

LCOLS is then multiplied by 0.01 and the result is sent to %getBarInfo via parameter, LCOLFRAC.

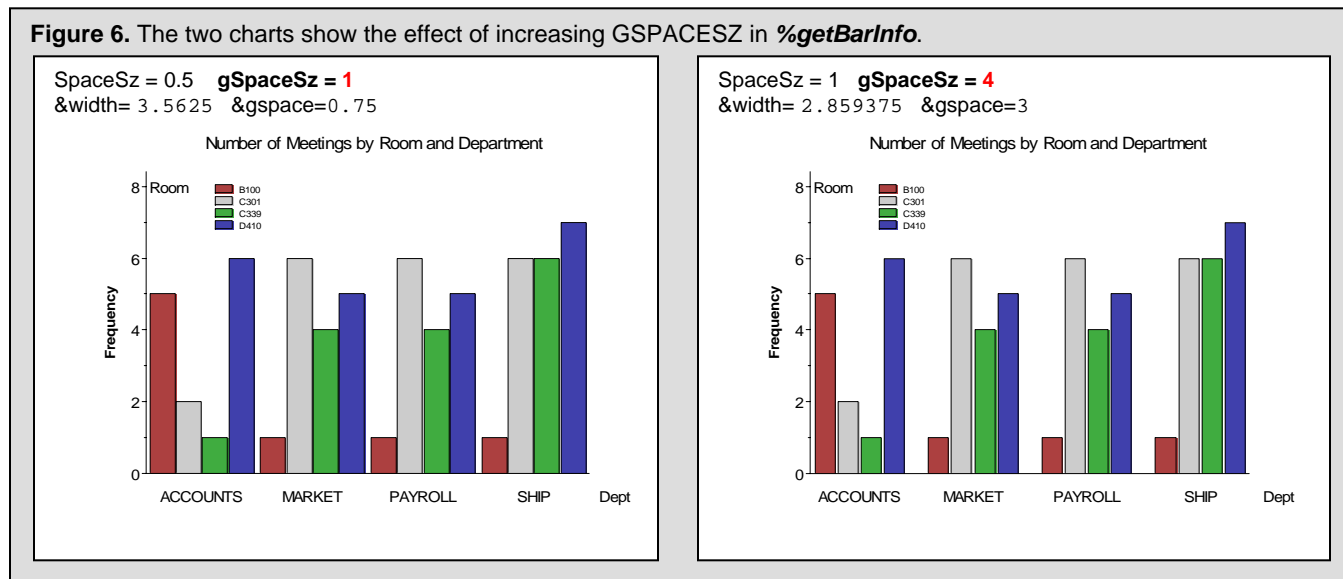
Selected assignment statements from the macro are translated into pseudo-code below. Parameter names are italicized:

```
AxisLength = 100 - Xorigin
NumGrpSpaces = nGrps-1
NumSpaces = nBars - numGrpspaces - 1
AxisLength = AxisLength - (numSpaces X SpaceSz) - (numGrpSpaces X GSpaceSz)
WidthPct = AxisLength / nBars
WidthCells=LcolFrac X WidthPct
GspaceCells= LcolFrac X GspaceSz
```

If **%getBarInfo** is embedded in a larger macro then NGRPS and NBARS can be calculated by processing the input data. This way optimal bar widths can be assigned dynamically.

Instructions for assigning the macro output string as arguments to WIDTH and GSPACE options in GCHART can be found in the appendix. What needs to be emphasized here is that a value for SPACE is not returned from **%getBarInfo**, because it is not used in PROC GCHART. By leaving SPACE undefined, SAS can adjust its value internally to generate a balanced chart with the same offset sizes.

To validate **%getBarInfo**, two hierarchical group bar charts with 16 bars are displayed in Figure 6 with different values for SPACE and GSPACE. While values for &WIDTH and &GSPACE do not reflect the ratios between input parameters, SPACESZ and GSPACESZ, spaces between the groups are still larger in the second graph from Figure 6.



The SAS code for the first chart in Figure 6 is listed below. Commands are grayed out that aren't pertinent to the current discussion. Highlighted items are elaborated in the bullet-list that follows.

```
%let result =
  %getBarInfo(Xorigin=15, SpaceSz=0.5, GspaceSz=1, Ngrps=4, Nbars=16, LCOLfrac=0.75);
%let width=%scan(&result,1,':');
%let gspace=%scan(&result,2,':');

pattern1 color=CXAC4040; pattern2 color=graycc;
pattern3 color=CX40AC40; pattern4 color=CX4040AC;
```

```

legend1
  across=1 shape=bar(3,2) label=("Room" h=11 pt position=left j=c)
  position=(top inside left) mode=share value=(h=9pt 'B100' 'C301' 'C339' 'D410');

axis1
  label=(a=90 f="Arial/Bold" "Frequency") order=(0 to 8 by 2) minor=(n=1)
  origin=(15pct, 10pct) offset=(,4pct);
axis2
  label=none value=none origin=(15pct, 10pct);
axis3 label=("Dept") offset=(0.5 pct, 0.5 pct);

title1 move=(+9pct,+0pct) 'Number of Meetings by Room and Department';

filename chrt 'c:\N07\BetterBarChart\Paper\EMF\chrt6.emf';
goptions htext=13pt htitle=15pt gsfname=chrt;

proc gchart data=N07.moreMeetings;
  vbar room / width=&width gspace=&gspace
    type=FREQ Group=Dept patternID=subgroup noframe
    legend=legend1 subgroup=room
    raxis=axis1 maxis=axis2 gaxis=axis3
    coutline=black woutline=1;
run; quit;

```

- **xorigin=15** parameter must agree with the Axes options **origin=(15pct,...)**. **NOTE!!** Do not use **ORIGIN=** or **OFFSET=** options in the **WIDTH=large number** solution.
- **offset = (0.5pct, 0.5pct)** There is no need inside **%getBarInfo** to allot space for offsets. SAS handles them automatically. The same is not true for axes lengths, however. Axes lengths have no impact on bar widths, and using the length option in an axis statement may cause GCHART to fail.
- **title1 move=(+9pct, +0pct)** is a relative move that centers the title over the data rectangle. (0.6 X 15 [Xorigin] = 9). Centering text over the data rectangle is not only aesthetically pleasing, it can be used to validate bar expansion. The first graph in Figure 5, for example, does not use the full data rectangle resulting in a title that is off-center.
- **width=&width gspace=&gspace** is where the output from **%getBarInfo** is used.

LEGENDS IN PROC GCHART

While **%getBarInfo** optimizes the placement of bars in a bar chart, legends can also be used to eliminate line-wrap from graphics output. Despite the requirement for a **SUBGROUP** option assignment, legends can be inserted into both *midpoint* and *group* charts as well as *subgroup* charts. In Figures 7-9 below, relevant GCHART options are listed for generating charts both with and without legends. Full source code is on the NESUG CD.

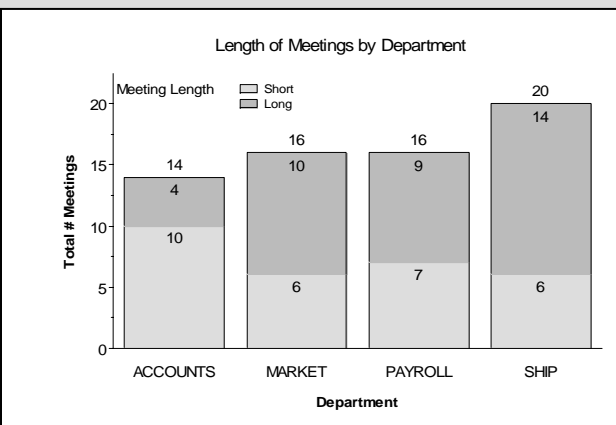
Figure7. Legends are generated by default in *Subgroup* charts.

With Legend

```

vbar Dept /width=&width type=FREQ
  inside=freq outside=freq
  subgroup=hrGrp legend=legend1
  noframe raxis=axis1 maxis=axis2 ... ;

```

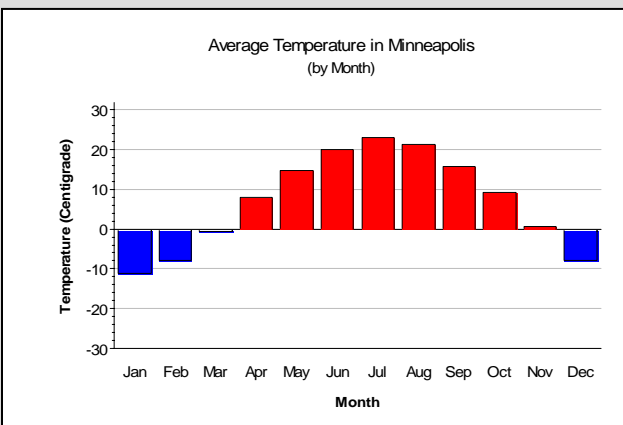


No Legend

```

vbar month / discrete noframe
  width=&width type=sum sumvar=TC
  subgroup=AboveZeroYvN nolegend
  raxis=axis1 maxis=axis2 ... ;

```



A genuine *subgroup* chart is only generated when *midpoint* and *subgroup* variables are different. The first chart in Figure 7 uses DEPT for the *midpoint* and HRGRP for the *subgroup* whereas MONTH and ABOVEZEROYVN play the same roles in the second chart.

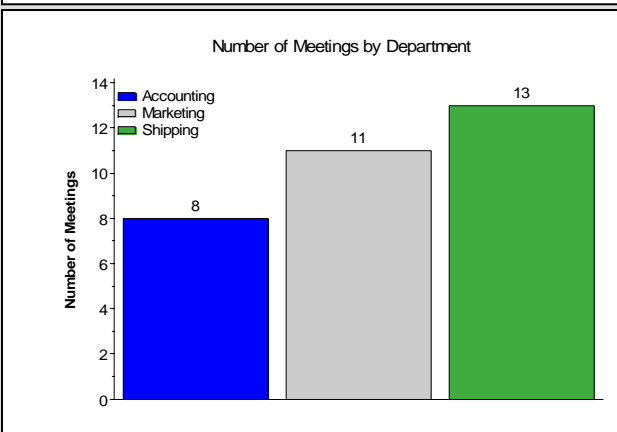
Typically a legend is needed for identifying bar segments in a *subgroup* chart. However, segmented bars don't appear in the second chart, because temperatures are only above or below zero. The superfluous default legend is hidden by NOLEGEND in the code. The code for this chart is adapted from Col2.sas in *Robert Allison's SAS/Graph Examples!* [10].

In Figures 8 and 9, *midpoint* and *group* charts are disguised as *subgroup* charts by assigning midpoint variables to the SUBGROUP option.

Figure8. A legend in a *midpoint* chart allows the response axis to be fully extended. To get different color bars in a midpoint chart without a legend, use the PATTERNID option.

With Legend

```
vbar dept / noframe
width=&width type=FREQ outside=FREQ
subgroup=dept legend=legend1
raxis=axis1 maxis=axis2 ... ;
```



No Legend

```
vbar dept / noframe
width=&width type=FREQ outside=FREQ
patternID=midpoint
raxis=axis1 maxis=axis2 ... ;
```

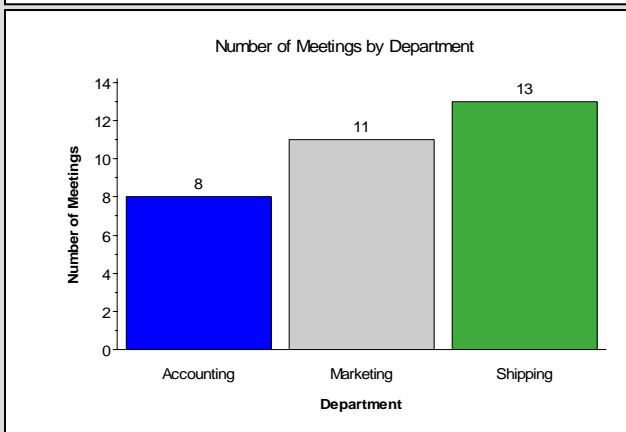
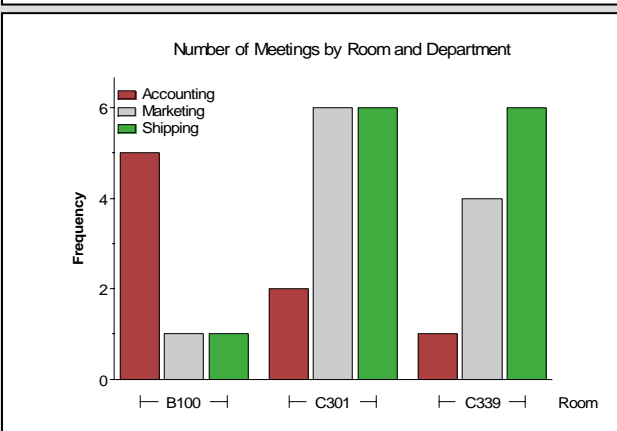


Figure9. The midpoint axis can be removed from display when a legend is used in a *group* chart. In the second chart, PATTERNID is set to GROUP, and line-wrap along the midpoint axis is avoided by invoking %getBarInfo.

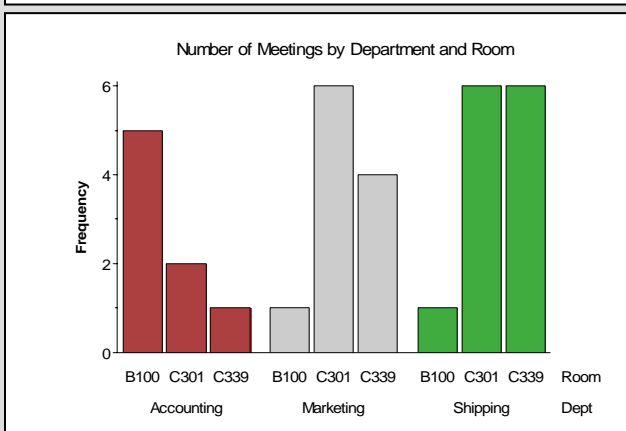
With Legend

```
vbar Dept / noframe
type=FREQ width=&width gspace=&gspace
subgroup=dept legend=legend1 Group=Room
raxis=axis1 maxis=axis2 gaxis=axis3 ... ;
```



No Legend

```
vbar Room /
type=FREQ width=&width gspace=&gspace
Group=Dept patternID=group noframe
raxis=axis1 maxis=axis2 gaxis=axis3 ... ;
```



Additional techniques for managing the placement of text in a graph are described in the companion NESUG paper [3], pp. 6-9.

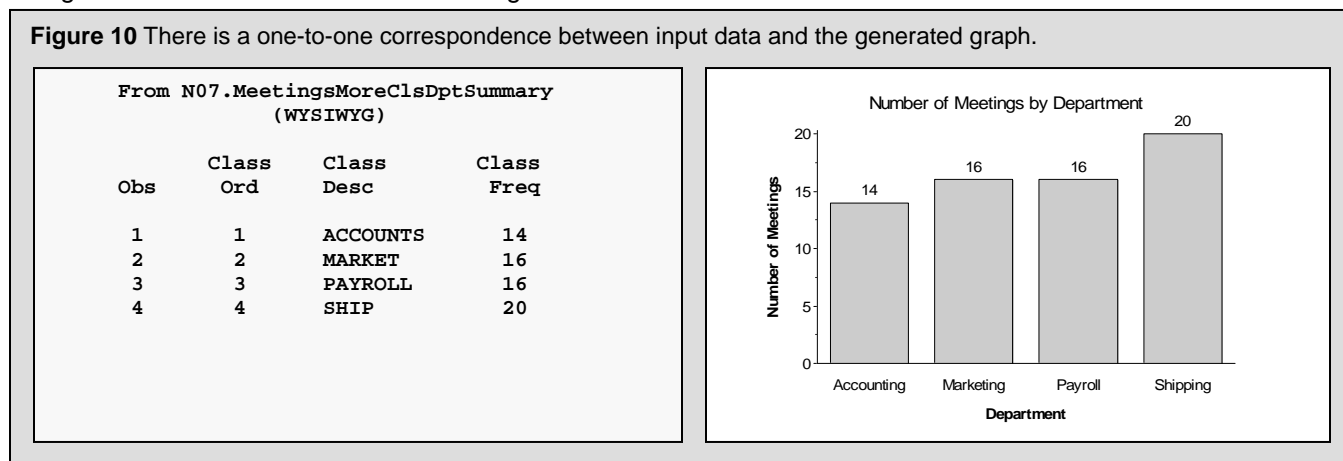
USING SUMMARY DATA AS INPUT TO PROC GCHART

The recommendation for using summary data comes from *Let Summary Sum and Tabulate Format* by Marianne Whitlock [5]. In her paper, Whitlock describes "a method for doing all calculating before using TABULATE, and using the resulting data set as input to TABULATE" [5], p.1. Just replace "TABULATE" with "GCHART" and the process is the same. Working with summarized data in PROC GCHART simplifies processing and increases the variety of the graphs that can be produced.

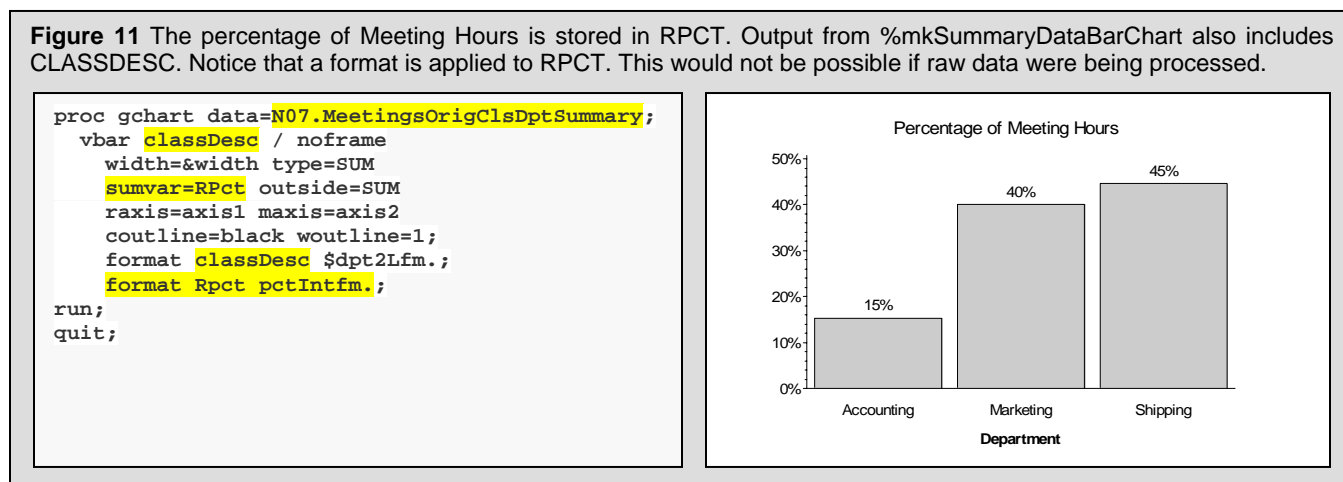
The **%mkSummaryDataBarChart** macro converts raw data with categorical midpoint, subgroup, and group variables to summary data sets that are used to generate the graphs displayed in this section. The macro is intended to serve only as a prototype, since graphics output is determined by site requirements. It is available for download from the NESUG CD.

Simplifying Processing:

Summary data contain fewer records in WYSIWYG order. That this property simplifies validation is demonstrated in Figure 10 where 66 records from the original raw data are reduced to 4.



Statistics that are complicated in GCHART can also be greatly simplified by moving their calculation to the program that creates summary data. Consider, for example, the circuitous route required for calculating the percentage of HOURS by DEPT in the MEETINGS data set. It is not possible to directly obtain percentages of response variables in PROC GCHART. Instead, fractional HOURS have to be multiplied by 10 to get a preliminary *integer* result. Then a *frequency* is applied to the result that yields the correct proportion [3], p.62. Not only is this process complicated, it fails when raw data contain both negative and positive values. As an alternative, RPCT can be pre-calculated with an application of PROC SUMMARY and some division inside **%mkSummaryDataBarChart**. In Figure 11 the percentage of meeting hours is displayed by department for the MEETINGS data set.

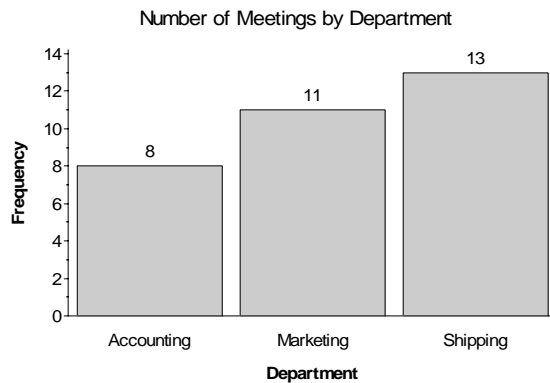


Lastly, almost identical GCHART commands can be issued to obtain *frequency*, *sum*, *percent* and *mean* bar charts. In Figure 12 the appropriate code fragments are placed over each of the four types of chart generated. Not shown are changes that need to be made to AXIS and TITLE statements. Again, for a complete listing see the NESUG CD.

Figure12. Almost identical code is used to generate frequency, sum, percent, and mean charts with summary input data to PROC GCHART. Only the argument to the SUMVAR option changes in the code associated with the four panels below.

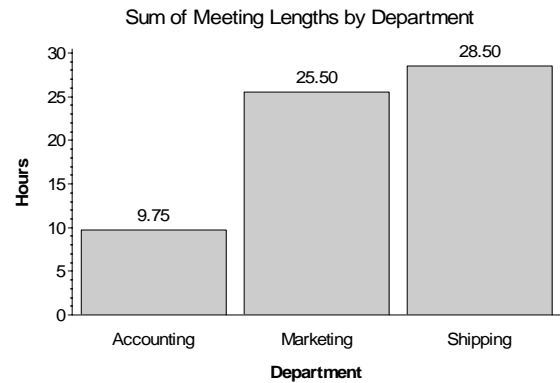
Frequency Chart

```
proc gchart data=N07.MeetingsOrigClsDptSummary;
  vbar classDesc / noframe width=&width type=SUM
  sumvar=ClassFreq outside=SUM ... ;
  format classDesc $dpt2Lfm.;
run; quit;
```



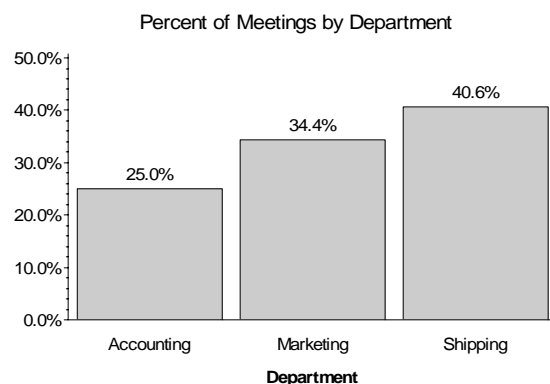
Sum Chart

```
proc gchart data=N07.MeetingsOrigClsDptSummary;
  vbar classDesc / noframe width=&width type=SUM
  sumvar=sum outside=SUM ... ;
  format classDesc $dpt2Lfm.;
run; quit;
```



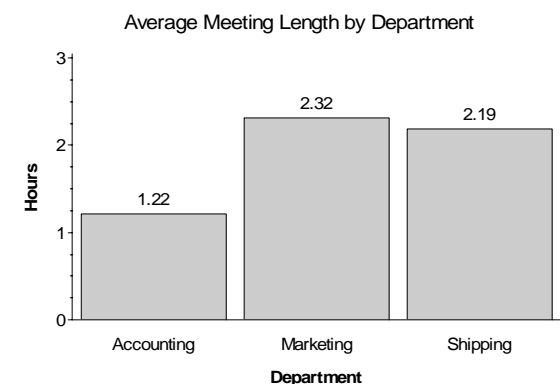
Percent Chart

```
proc gchart data=N07.MeetingsOrigClsDptSummary;
  vbar classDesc / noframe width=&width type=SUM
  sumvar=OAPCT outside=SUM ... ;
  format classDesc $dpt2Lfm.;
  format oapct pctRfm.; run; quit;
```



Mean Chart

```
proc gchart data=N07.MeetingsOrigClsDptSummary;
  vbar classDesc / noframe width=&width type=SUM
  sumvar=mean outside=SUM ... ;
  format classDesc $dpt2Lfm.;
run; quit;
```



Generating Different Types of Charts with Summary Data

Besides percentages for response variables, **%mkSummaryDataBarChart** can provide additional statistics that are not available when raw data are used in PROC GChart. Table1 below lists the variables from the summary data sets generated by **%mkSummaryDataBarChart**. Backgrounds are highlighted for those variables that cannot be graphed directly from raw data in PROC GCHART.

Table 1. Data Dictionary for Output Datasets Generated in **%mkSummaryDataBarChart**

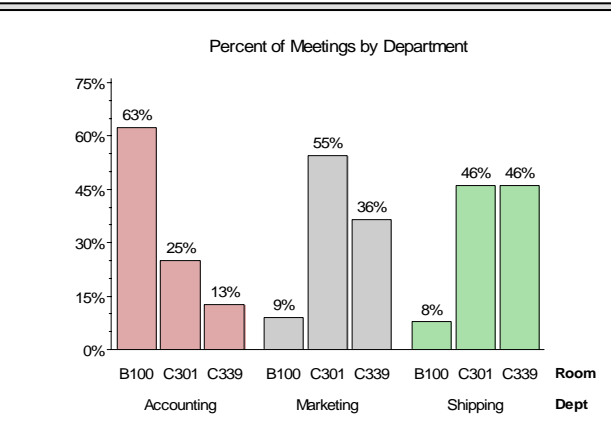
#	Variable	Type	Len	Label
1	ClassOrd	Num	8	Class (MP) Number
2	ClassDesc	Char	60	Class Description
3	GroupOrd	Num	8	Group Number
4	GroupDesc	Char	60	Group Description
5	subGroupOrd	Num	8	Subgroup Number
6	subGroupDesc	Char	60	Subgroup Description
7	mean	Num	8	MEAN RVar
8	sum	Num	8	SUM RVar
9	lclm	Num	8	LCLM RVar
10	uclm	Num	8	UCLM RVar
11	min	Num	8	MIN RVar
12	max	Num	8	MAX RVar
13	hiStdErr	Num	8	Upper SEM RVar
14	loStdErr	Num	8	Lower SEM RVar
15	hiStd	Num	8	Upper STD RVar
16	loStd	Num	8	Lower STD RVar
17	Rpct	Num	8	RVar Percent
18	ClassFreq	Num	8	Class (MP) Frequency
19	GroupFreq	Num	8	Group Frequency
20	subGroupFreq	Num	8	Subgroup Frequency
21	OAPct	Num	8	Overall Percent
22	mp100pct	Num	8	Midpoints add to 100%
23	g100pct	Num	8	Groups add to 100%
24	sg100pct	Num	8	Subgroups add to 100%

Variables MIN ... LOSTD are reviewed later when error bars are discussed. RPCT and OAPCT have already been covered. MP100PCT, G100PCT, and SG100PCT are illustrated in Figures 13 and 14 below.

Figure13 Bars with the same colors sum to 100% in the *group* charts below. G100Pct duplicates the G100 option in PROC GCHART, but MP100Pct would be impossible to reproduce without a summary input data set.

Group Chart: G100Pct

```
proc gchart data=N07.MeetingsOrigGrpSummary;
vbar classDesc / ... group=groupDesc type=SUM
sumvar=G100Pct outside=SUM patterned=group ... ;
format G100Pct pctIntfm.;
run; quit;
```



Group Chart: MP100Pct

```
proc gchart data=N07.MeetingsOrigGrpSummary;
vbar classDesc / ... group=groupDesc type=SUM
sumvar=MP100Pct outside=SUM patterned=midpoint;
format MP100Pct pctIntfm.;
run; quit;
```

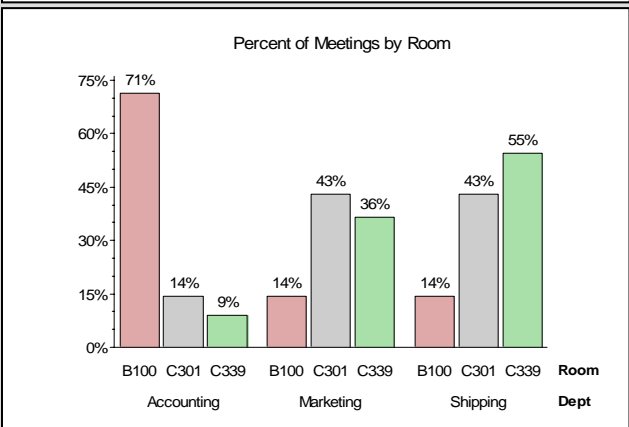
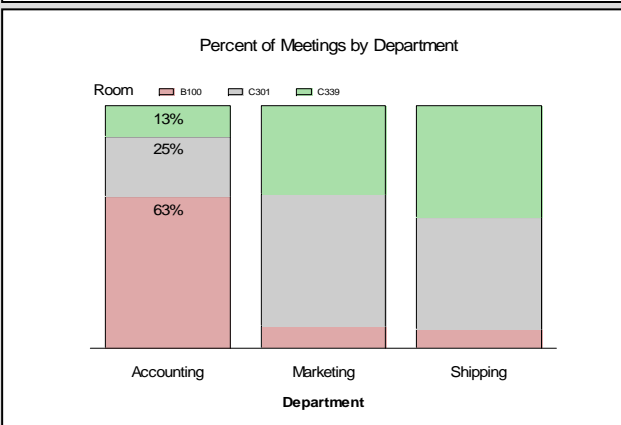


Figure14 When subgroups are plotted, MP100PCT and SG100PCT cannot be generated directly in GCHART. Since bars are colored by subgroup affiliation, only the first three percentages adding to 100 are displayed to emphasize the difference between MP100PCT and SG100PCT.

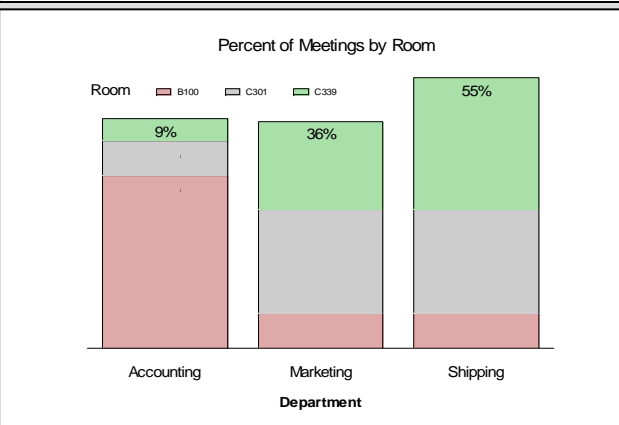
Subgroup Chart: MP100Pct

```
proc gchart data=N07.MeetingsOrigSGrpSummary;
vbar classDesc / noframe width=&width type=SUM
subgroup=SubgroupDesc sumvar=MP100Pct
outside=SUM ... ;
format MP100Pct pctIntfm.;
run; quit;
```



Subgroup Chart: SG100Pct

```
proc gchart data=N07.MeetingsOrigSGrpSummary;
vbar classDesc / noframe width=&width type=SUM
subgroup=SubgroupDesc sumvar=SG100Pct
outside=SUM ... ;
format SG100Pct pctIntfm.;
run; quit;
```

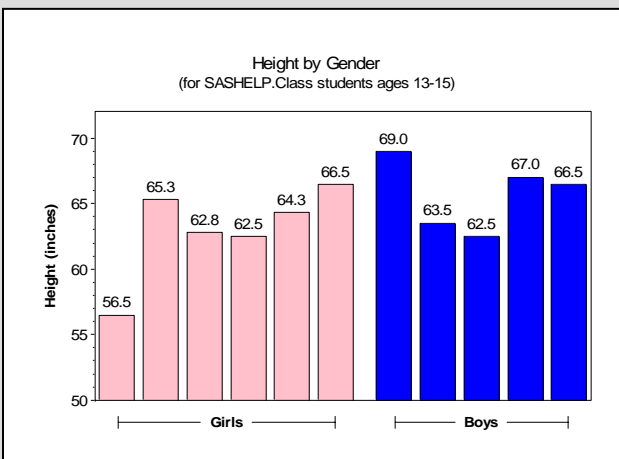


Again, not much code needs to be changed in GCHART to generate the different types of percent charts. From an inspection of the second charts in Figures 13 and 14, it is apparent that Accounting meets most frequently in B100, Marketing in C301, and Shipping in C399.

A non-hierarchical bar chart where subordinate membership is not repeated across groups can also be made from summary data. The SASHELP.Class data set already in summary format contains heights and weights for students who are grouped by gender. The NOZERO option is used to create the non-hierarchical bar chart in Figure 15. For more complete coverage of this topic, see the NESUG companion paper [3], p. 11.

Figure 15. The SASHELP.class data set is used to plot heights for students ranging in age from 13 to 15 years.

```
proc gchart data=N07.sasClass;
vbar Name /
width=5 type=SUM sumvar=Height outside=sum
Group=sex patternID=group NOZERO
raxis=axis1 maxis=axis2 gaxis=axis3
coutline=black woutline=1;
run; quit;
```

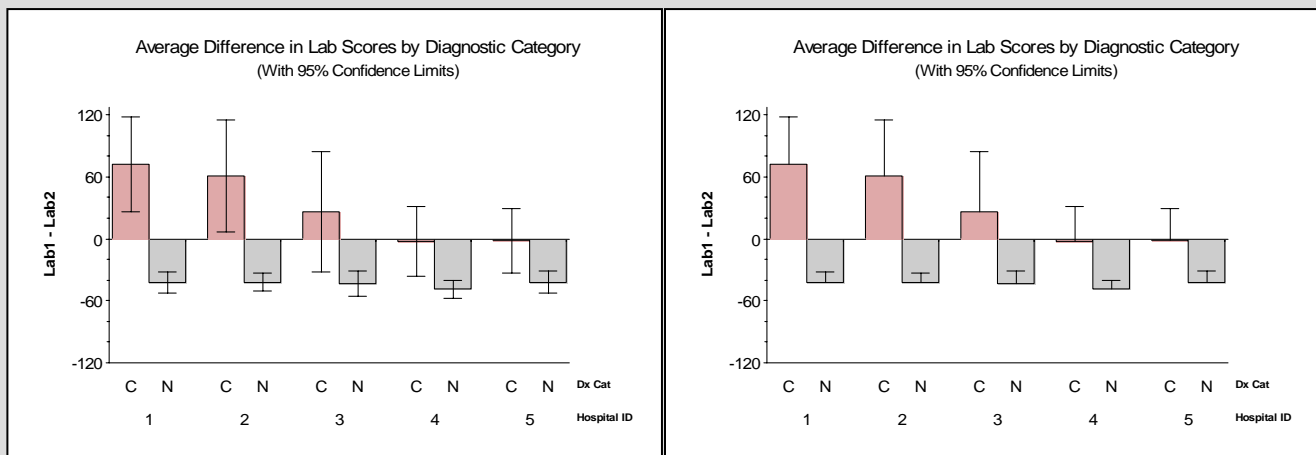


ERROR BARS IN PROC GCHART

New in Version 8 SAS/GRAPH software is the ERRORBAR option for creating error bars in a bar chart. ERRORBAR can take 'TOP', 'BOTH' or 'BARS' as arguments. For an example of the 'BARS' style see the horizontal bar chart on page 598 in the version 8 SAS/GRAPH manual [7]. Examples of 'BOTH' and 'TOP' error bars are displayed in this section.

Error bars generated inside PROC GCHART have several limitations. First, they cannot be graphed with a legend, because a legend requires a subgroup variable declaration and error bars cannot be plotted when subgroups are defined - even as disguises. Secondly, neither ERRORBAR='TOP' nor INSIDE= options work properly when chart bar heights are less than zero. Lastly, GCHART error bars must always be centered at the chart bar heights with lengths calculated from raw data sent to the procedure [7], p. 547. Needless to say uneven error bars cannot be plotted. Below in Figure 16 are two examples of charts generated with the ERRORBAR option from inside PROC GCHART.

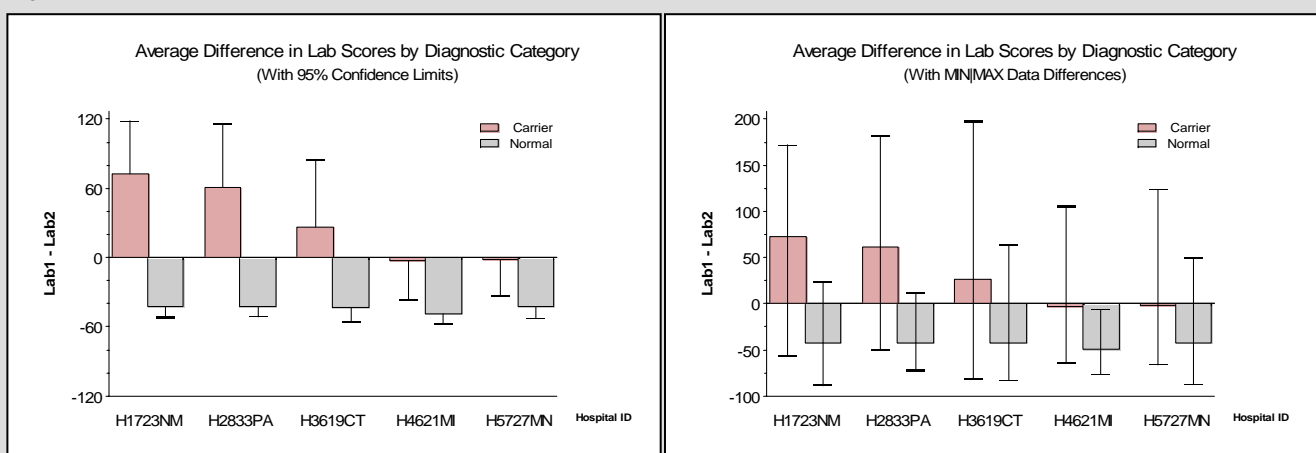
Figure 16. BOTH and TOP error bars from PROC GCHART. Without a legend, C and N Treatment groups are indecipherable, and error bars for the below-zero chart bars in the second panel should go down, not up.



Use ANNOTATE to Create Error Bars

The bar charts in Figure 17 show that the ANNOTATE error bars work better than their GCHART counterparts. A legend identifies 'C' as 'Carrier' and 'N' as 'Normal'. TOP error bars go down when the data are negative, and uneven error bars can be plotted that connect minimum and maximum values for a class within a group.

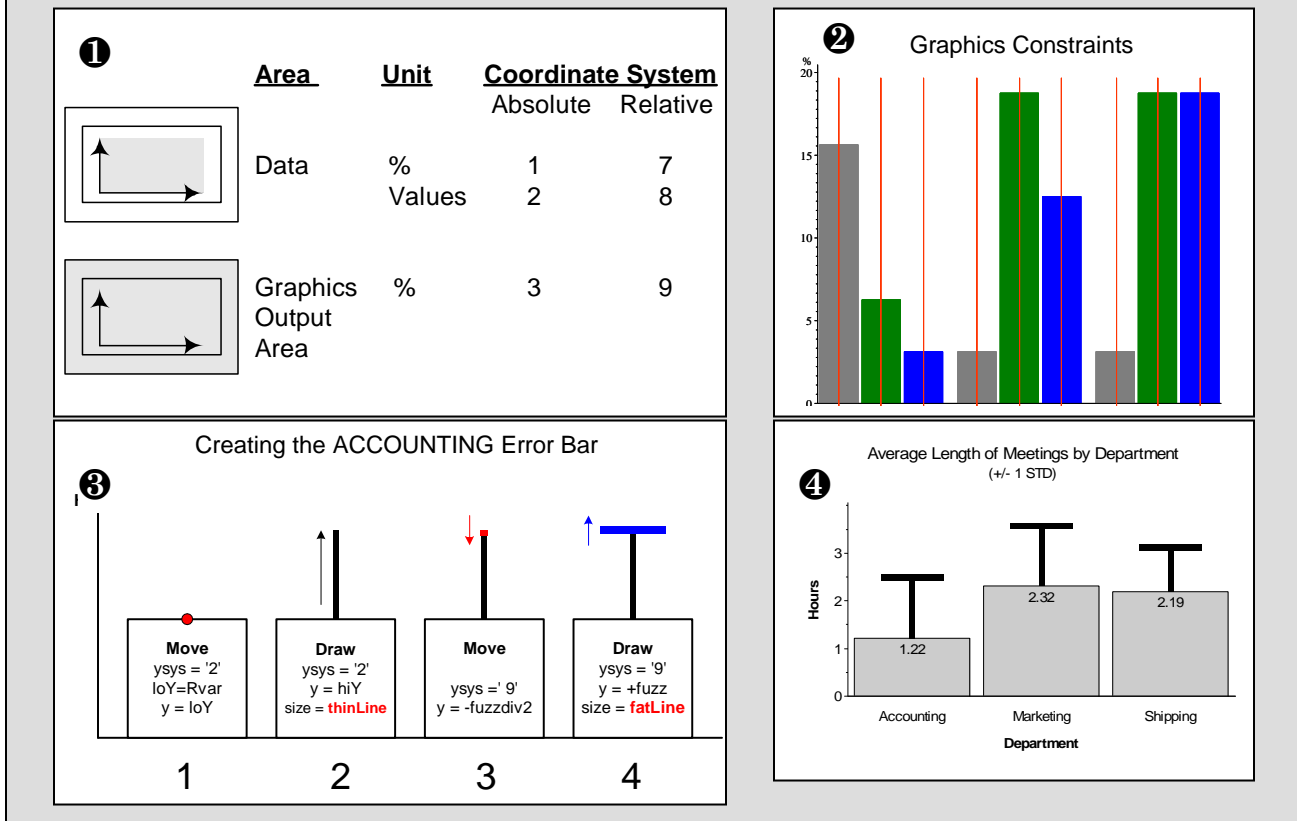
Figure 17. GCHART error bar problems are fixed with ANNOTATE. In the first panel, the error bars go down when averages drop below zero. 'BOTH' is required in the second panel, since the error bars are uneven.



Constructing error bars with ANNOTATE requires familiarity with the absolute and relative coordinate systems that drive XSYS, YSYS and HSYS assignments. The connection between YSYS, graphics constraints, and

MOVE|DRAW commands are displayed pictorially in Figure 18. Panel 1 from the figure is Copyright 1999, SAS Institute Inc., Cary, NC, USA. All Rights Reserved. Adapted with permission of SAS Institute Inc., Cary, NC.

Figure 18 How EBARANNO (ANNOTATE data set) works: ❶ maps the coordinate systems used in ❸. ❷ is a reminder that the only addressable regions in a bar chart lie along the midpoint axis, and ❸ shows the first error bar in ❹ being created without violating region restrictions defined in ❷.



GCHART'S Raxis works just like GPLOT'S Vaxis in ANNOTATE

From Figure 18, it can be seen that the RAXIS has full access to all real numbers between 0 and 4 in the fourth panel whereas access to MAXIS is restricted to 3 midpoint values. Given this configuration, the question that needs to be addressed is how to create the wide ticks that cap the error bars. The answer is really quite simple as demonstrated in the third panel from Figure 18. Just *draw* a thin line from the Mean to the top of the error bar. Then *move* down a short distance from the top and *draw* upwards again, this time with FATLINE set to &WIDTH in percent from %getBarInfo. The code fragment from EBARANNO that shows the process at work is displayed immediately below. The macro %displayEbars containing EBARANNO is listed in the appendix and available on the CD. From EBARANNO:

```
%if &_EbarType eq TOP %then %do;
  if Rvar ge 0 then loY=RVar; else hiY=Rvar;
%end;
if loY ne . and hiY ne . then do;
  function='move';
  %if &_EbarType eq TOP %then %do;
    if hiY gt 0 then do;
      ysys='2'; y=loY; output; /* #1 3rd panel */
      function='draw'; size=thinLine; y=hiY; output; /* #2 3rd panel */
      ysys='9'; function='move'; y=-fuzzdiv2; output; /* #3 3rd panel */
      function='draw'; size=fatLine; y=+fuzz; output; /* #4 3rd panel */
    end;
  ...
%end;
...
end;
```

- In the initial **move**, **ysys='2'** references absolute values in the data rectangle with a straight variable assignment; **y=loY**. Next, a thin line is drawn to HIY with **draw** and **y=hiY**.

- The assignment to YSYS changes to **ysys='9'** for a relative move that uses the entire graphics output area in percent as the coordinate system. FUZZ is a small number and FUZZDIV2 is one half of FUZZ. To use a relative coordinate system, a + or - must precede the variable as in (move) **-fuzzdiv2** and (draw) **+fuzz**. The change in direction for the last MOVE and DRAW commands centers FATLINE directly over HIY.

Assignments to the X variable are pretty much absent in EBARANNO.

Inside the EBARANNO data set, XSYS is set to '2' with a RETAIN statement. To locate horizontal coordinates, *midpoint*, and when needed, *group* variables must be defined. In **%displayEbars**, these assignments are made via macro parameter as in **midpoint = &MPvar**. A value for X would be ignored at this point.

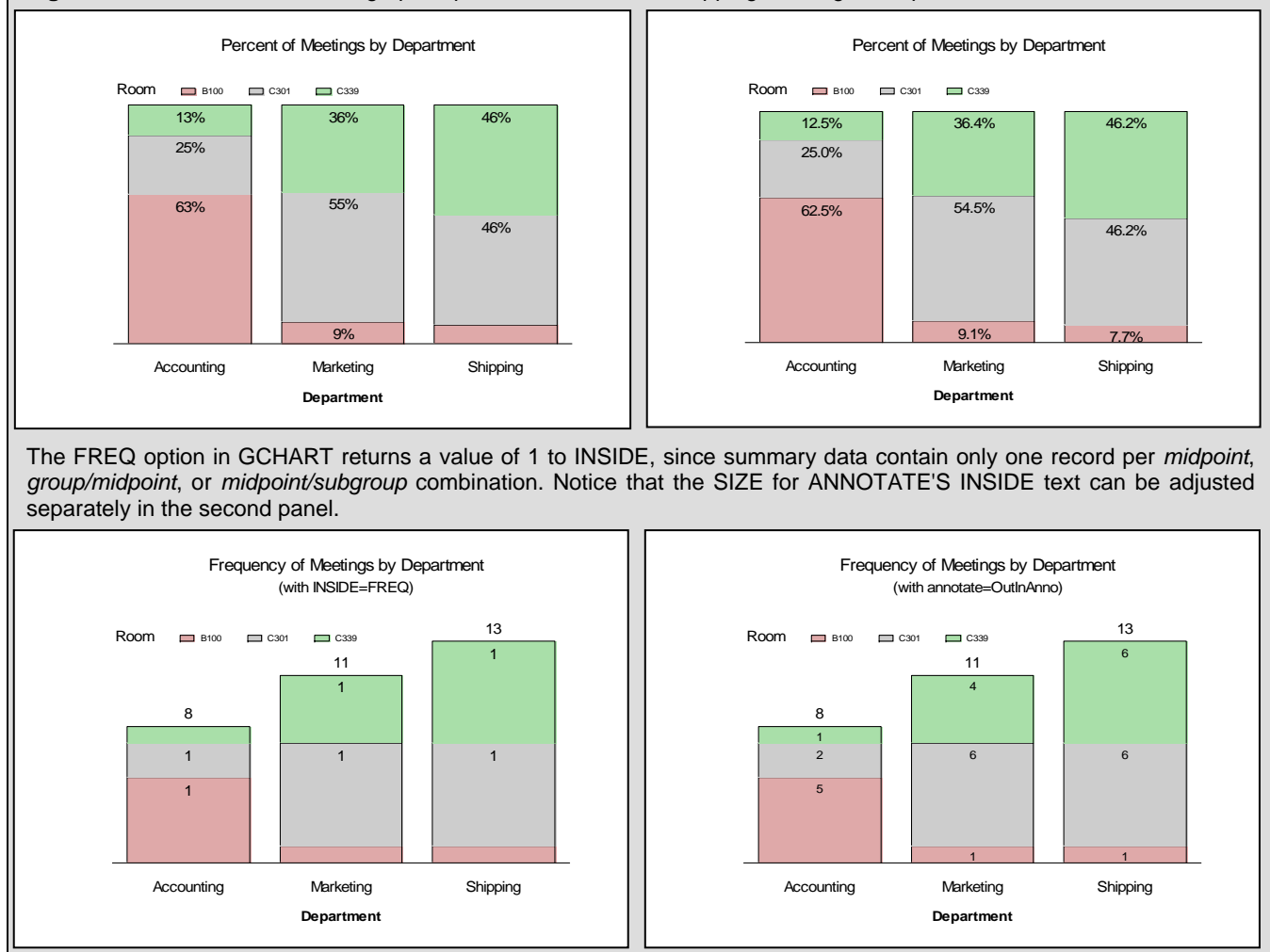
When XSYS changes from '2' in the data step, however, a value for X is required. In the code fragment below, **xsys='1'** (absolute data percent) and **ysys='2'** (absolute data values) so that a straight line can span the horizontal axis at **y='0'**. The following assignments guarantee that a horizontal zero line is drawn *after* the chart bars are laid down:

```
if last then do;
  xsys='1'; ysys='2'; x=0; y=0; function='move'; output;
  function='draw'; size=0.5; x=100; output;
end;
```

USE ANNOTATE TO CREATE INSIDE AND OUTSIDE TEXT

The second chart in Figure 19 shows that ANNOTATE prevents text from disappearing when an attempt is made to label all segments in a subgroup chart. The fourth panel shows ANNOTATE solving a serious problem that arises when summary data are used as input to PROC GCHART.

Figure 19. From the ANNOTATE graph in panel #2, 7.7% of the shipping meetings take place in room B100.

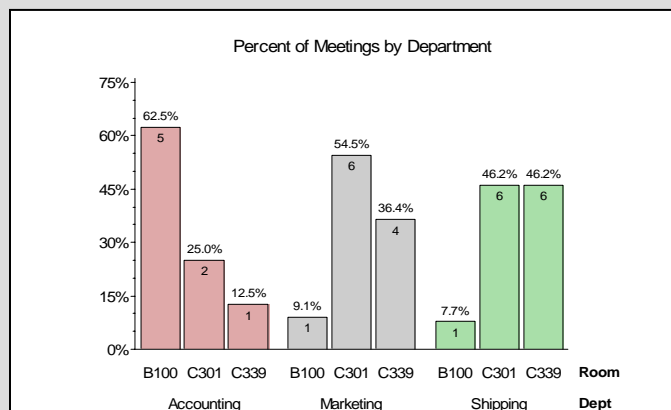


Unlike EBARANNO, a value for Y is not needed for labeling chart bars in the OUTINANNO data set. Instead, *midpoint* plus *group* or *subgroup* variables, if relevant, are sufficient. In the left panel of Figure 20, variables GROUP and MIDPOINT along with RVAR, OUTSIDE and INSIDE are newly created and stored in PLOTDAT. There is no need to declare a function in OUTSIDEAN, since LABEL is the default. Without MOVE or DRAW functions, there is no role for a Y variable in the data step. If one were present, it would be ambiguous in a subgroup chart when stacked bars were being labeled. As a final step in the **%OutsideInside** macro, data sets OUTSIDEAN and INSIDEAN are conditionally appended to form the final OUTINANNO data set.

Figure 20. Text is maneuvered with *position* commands in the annotate data set so that labels are placed slightly above or below the bars. Note that INSIDE, OUTSIDE and the RAXIS values all support different formats. INSIDE and OUTSIDE sizes are the same, however.

```
data outsideAn;
  length color style text $8;
  retain color 'black' when 'a'
  style 'Arial' xsys ysys '2'
  hsys '3' size &textSize;

  set plotDat;
  if rvar ge 0 OR rvar eq . then
    position='2'; /* one cell above */
  else
    position='8'; /* one cell below */
  text=left(put(outside,&outsideFmt));
run;
```



COMBINING THE FEATURES TO GENERATE A BETTER BAR CHART

The source code below generates the bar chart in Figure 21 that incorporates all recommendations made in this paper. Highlights are used to emphasize what needs to be coordinated in order to generate reliable output:

```
proc format;
  value hospFm 1= H1723NM 2= H2833PA 3= H3619CT 4= H4621MI 5= H5727MN;
run;

%OutsideInside(_indat=N07.biomedGrpSummary, _MPvar=ClassOrd, _groupVar=groupOrd,
  _Rvar=MEAN, _insideVar=MEAN, _textSize=3.75,
  _insideFmt=5.1);

%let result = %getBarInfo(Xorigin=13, SpaceSz=1, GspaceSz=4, Ngrps=5, Nbars=10, LCOLfrac=0.75);
%let width=%scan(&result,1,':'); %let gspace=%scan(&result,2,':');

%displayEbars(_indat=n07.biomedGrpSummary, _MpVar=classORD, _groupVar=groupORD,
  _Rvar=MEAN, _loY=LCLM, _hiY=UCLM, _fatLine=&width, _EbarType=TOP);

pattern1 color=CXDFA9A9;
pattern2 color=graycd;
legend1
  across=1 shape=bar(3,2) label=NONE
  position=(top inside right) mode=share
  value=(h=11pt 'Carrier' 'Normal');

axis1 label=( a=90 f="Arial/Bold" "Lab1 - Lab2") order=(-120 to 120 by 60) minor=(n=2);
axis2 label=none value=none;
axis3 label=(h=10Pt f="Arial/Bold" "Hospital ID") offset=(0.5 pct, 0.5 pct);

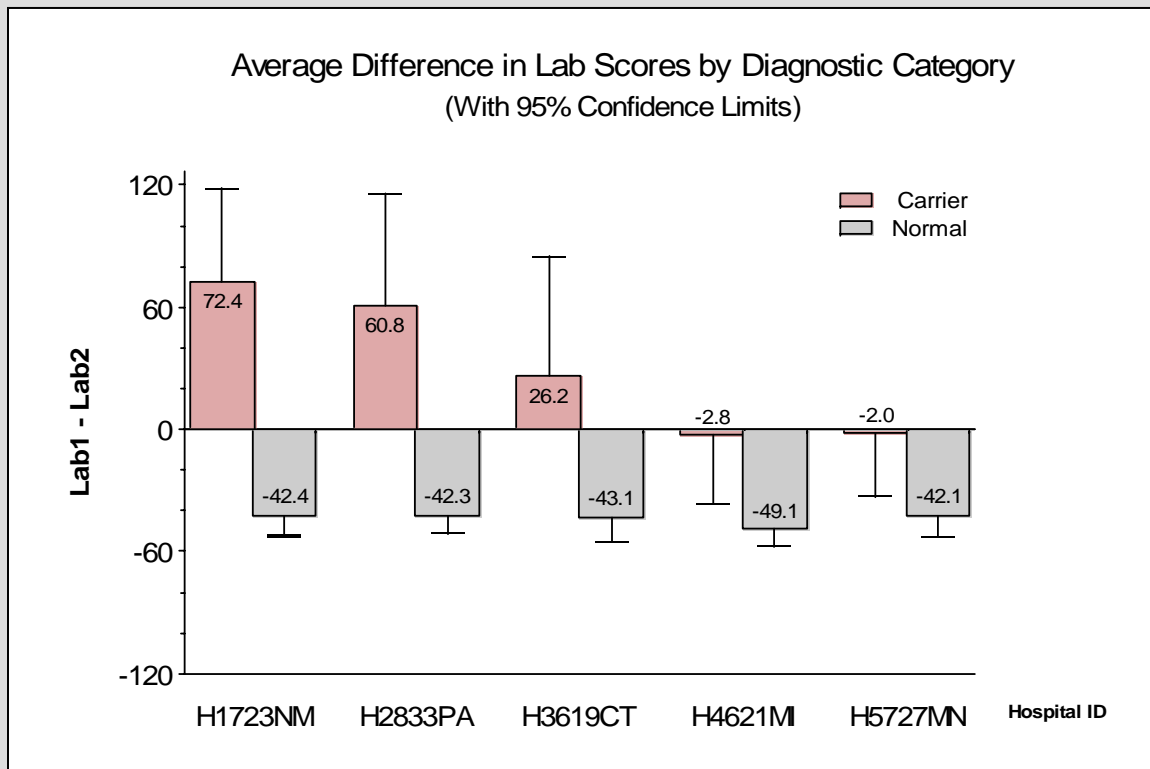
title1 move=(+8pct,+0pct) 'Average Difference in Lab Scores by Diagnostic Category';
title2 move=(+8pct,+0pct) '(With 95% Confidence Limits)';
filename chrt 'c:\N07\BetterBarChart\Paper\EMF\Chrt14_5.emf';
goptions htext=13pt htitle=15pt gsfname=chrt;
```

```
proc gchart data=N07.biomedgrpSummary annotate=OutInAnno;
  vbar classORD /noframe discrete width=&width annotate=EBarAnno
  gspace=&gspace type=SUM sumvar=MEAN
  Group=groupORD subgroup=classOrd legend=legend1
  raxis=axis1 maxis=axis2 gaxis=axis3
  coutline=black woutline=1;
  format groupOrd hospfm.;
run;
quit;
```

- `N07.biomedgrpSummary` appears as a parameter in `%OutsidelInside`, `%DisplayEBars`, and PROC GCHART.
- `MPVar` and `groupVar` that reference the midpoint and group variables must be the same in `%OutsidelInside`, `%DisplayEBars`, and PROC GCHART. (In this example, all references are to CLASSORD and GROUPORD).
- `RVar` is the same in both `%OutsidelInside`, `%DisplayEBars`, and it has to agree with the SUMVAR option in PROC GCHART.
- `&width` from `%GetBarInfo` defines the bar width in cells for GCHART and the height of `_fatLine` in percent for `%DisplayEBars`.
- `&gspace` from `%GetBarInfo` defines gspace in cells for GCHART.
- `annotate=OutInAnno` is the output data set from the `%OutsidelInside` macro.
- `annotate=EbarAnno` is the output data set from the `%DisplayEBars` macro.
- Finally, `subgroup=classord` and `legend=legend1` generate a legend in a group bar chart with error bars.

If you systematically review NP16.sas on the NESUG CD, the coordination highlighted above will become second-nature.

Figure 21. All recommendations made in this paper are used to enhance the bio-medical bar chart.



SUMMARY AND CONCLUSIONS

Techniques have been described in this paper for improving the quality and extending the breadth of the GCHART procedure. A WIDTH=*large number* or formula optimizes bar widths and intervening spaces so that a chart covers the entire data rectangle in a graph. Text management is improved with the insertion of legends into midpoint and group charts as well as the default subgroup chart. Using summary rather than raw data also enables any type of statistic available in SAS software to be charted, and finally ANNOTATE applied to the summary data generates customized error bars and imitates the placement of INSIDE and OUTSIDE text in PROC GCHART.

ACKNOWLEDGEMENTS

The author thanks Craig Anderson, project manager, for his support and guidance over the past couple of years. This paper is dedicated to him.

Gratitude is also extended to SAS Institute Inc. for granting permission to use the DEPT data set from *The How-To Book for SAS/GRAPH® Software* by Thomas Miron [4]. A larger version of this data set with four departments and four rooms is used in a couple of examples.

COPYRIGHT STATEMENT

The paper, *Building a Better Bar Chart with SAS/GRAPH® Software*, along with all associated files on the NESUG CD is protected by copyright law. This means if you would like to use part or all of the original ideas or text from these documents in a publication where no monetary profit is to be gained, you are welcome to do so. All you need to do is to cite the paper in your reference section. For ALL uses that result in corporate or individual profit, written permission must be obtained from the author. Conditions for usage have been modified from <http://www.whatiscopyright.org>.

REFERENCES

- [1] Carpenter, Art. *Annotate: Simply the Basics*. Cary, NC: SAS Institute Inc., 1999.
- [2] Carpenter, Art. *Carpenter's Complete Guide to the SAS® Macro Language: Second Edition*. Cary, NC: SAS Institute Inc., 2004.
- [3] Miron, Thomas. *The How-To Book for SAS/GRAPH Software*. Cary, NC: SAS Institute Inc., 1995. Copyright 1995, SAS Institute Inc., Cary, NC, USA. All Rights Reserved. Reproduced with permission of SAS Institute Inc., Cary, NC.
- [4] Watts, Perry. *Charting the Basics with PROC GCHART*. Proceedings of the 20th Annual Northeast SAS Users Group Conference. Baltimore, MD, 2007, paper #FF17.
- [5] Whitlock, Marianne. *Let Summary Sum and Tabulate Format*. Proceedings of the Twenty-Seventh SAS® User Group International Conference, Cary, NC: SAS Institute Inc., 2002, paper #78.

Web Citations:

- [6] <http://lib.stat.cmu.edu/datasets/biomed.data>. *From Stat Lib --- Data Sets Archive*. Source of the bio-medical data set used in this paper. The data were transformed after download. More specifically, hospital numbers were reassigned with the MOD function, and a ceiling to maximum values was set for the differences between Lab1 and Lab2 results.

SAS Institute References:

- [7] SAS Institute Inc. *SAS/GRAPH® Reference, Version 8*, Cary NC: SAS Institute Inc., 1999.
- [8] SAS Institute Inc. *SAS/GRAPH® Reference, Volumes 1, 2, and 3*, Cary NC: SAS Institute Inc., 2004.
- [9] TS-DOC:TS-674. *An Introduction to Exporting SAS/GRAPH Output to Microsoft Office SAS Release 8.2 and higher*. <http://support.sas.com/techsup/technote/ts674/ts674.html>.

CD:

- [10] *Robert Allison's SAS/Graph Examples!* See Col2.sas.

WHAT'S ON THE NESUG CD:

- 1) Contents of the **DATA** subdirectory:
ALL data sets used to generate the bar charts in this paper.
- 2) Contents of the **MACROS** subdirectory:
All macros used to generate the bar charts and data used in this paper.
 - displayEbars.sas (also listed in the appendix)
 - GetBarInfo.sas (also listed in the appendix)
 - MkSummaryDataBarChart.sas
 - OutsideInside.sas (also listed in the appendix)
- 3) Contents of the **PROGRAMS** subdirectory:
All programs that alter data or generate bar charts.
 - AddtoMeetingsDat.sas: Adds records to Miron's MEETINGS data set.
 - MkBioMedData.sas: Alters [6] in the reference section.
 - MkSummaryBioMedData.sas: calls **%MkSummaryDataBarChart**.
 - MkSummaryMeetingsDat.sas: calls **%MkSummaryDataBarChart**.
 - NP16.sas: generates all the graphs used in this paper. Appendix macros are invoked in this program.

TRADEMARK CITATION

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

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

CONTACT INFORMATION

The author welcomes feedback via email at perryWatts@comcast.net

APPENDIX: MACROS FOR GCHART

```

/* -----
Program   :   GetBarInfo.sas
Purpose   :   Obtain values for WIDTH and GSPACE (in Cells) given
               the number of bars, groups, Space (in percent) and Gspace (in
               percent).
Notes     :   1) This is a macro function. See "Carpenter's Complete Guide to
               the Macro Language: Second Edition" for a description of macro
               functions and instructions for how to use a macro library.
               2) A return value in cells for SPACE is not needed.
               3) Use a scan function to parse the return value e.g:
%let result=%getBarInfo(Xorigin=17, SpacesSz=2, GspaceSz=2, Ngrps=1,
                        Nbars=3, LCOLfrac=0.75);
%let width=%scan(&result,1,':');
%let gspace=%scan(&result,2,':');

Parms     :   Xorigin      - in Percent
               SpacesSz    - space between bars (in percent)
               GspaceSz    - space between groups (in percent)
               NGrps       - Number of groups (If midpoint chart, enter 1)
               NBars       - #groups * #midpoints for hierarchical bar chart
                           or a midpoint or subgroup barchart.
               LCOLFrac    - Multiply LCOLS from PROC GDEVICE by 0.01
                           e.g. for EMF = 0.75
----- */
%macro getBarInfo(Xorigin=, SpacesSz=, GspaceSz=, Ngrps=, Nbars=, LCOLfrac=);
%local axislen nGspaces nSpaces width space gspace;
%let axisLen = %sysevalf(100 - &Xorigin);
%if &Ngrps le 1 %then %do;
%let GspaceSz=0; %let Ngrps=1;
%end;
%let nGspaces = %sysevalf(&nGrps - 1);
%let nSpaces=%eval(&nbars - &nGspaces - 1);
%let axisLen =
%sysevalf( &axisLen - %sysevalf(&nspaces * &SpacesSz) - %sysevalf(&nGspaces * &GspaceSz));
%let width = %sysevalf(&axisLen. / &nbars.);
/* CONVERT FROM PERCENT TO CELLS WITH LCOLFRAC */
%let space = %sysevalf(&LCOLfrac * &SpacesSz); /*NOT USED FURTHER ON, HOWEVER*/
%let width=%sysevalf(&LCOLfrac * &width);
%let gspace = %sysevalf(&LCOLfrac * &GspaceSz);
%trim(&width):%trim(&gspace);
%mend getBarInfo;

```

```

/* -----
Program   :   displayEbars.sas
Purpose   :   Create error bars for bar charts via macro.
Notes     :   Works only with summary data.
               Designed for the EMF device.
               Subgroup Charts cannot have error bars.
Input      :   from _indat parameter
Output     :   data set work.EBarAnno
Parms     :
Name       :   Description
-----
_inDat     :   Input dataset (including Libname if applies)
_MPvar     :   Midpoint variable: Can be character or Numeric. Must
               match GCHART midpoint variable.
_groupVar  :   Group variable (if applies) Can be character or
               Numeric. Must match GCHART group variable.
_Rvar      :   Response Variable (Typically MEAN)
_loY       :   Low-Range variable
_hiY       :   High Range Variable
_fatLine   :   In percent - for the wide tick widths at tops and bottoms
               of error bars. For EMF device, &width in CELLS is used
               for the width in PERCENT of the bars.
_thinLine  :   In percent - for the width of the error bar line itself.
               _ThinLine is also used to calculate the vertical HEIGHT
               of the _fatLine tick. (Range generally is 0.5 to 2).
               With EMF device wide ticks can be so short that they
               are not visible on the screen (print is OK). CGM does
               not have this problem.

```

```

_EbarType                TOP or BOTH -- same as GCHART except that bars plot
                          as non-existent BOTTOM when values are negative.
----- */

%macro displayEbars(_indat=, _MPvar=, _groupVar=, _Rvar=, _loY=, _hiY=,
                    _fatLine=&width, _thinLine=0.5, _EbarType=TOP);
%if &_groupVar ne %str( ) %then
    %let keep=%str(xsys ysys hsys function size line x midpoint y color group when);
%else
    %let keep=%str(xsys ysys hsys function size line x midpoint y color when);
%let fuzz = &_thinLine;
%let fuzzdiv2=%sysevalf(&fuzz / 2);

data EBarAnno(keep=&keep);
    length function color $8;
    retain fuzz &fuzz fuzzdiv2 &fuzzdiv2 thinLine &_thinLine fatLine &_fatLine;
    retain xsys '2' hsys '3' when 'a' line 1 color 'black';
    set &_indat end=last;
    %if &_groupVar ne %str( ) %then group=&_groupVar; ;
    midpoint=&_MPvar;
    loY=&_loY; hiY=&_hiY; Rvar=&_Rvar;
    %if &_EbarType eq TOP %then %do;
        if Rvar ge 0 then loY=Rvar;
        else hiY=Rvar;
    %end;
    if loY ne . and hiY ne . then do;
        function='move';
    %if &_EbarType eq TOP %then %do;
        if hiY gt 0 then do;
            ysys='2'; y=loY; output; /*Error bars go up. Start from bottom (top of positive bar)*/
            function='draw'; size=thinLine; y=hiY; output;
            ysys='9'; function='move'; y=-fuzzdiv2; output;
            function='draw'; size=fatLine; y=+fuzz; output;
        end;
        else do; /*Error bars go down. Start from "top". (bottom of negative bar)*/
            ysys='2'; y=hiY; output;
            function='draw'; size=thinLine; y=loY; output;
            ysys='9'; function='move'; y=+fuzzdiv2; output;
            function='draw'; size=fatLine; y=-fuzz; output;
        end;
    %end;
    %else %do; /*BOTH*/
        ysys='2'; Y=rVar; output; /*Error bar goes up. Start from MEAN. */
        function='draw'; size=thinLine; y=hiY; output;
        ysys='9'; function='move'; y=-fuzzdiv2; output;
        function='draw'; size=fatLine; y=+fuzz; output;
        /*Go back to the MEAN and then plot the lower half of the E Bar */
        ysys='2'; function='move'; Y=rVar; output;
        function='draw'; size=thinLine; y=loY; output;
        ysys='9'; function='move'; y=+fuzzdiv2; output;
        function='draw'; size=fatLine; y=-fuzz; output;
    %end;
    end;
    if last then do;
        xsys='1'; ysys='2'; x=0; y=0; function='move'; output;
        function='draw'; size=0.5; x=100; output;
    end;
run;
%mend displayEbars;

```

```

/* -----
Program   :   OutsideInside.sas
Purpose   :   Create an annotate data set that places OUTSIDE and/or INSIDE
               (resizable) text on a bar chart.
Notes     :   Works only with summary data.
Input     :   from _indat parameter
Output    :   data set work.OutInAnno
Parms     :

Name      :   Description
-----
_InDat    :   Input dataset (including Libname if applies)
_MPvar    :   Midpoint variable: Can be character or Numeric. Must

```

```

match GCHART midpoint variable.
_groupVar      Group variable (if applies) Can be character or
               Numeric. Must match GCHART group variable.
_sgroupVar     Subgroup variable (if applies). Can be character or
               Numeric. Must match GCHART subgroup variable.
_Rvar          Response Variable (e.g. MEAN, CLASSFREQ).
_outsideVar    Outside variable that supplies text for OUTSIDE option.
_insideVar     Inside variable that supplies text for INSIDE option.
_textSize      In percent.
_outsideFmt    format for OUTSIDE text (typically matches format applied
               to RVAR in PROC GCHART).
_insideFmt     format for INSIDE text.
----- */

%macro OutsideInside(_indat=, _MPvar=, _groupVar=, _sgroupVar=, _Rvar=, _outsideVar=,
                   _insideVar=, _textSize=, _outsideFmt=best8., _insideFmt=best8.);
/* CLEAR OUT DATA SETS FROM PREVIOUS RUNS*/
proc datasets lib=work nolist;
  delete plotdat outsideAn insideAn OutInAnno;
quit;
/* CREATE PLOTDAT */
data plotDat(keep=group midpoint subgroup rvar outside inside);
  set &_indat;
  %if &_groupVar ne %str( ) %then %str(group=&_groupVar);
  %else %str(group = .);
  %if &_sgroupVar ne %str( ) %then %do;
    %str(subgroup=&_sgroupVar);
    %let _outsideVar=%str( ); /*OUTSIDE DOES NOT WORK IN ANNOTATE FOR SUBGROUP CHARTS */
  %end;
  %else %str(subgroup = .);
  midpoint=&_MPvar;
  %if &_outsideVar ne %str( ) %then %str(outside=&_outsideVar);
  %if &_insideVar ne %str( ) %then %str(inside=&_insideVar);
  rvar=&_rvar;
run;
/* CREATE ANNOTATE DATA SETS */
%if &_OutsideVar ne %then %do;
  data outsideAn;
    length color style text $8;
    retain color 'black' when 'a' style "Arial"
           xsys ysys '2' hsys '3' size &_textSize;
    set plotDat;
    if rvar ge 0 OR rvar eq . then position='2';
    else position='8';
    text=left(put(outside,&_outsideFmt));
  run;
%end;
%if &_InsideVar ne %then %do;
  data InsideAn;
    length color style text $8 position $1;
    retain color 'black' when 'a' style "Arial"
           xsys ysys '2' hsys '3' size &_textSize;
    set plotDat;
    if subgroup ne . then do;
      if inside gt 0;
        if rvar gt 0 then position='E'; /*half cell below*/
        else position='B';             /*half cell above*/
      end;
    else do;
      if rvar gt 0 then position='E'; /*half cell below*/
      else position='2';             /*one cell above*/
    end;
    if outside eq . and inside eq 0 then position='2';
    text=left(put(inside,&_insideFmt));
  run;
%end;
%if &_OutsideVar ne %str( ) AND _InsideVar ne %str( ) %then %let SetNames= OutsideAn InsideAn;
%else %if &_OutsideVar ne %str( ) %then %let setNames = OutsideAn;
%else %let setNames= InsideAn;

  data OutInAnno;
    set &setNames;
  run;
%mend outsideInside;

```