Paper 093-2007

How to Make the "Best Choice" from the Many Ways to Create and Deliver SAS® Graphs

LeRoy Bessler, Assurant Health, Milwaukee, Wisconsin, USA

Abstract

Whether you are a point-and-click user of SAS Enterprise Guide or a code-oriented SAS programmer, you have multiple options for packaging and delivery, and a possibly bewildering array of options and device drivers for graphs. For production packaging, you can go directly to a finished product/file such as HTML for the web, Word-compatible RTF, or PDF. For an ad hoc analytical or research project, you may need to insert graphs in a custom, hands-on-developed Microsoft PowerPoint presentation or Microsoft Word document. Whether for production or ad hoc graphic output, besides the all of the design options, there is a huge array of SAS/GRAPH® device drivers to pick from. This paper shares ideas, insights, and practical experience to help you make the best choice to suit your needs, and to get the best out of option-rich software from the standpoint of usability (i.e., communication effectiveness) and reliability.

Introduction

The Best Choice must be a communication-effective choice and a reliable choice.

This paper is part of a trio. Here is how the three instruments play together. Reference 1 covers communication-effective graph design and construction, and communication-effective web design. It provides widely usable examples of bar charts and trend charts. Reference 2 covers communication-effective pie charts. This paper includes:

(a) fully commented coding details for customizing a basic version of the three most popular types of presentation graphs to implement the principles of communication effectiveness that are presented at great length in Reference 1;

(b) recommendation of SAS/GRAPH device drivers for different situations;

(c) how to prepare graphs as Graphic Stream Files (GSF) on disk for insert into Microsoft PowerPoint or Microsoft Word, and how to customize them after insert, if desired;

(d) ODS destinations of RTF, PDF, and HTML, with the HTML discussion limited to how to optimally size a graph for a web page (the HTML destination and web design are covered extensively in Reference 1); and

(e) "Accessibility"—how to reach the widest possible audience with graphic data presentation.

Sometimes finding the best choice is a matter of finding the right choice.

It can be challenging when, without any warning in the SAS log, a device driver other than what you specified is used, or when the unattractive and can-be-hard-to-read SIMULATE font ("Is that 0 or D?") is substituted for what you specified. The SAS/GRAPH manual is over 1,600 pages, and the ODS manual is over 700 pages. Add to that the information at support.sas.com in SAS Usage Notes, Technical Notes, Communities, and Samples. Rather than refer you to a reading assignment of probably well over 2,500 pages, my modest goal here is to give you some suggestions that are likely to work for you as a reliable model or starting point, so that you can efficiently exploit that huge knowledge base along the shortest route to a solution that meets your needs exactly.

Improving a Pie Chart (Text is more readable for both charts if not reduced to half page width.)

Figure 1: Default Pie Chart

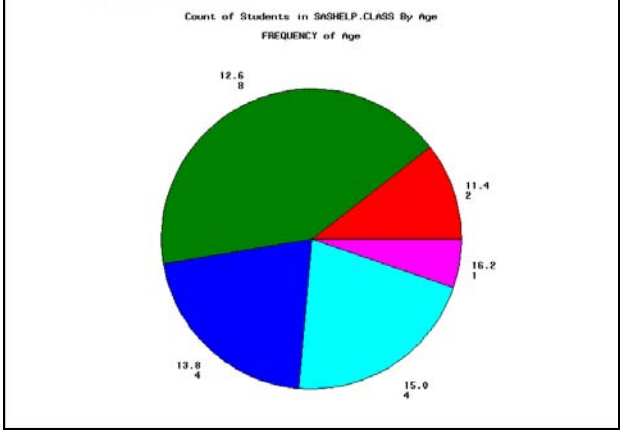
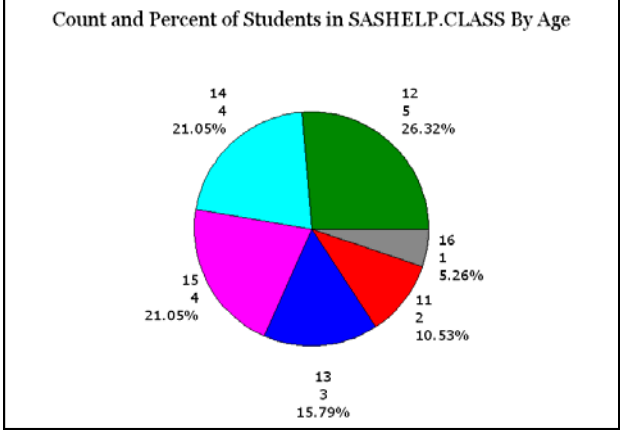


Figure 2: Custom Pie Chart



Here is the code used to create the Custom Pie Chart:

```
goptions border; /* border around image, but adding Word border in the paper */
goptions ftext='Verdana'; /* font  for text for which no font=   is assigned */
goptions htext=3.75 PCT;  /* height of text for which no height= is assigned */
title1 height=1 PCT ' ';          /* white space at top    of graph */
footnote1 angle=0 height=1 PCT ' ';   /* white space at bottom of graph */
footnote2 angle=+90 height=1 PCT ' '; /* white space at right  of graph */
footnote3 angle=-90 height=1 PCT ' '; /* white space at left   of graph */
title2 font='Georgia' height=5 PCT
  justify=LEFT color=CX000000 '  Figure 2: ' color=CX0000FF 'Custom Pie Chart'
  justify=LEFT height=2.5 PCT " "; /* white space before the next TITLE line */
title3 font='Georgia' height=5 PCT color=CX000000
  'Count and Percent of Students in SASHELP.CLASS By Age';

/* in HEIGHT= above, and in OFFSET= later, PCT means Percent of total graph height */

proc gchart data=sashelp.class;
* For a pie chart of total weight of students in each Age group,
  use SUMVAR=WEIGHT after the / in the PIE statement. *;

pie age /* create pie slices for AGE */
  /
  discrete   /* use the discrete values of AGE,
                not midpoints of default subranges  */
  noheading  /* suppress the default heading
                (in this case, "FREQUENCY of Age")  */
  descending /* order the slices by decreasing size */
  percent=outside;
/* Display Percent of Total Frequency for each slice.
   VALUE and SLICE (Slice Name) are defaults.
   Other options are NONE, INSIDE, and ARROW.
   "ARROW" connects display outside to the slice. */
/* To turn off pie slice color, include a PATTERN statement like this:
   PATTERN1 VALUE=PEMPTY REPEAT=99;
   99 can be any number greater than or equal to the number of slices.
   For how to control pie slice color,
   please see Reference 2 or the SAS/GRAPH manual. */

run; quit;
```

Here is the code to create the Default Pie Chart:

```
/* three title statements */
proc gchart data=sashelp.class;
pie age; run; quit;
```

Improving a Horizontal Bar Chart

Figure 3: Default Bar Chart

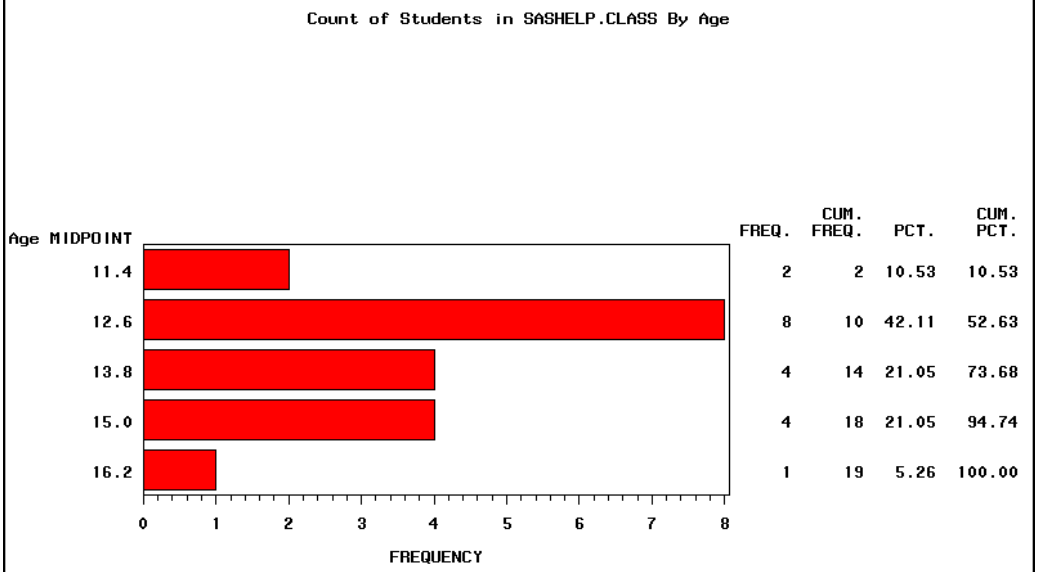
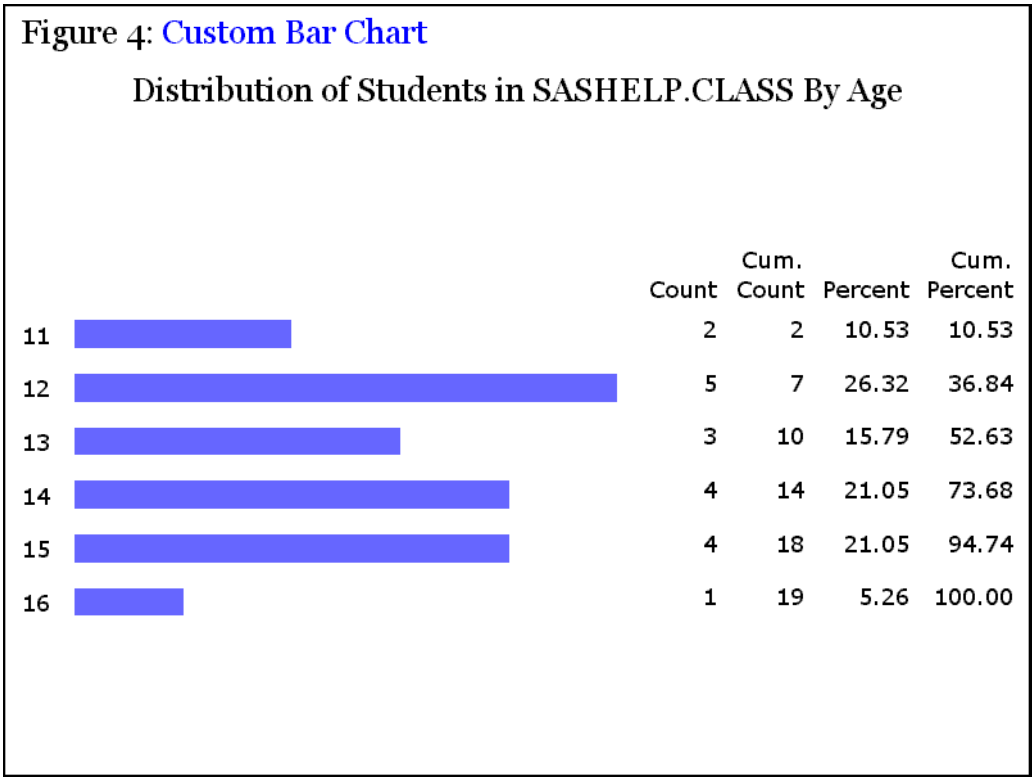


Figure 4: Custom Bar Chart



Customization replaces default subrange midpoints for Age with the actual discrete Age values, eliminates the unneeded response axis entirely, removes the midpoints axis line and chart frame, suppresses the midpoints axis label which is already identified in the graph title, takes control of the column label text, and uses a lighter bar color that is less image-overwhelming.

Here is the code used to create the Custom Bar Chart:

```
goptions border; /* border around image, but adding Word border in the paper */
goptions ftext='Verdana'; /* font  for text for which no font=   is assigned */
goptions htext=3.75 PCT;  /* height of text for which no height= is assigned */

title1 height=1 PCT ' ';              /* white space at top    of graph */
footnote1 angle=0 height=1 PCT ' ';   /* white space at bottom of graph */
footnote2 angle=+90 height=1 PCT ' '; /* white space at right  of graph */
footnote3 angle=-90 height=1 PCT ' '; /* white space at left   of graph */

title2 font='Georgia' height=5 PCT
  justify=LEFT color=CX000000 '  Figure 4: ' color=CX0000FF 'Custom Bar Chart'
  justify=LEFT height=2.5 PCT " "; /* white space below TITLE2 */

title3 font='Georgia' height=5 PCT color=CX000000
  'Distribution of Students in SASHELP.CLASS By Age';

pattern1
  value=solid     /* solid fill for the bars   */
  color=CX6666FF; /* lighter browser-safe blue */

axis1
  label=none /* suppress the axis label */
  major=none /* suppress tick marks for every value along the axis */
  minor=none /* suppress tick marks between any major tick marks */
  style=0    /* suppress the axis line */
  offset=(0,7 PCT);
  /* OFFSET for the top of the axis prevents Count and Percent labels from
     overlapping the values listed below them for the top bar in the charts. */

axis2 label=none major=none minor=none style=0
  value=none /* suppress values along the axis */
  offset=(2 PCT, 2 PCT); /* create white space at each end of the axis */

proc gchart data=sashelp.class;

hbar age /
  discrete /* use the discrete values of AGE,
              not midpoints of default subranges                              */
  freq /* Frequency of the discrete values of AGE is the RESPONSE variable   */
       /* list the values for each AGE in a column to the right of the bars  */
  freqlabel='Count' /* custom label for Frequency column                      */
  cfreq  /* provide a Cumulative Frequency column  to the right of the bars  */
  cfreqlabel='Cum. Count' /* custom label for Cumulative Frequency column    */
  percent /* list the Percent of total frequency for each AGE in a column    */
  percentlabel='Percent' /* custom label for Percent column                  */
  cpercent  /* provide a Cumulative Percent column  to the right of the bars */
  cpercentlabel='Cum. Percent' /* custom label for Cumulative Percent column */
  maxis=axis1 /* AXIS1 statement defines the axis for the MIDPOINT variable  */
  raxis=axis2 /* AXIS2 statement defines the axis for the RESPONSE variable  */
  width=1.6   /* make bar width close to the height of the text             */
  space=1.6;  /* make space between bars equal to the bar width             */

run; quit;
```

Here is the code to create the Default Bar Chart:

```
/* three title statements */
proc gchart data=sashelp.class;
hbar age;
run; quit;
```

Improving a Trend Plot (Please see Reference 1 for the author's preferred alternatives.)

Figure 5: Default Plot

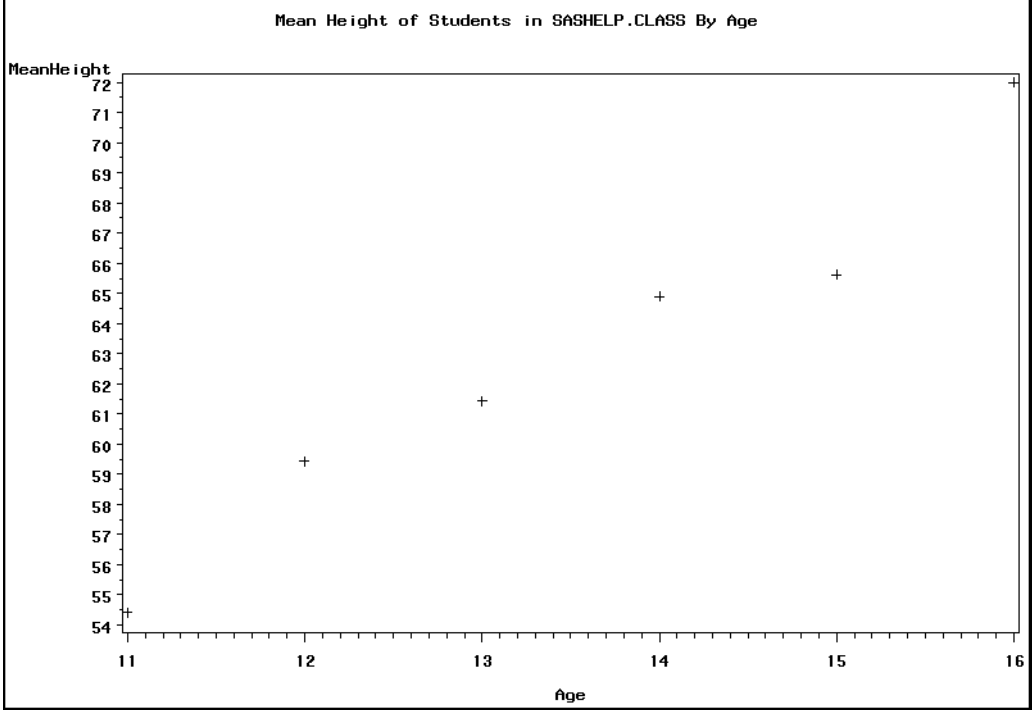
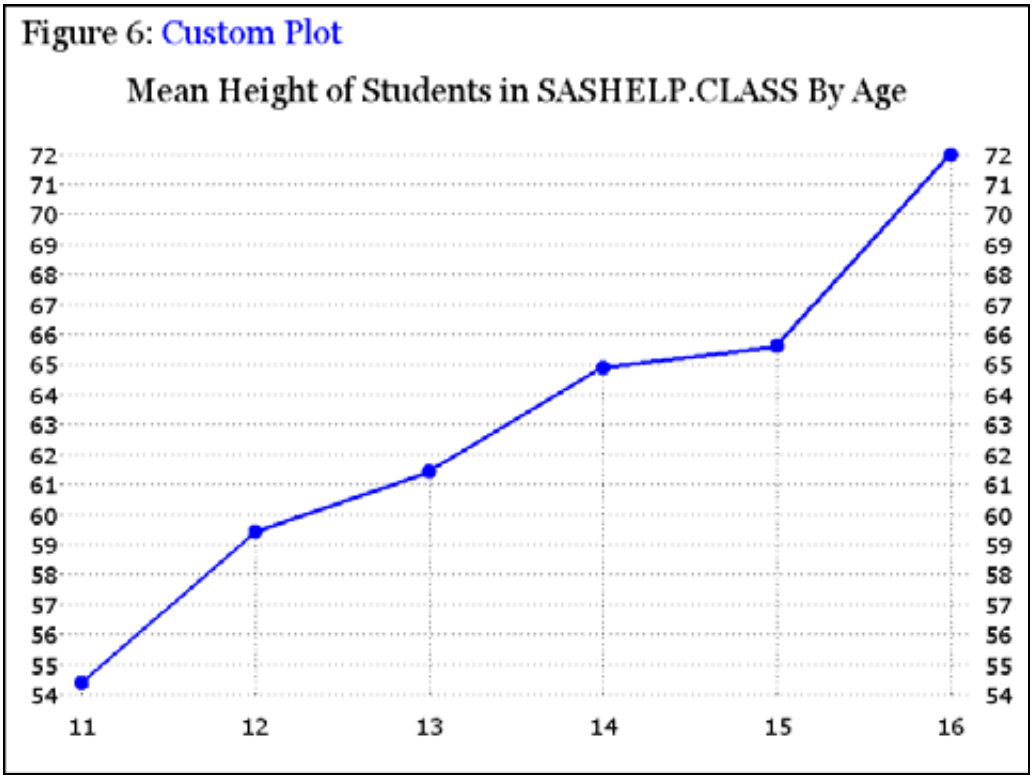


Figure 6: Custom Plot



In the graph above, with so few x values, there really is no justification to provide the needle lines from the plot points to the tick-mark values. In a more realistic situation, however, they would help. To eliminate reliance on this very busy design in order to assure that the viewer can estimate the y values, Reference 1 offers better solutions.

Here is the code used to create the Custom Plot:

```
proc means data=sashelp.class mean noprint nway;
class age;
var height;
output out=MeanHeightByAge mean=MeanHeight;
run;

/* include the same GOPTIONS statements
   and
   same TITLE1 & FOOTNOTE statements
   as for the Bar Chart */

title2 font='Georgia' height=5 PCT
  justify=LEFT color=CX000000 '  Figure 6: ' color=CX0000FF 'Custom Plot'
  justify=LEFT height=2.5 PCT " ";

title3 font='Georgia' height=5 PCT color=CX000000
'Mean Height of Students in SASHELP.CLASS By Age';

symbol1
  value=dot /* other possibilities include POINT, NONE, etc. */
  height=1.5
  interpol=join /* connect the dots */
  line=1 /* solid line */
  width=2 /* thicken it */
  color=CX0000FF; /* browser-safe blue */

symbol2
  value=none /* suppress the marker */
  interpol=needle /* line from the axis to the marker */
  line=33 /* finest dotted line */
  color=CX000000; /* browser-safe black */

axis1 label=none major=none minor=none style=0;
axis2 label=none major=none minor=none style=0 offset=(2 PCT, 2 PCT);

/* For this data, there was no need to use the ORDER= and VALUE=
   axis tick mark value controls available for the AXIS statement. */

proc gplot data=MeanHeightByAge;

* PLOT  statement defines a plot using the Left-Hand-Side vertical axis *;
plot MeanHeight*age=1 /* use SYMBOL1 statement for this plot */
  /
  vaxis=axis1 /* AXIS1 statement defines the vertical   axis          */
  haxis=axis2 /* AXIS2 statement defines the horizontal axis          */
  autovref /* a reference line at every vertical axis tick mark value */
  lvref=33; /* linetype 33 for vertical axis reference lines          */
  /* (You can use cvref= to customize color of reference lines.)      */

* PLOT2 statement defines a plot using the Right-Hand-Side vertical axis *;
plot2 MeanHeight*age=2 /* use SYMBOL2 statement for this plot */
  / vaxis=axis1;

/* For the author's preferred alternatives to this Custom Plot,
   please see Reference 1. */

run; quit;
```

Here is the code to create the Default Plot:

```
/* three title statements */
proc gplot data=MeanHeightByAge;
plot MeanHeight*age;
run; quit;
```

SAS/GRAPH Device Drivers

SASPRTC is the vendor's current recommended driver for PDF graphs, replacing PDFC in that distinction. For gray-shade (monochrome black-and-white) graphs, SASPRTG (SASPRTM) is the equivalent of SASPRTC. I also had success with the GIF driver for PDF graphs.

PNG is the driver with which I had success for RTF graphs.

GIF is my recommended driver for HTML (web) graphs and for graphs that will be exported to Microsoft PowerPoint or Microsoft Word. I used the GIF driver for Figures 1-6. For me the GIF driver has performed reliably. Except for photographs, GIF is the most widely used image file type on the web. All web browsers and PCs work with GIF.

GIF files are small. They will download quickly if staged for the web. The GIF driver can optionally produce transparent images, which means that the graph elements and the text are opaque, but web page background can show through. The related GIFANIM driver can be used to create animated graphs, which work as expected if imbedded in a web page or a PowerPoint slide (See Reference 3 for an example of web animation.)

I have used SAS/GRAPH since 1981. During more than a quarter of a century, I have had to work with many different types of output, display devices, printers, computer types (mainframes, mid-range / UNIX, and PCs), operating systems, and versions/releases of SAS/GRAPH. I have been forced to deal with numerous different device drivers, including even third-party (i.e., non-SAS-provided) device drivers. The GIF device driver has been the most dependable.

Among the drivers **not** shown in this paper's examples are SASEMF, JPEG, Java, ActiveX, JAVA Image, and ActiveX Image, all of which are SAS Enterprise Guide Graph Results choices.

The SASEMF (and EMF) driver did not perform to my satisfaction. To explain why would be too long and complicated. You may have better luck, but I cannot recommend them.

The JPEG image format was developed by the Joint Photographic Experts Group. I do not know why there is a SAS/GRAPH JPEG driver. Graphs are not like continuous-tone photographs.

The Java & ActiveX drivers impose special requirements on the PC that will view their interactive output. If you have a business case to use these drivers, you must ensure that any PC used to display your graphs meets the requirements. How do you do that? Also, you will find that some features in SAS/GRAPH are not supported by the Java and ActiveX drivers. (Which features are unsupported is documented. For SAS 8, see Technical Notes TS-601 and TS-602 at support.sas.com. For SAS 9, see the SAS/GRAPH manual.) I prefer device drivers that support all SAS/GRAPH functions and features and whose output will display correctly on any PC.

However, if you look into the matter of Accessibility (discussed in a later section), you may conclude that ActiveX is an appropriate and important device driver, despite its requirements for the user's PC/laptop and its graphic function limitations.

The Java Image and ActiveX Image drivers generate non-interactive PNG files with Java and ActiveX technology. Unfortunately, the font support in these drivers is not equal to that of the SAS/GRAPH PNG device driver. I can think of no reason to use them.

Creating Graphic Stream Files on Disk for Insert to Microsoft PowerPoint or Microsoft Word

To route a graphic image file directly to disk, rather than use a SAS/GRAPH Export function from the OUTPUT window, is very simple. (Figures 1-6 were created as Graphic Stream Files, and then were inserted in this paper.) Before the graph creation code, use these statements:

```
ODS LISTING; /* Even though you may not be creating any tabular report (i.e., LISTING)
output, you must turn on the LISTING destination in order to create GSF output. */
ODS NORESULTS; /* Shut off the OUTPUT window. */
GOPTIONS GSFNAME=AnyName GSFMODE=REPLACE;
FILENAME AnyName "PathToTheFile\AnyFileName.FileExt";
```

FileExt is the three-character file extension appropriate to the graphic device driver that you are going to use (e.g., for DEVICE=GIF, use GIF as the file extension).

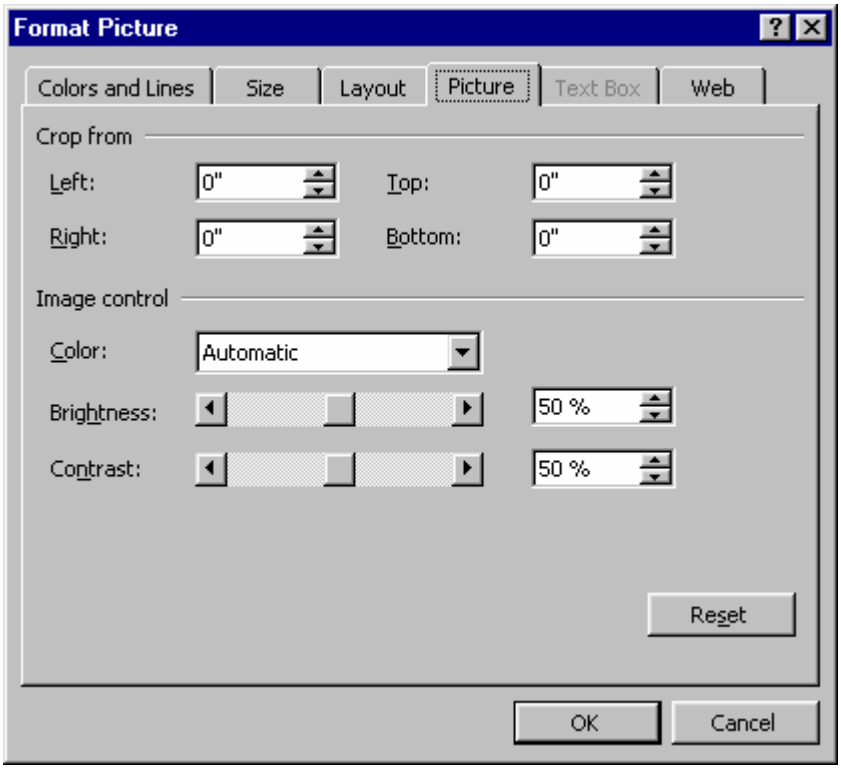
The PathToTheFile could be: `C:\AnyFolderName`

for which case the SAS log would contain a message like:

```
NOTE: 65 RECORDS WRITTEN TO C:\AnyFolderName\AnyFileName.GIF
```

Working with SAS/GRAPH Output in Microsoft Word

First, place your cursor at the desired location in the document. To import SAS/GRAPH output, use Insert > Picture > From File. When you have the file in your Word document, where it will have been automatically shrunk to fit (but preserving its aspect ratio) within the margins, you can right-click on the image to bring up this panel:



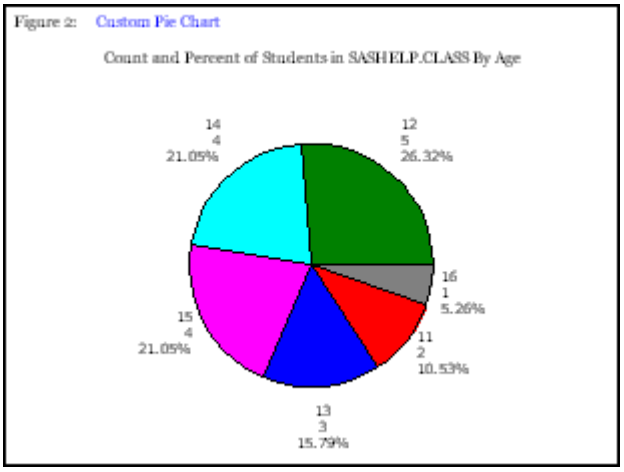
The Picture tab comes up in front, and you could use it to crop (trim the edges of) the image. The Layout tab is used if you wish to wrap text around an image that does not spread to the margins.

You can use the Size tab to shrink the image, presumably with “Lock aspect ratio” checked. An image may still be readable after half-sizing. At 3.2 inches, an available page display width of 6.5 inches will leave room to insert an inter-image gap with the keyboard Space bar between a pair of side-by-side images as done in this paper below.

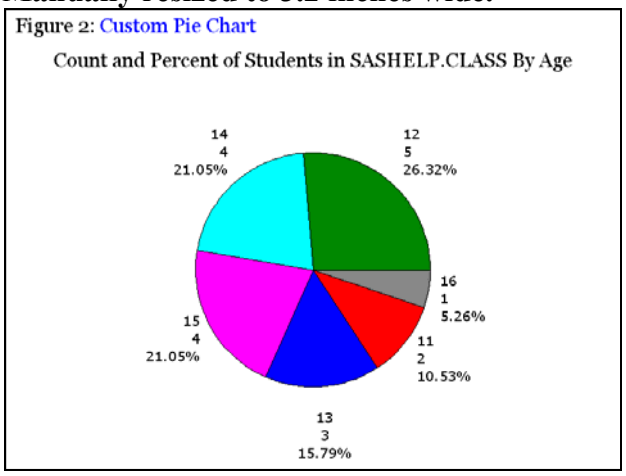
After right-clicking on the image, you can select Borders and Shading, and use the Borders tab to apply a Box around the image. Adding this extra border to an image with a SAS/GRAPH border has, I think, a pleasing visual effect, and has been applied to all figures in this paper.

This paper (before conversion to PDF) was created with Word, with available page width of 6.5 inches. Below are seven GIF images. Four of them compare images manually reduced in Word with the same images created at a little less than half of the page width (3.2 inches) by specifying “GOPTIONS XPIXELS=307 YPIXELS=230”. The next two compare Georgia and Verdana fonts after manual image reduction. The last image is simply auto-fit by Word to page width.

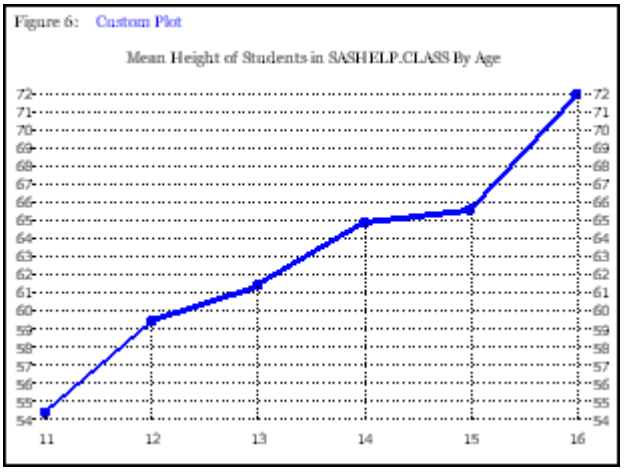
Created at 3.2 inches wide.



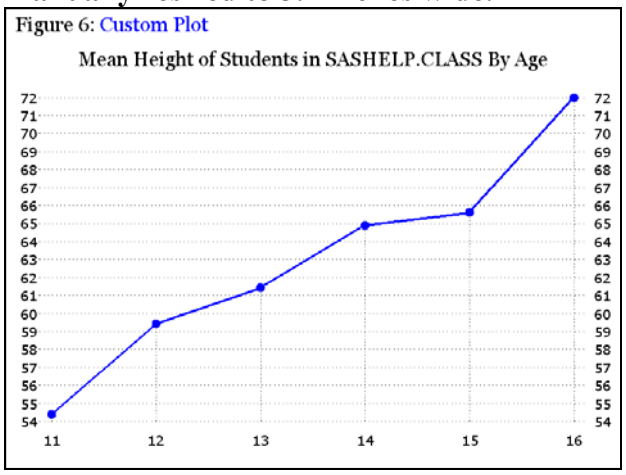
Manually resized to 3.2 inches wide.



Created at 3.2 inches wide.



Manually resized to 3.2 inches wide.



Above, manually resized graphic elements look better than those created at half-page width. The figures below focus on the effect of resizing of message-critical content: text in a graphic image.

This Georgia font text chart was created
with the SAS/GRAPH GIF driver,
and inserted in this paper
composed with Microsoft Word.

After insertion, this image
was manually resized
by using the Word Format Picture function.

Using Format Picture > Size in Word now shows:
original image is 8.33 inches X 6.25 inches,
displayed size is 3.2 inches X 2.4 inches,
image is displayed at 38% of original size.

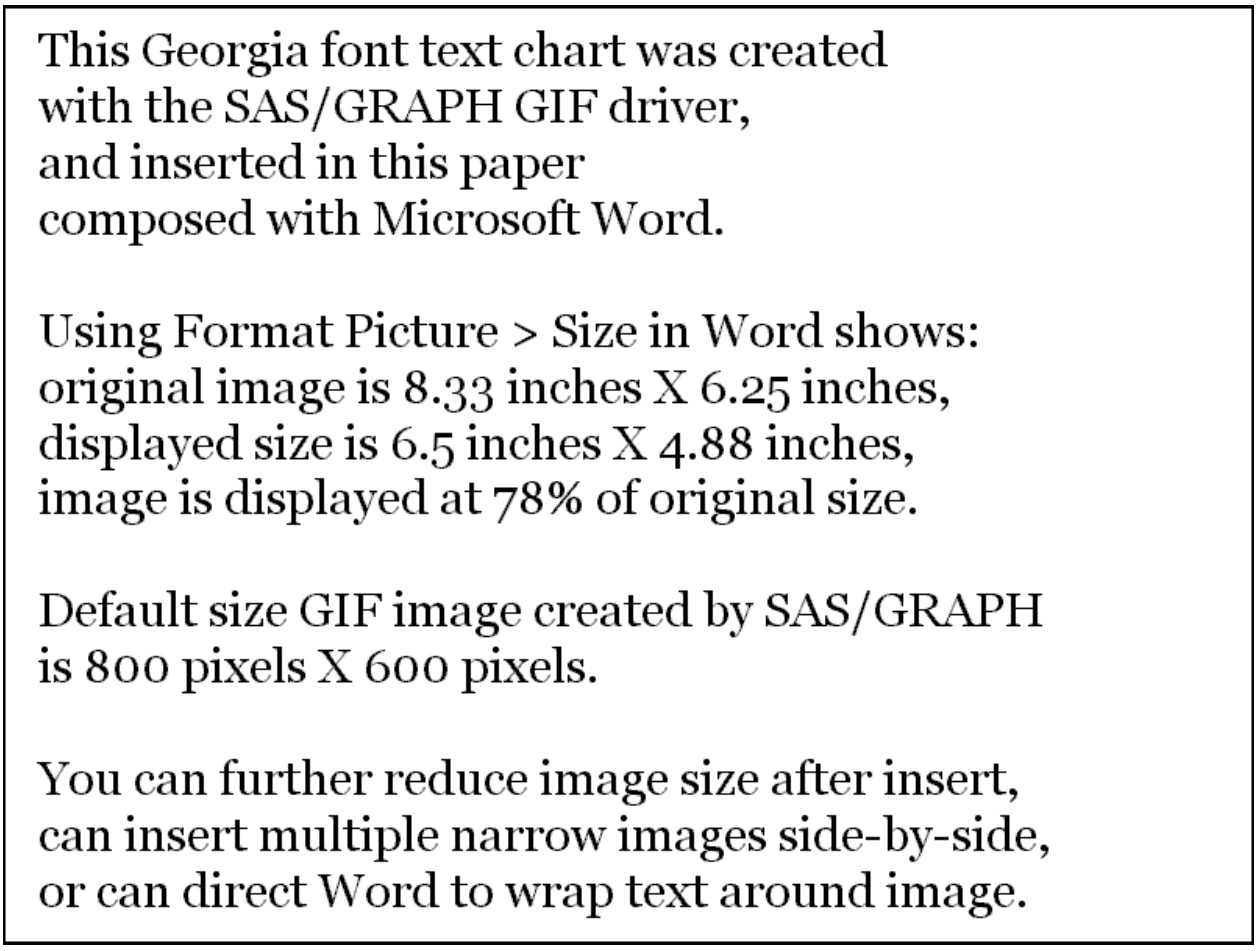
Default size GIF image created by SAS/GRAPH
is 800 pixels X 600 pixels.

This Verdana font text chart was created
with the SAS/GRAPH GIF driver,
and inserted in this paper
composed with Microsoft Word.

After insertion, this image
was manually resized
by using the Word Format Picture function.

Using Format Picture > Size in Word now shows:
original image is 8.33 inches X 6.25 inches,
displayed size is 3.2 inches X 2.4 inches,
image is displayed at 38% of original size.

Default size GIF image created by SAS/GRAPH
is 800 pixels X 600 pixels.



My Conclusions:

1. It is safer to create a GIF image at default size with the SAS/GRAPH GIF driver and resize it inside Microsoft Word than to right-size the original with GOPTIONS modifications at driver invocation. Since the image insertion in a Word document is a manual operation, taking a very few more manual steps to resize the image is simpler and more reliable than going through extra coding rigmarole to TRY to get an acceptable half-page-width original image file.

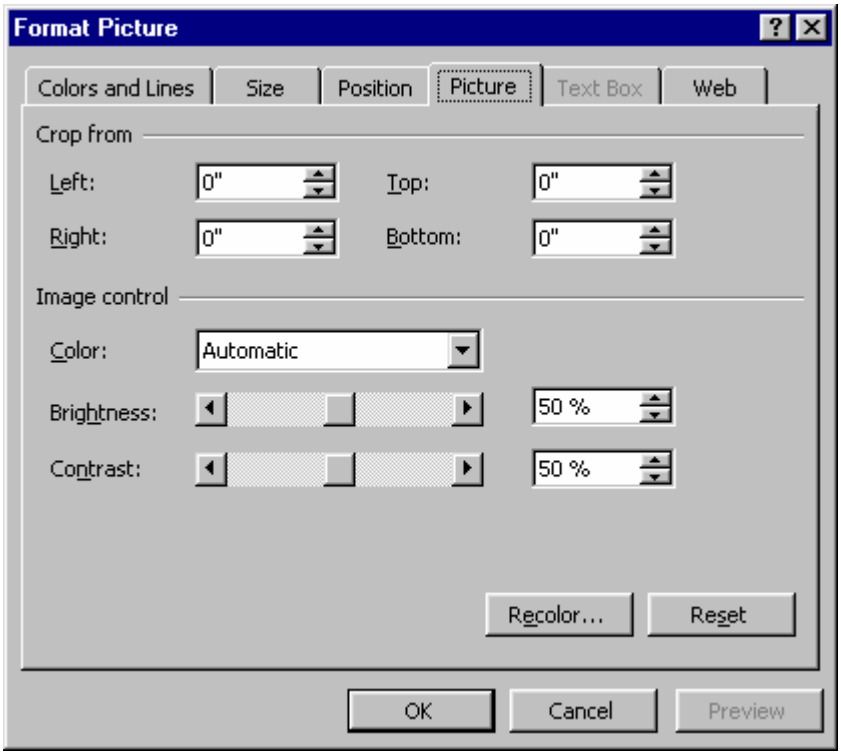
2. Just as Verdana is more readable than Georgia for small text parts of a graph, so, too, Verdana is more readable than Georgia for a graph that is manually resized after insertion in Word.

Working with SAS/GRAPH Output in Microsoft PowerPoint

First, open either a new or existing slide presentation, and create an empty slide. To import SAS/GRAPH output, use Insert > Picture > From File. As slide layouts for displaying an image, I have found the "Blank" slide and the "Title Only" slide to be the easiest ones to work with. "Title Only" does permit you to insert an image.

NOTE: It is possible to insert multiple images on a single slide. You simply need to resize and reposition them as desired. Readability of the result depends on design of the original images.

When you have the image in your slide, it will have been automatically shrunk (preserving its aspect ratio) to fit within the PowerPoint default slide margins. (I do not know whether you can permanently change those default margins.) Then you can right-click on the image to bring up this panel:

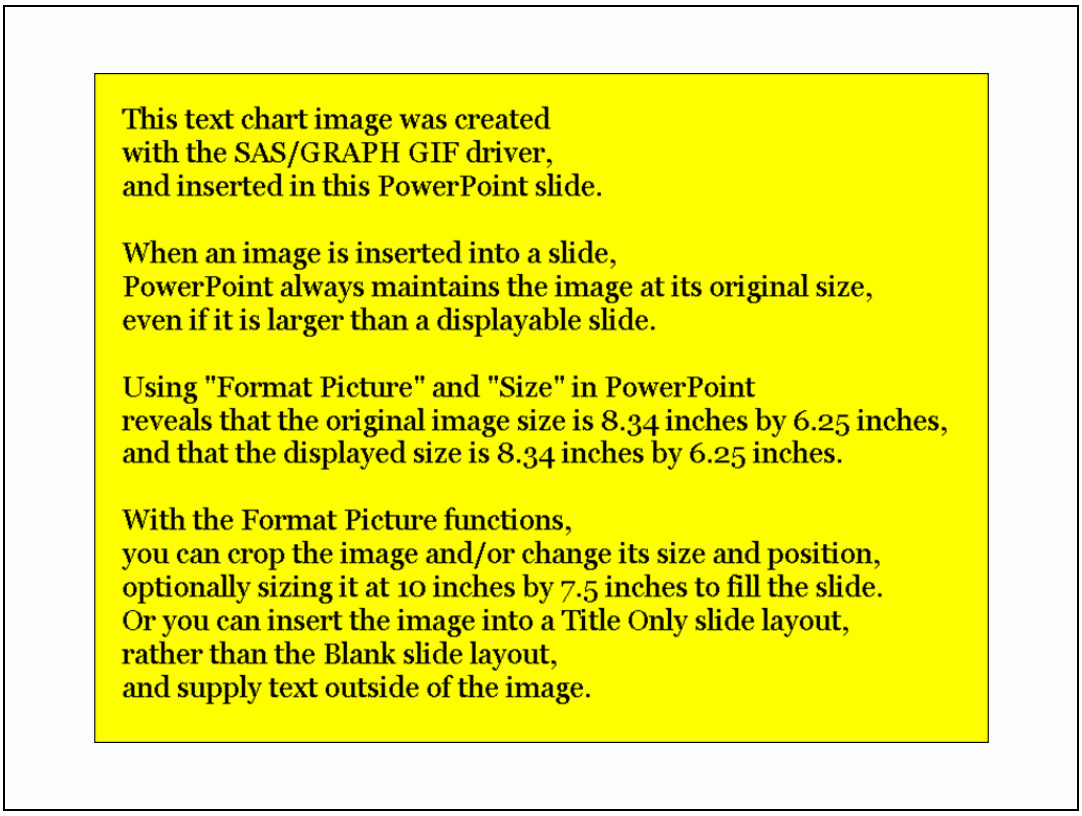


The Picture tab comes up in front, and you could use it to crop the image. The Position tab can be used to relocate the image after resizing. Unless I am using a title that I type into PowerPoint, I usually use the Size tab to expand the image to fill the slide, with "Lock aspect ratio" checked. So far, I have never found image quality to degrade after expansion to fill the slide.

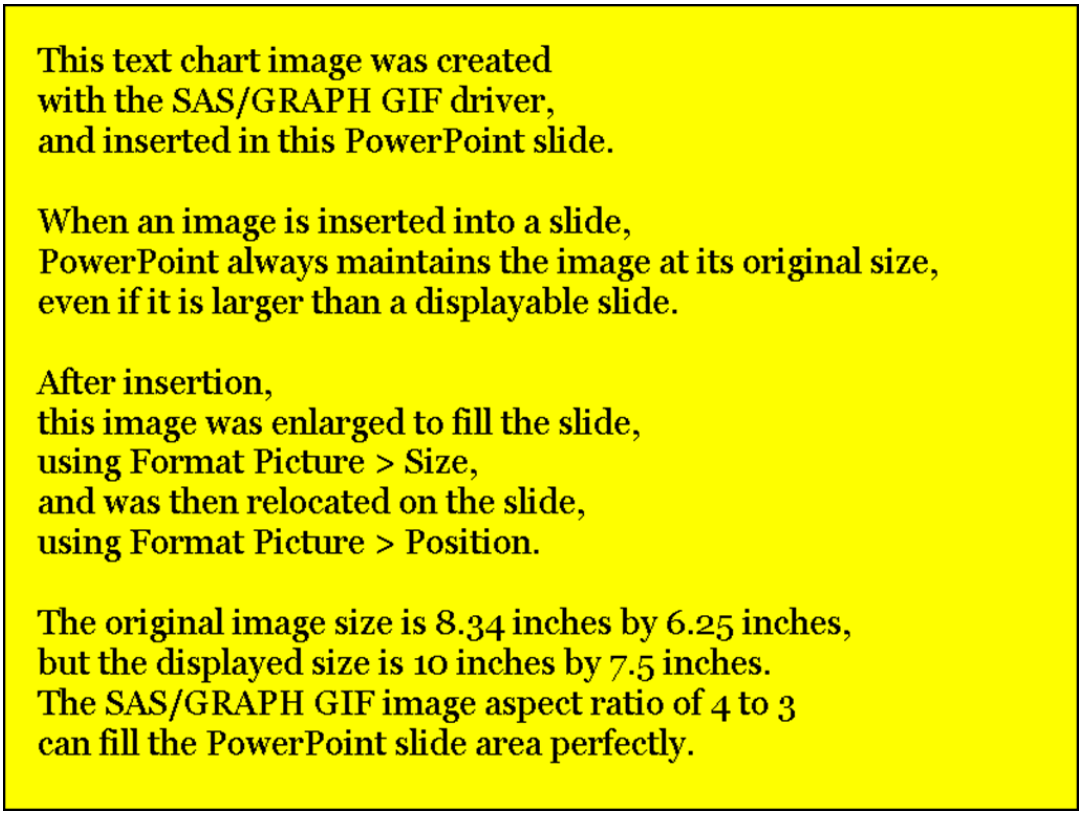
The full size of the display space for PowerPoint slides is 10 inches wide by 7.5 inches high. These dimensions respect the 4-to-3 ratio of traditional, non-wide-screen computer monitors, and of projection screens. **SAS/GRAPH GIF image files, with their 4-to-3 aspect ratio, are "PowerPoint-Ready".**

Below are screen prints from PowerPoint slide displays, whose imbedded images were created with PROC GSLIDE. The same dimension effects apply to all SAS/GRAPH procedure output.

Screen print of a PowerPoint slide without resizing the inserted image:



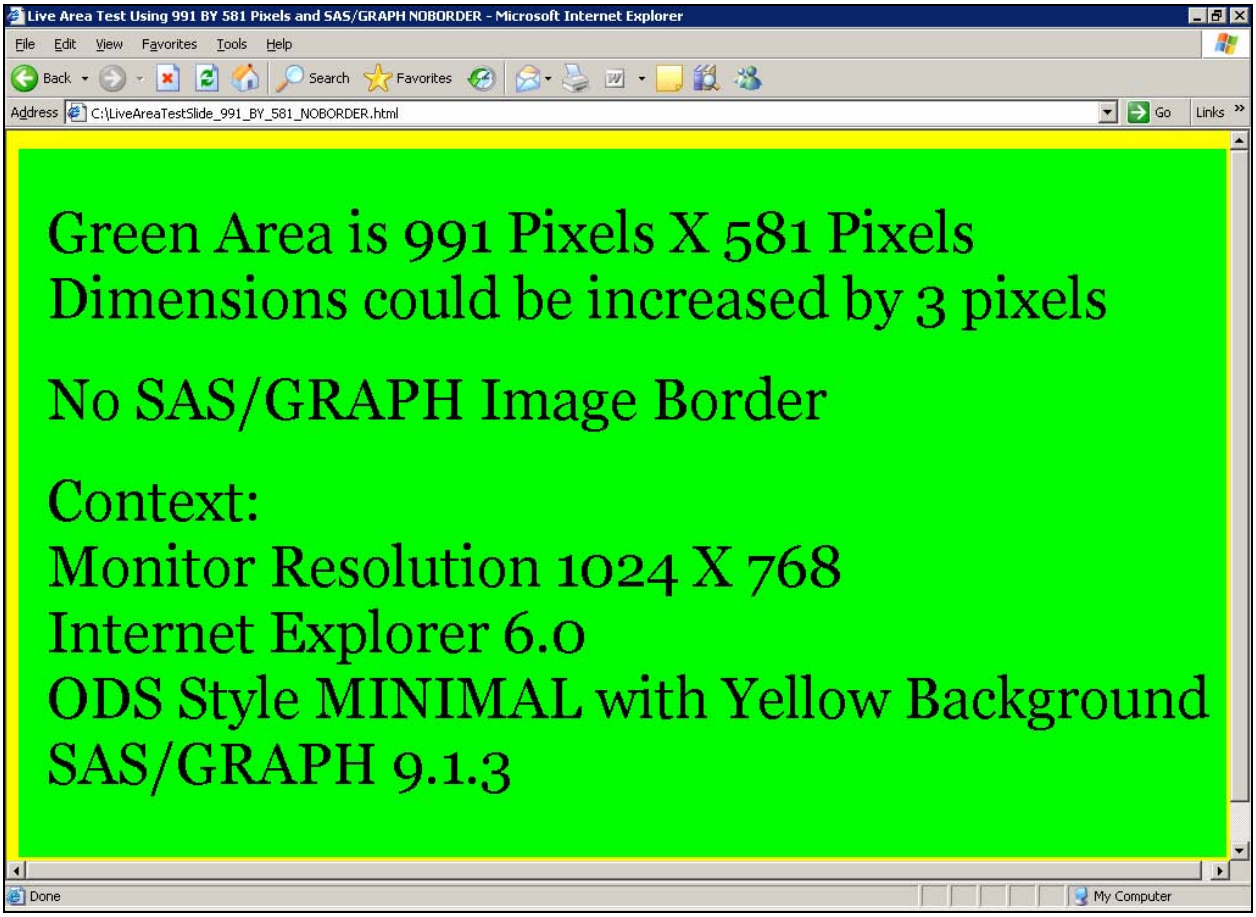
Screen print of a PowerPoint slide with image manually resized to fill the slide, with no loss of text quality:



ODS Destination HTML:
Optimizing your use of web page space (a.k.a. "Live Area" or "Live Space")

Web page users do not really like to scroll. In almost all cases, a web graph should be sized so that it can be seen in full without scrolling. Graphs that are needlessly designed to require scrolling are irritating, prevent getting what could have been a coherent "full visual impression", and are apt to be unprintable as a convenient unit. At the same time, to maximize readability, graphs should be designed as large as will fit in the space available.

Since Reference 1 covers web design and the ODS HTML destination more broadly, here my focus is on sizing a graph to make maximum use of the space inside the web browser frame, and below any optional toolbar(s) that the web user may have enabled. The SAS/GRAPH GSLIDE output in the web page below is deliberately under-sized only so that you can see the right-hand-side and bottom edges of the image where the yellow web page background is peeking out.



"Live Area" or "Live Space" is the area available inside the browser window to display web page content. The frame used by Windows itself and the frame used by a web browser reduce the usable display space of a CRT monitor or a flat panel display.

The size of the live area is affected by the screen resolution of the CRT monitor or flat panel. The commonest resolution in use today is 1024 pixels (width) by 768 pixels (height), with an aspect ratio of 4 X 3.

You can determine the resolution of your own screen on a Windows machine by clicking Start > Settings > Control Panel > Display Properties > Settings. The Screen Area slider shows your current setting, which you can change.

Other factors that affect the live space are the choice of web browser, the version of the web browser, and the absence or presence of optional toolbars, such as those for Google or Yahoo.

Within that context, the available space for your image is affected by the choice of ODS style (e.g., Minimal vs. Default) and the version of SAS. Also, SAS 8 and SAS 9 differ in the version of HTML language used, as well as in how that HTML language is used. (For SAS 8.2, which creates a larger top margin above the image in the live area, the example above would be limited to pixel dimensions of 991 by 551.)

Similar to the size of the available live area is measured in pixels, the size of a GIF graphic image is measured in pixels. Its size is controlled with GOPTIONS XPIXELS and YPIXELS.

Here is the code used to create the test web page above:

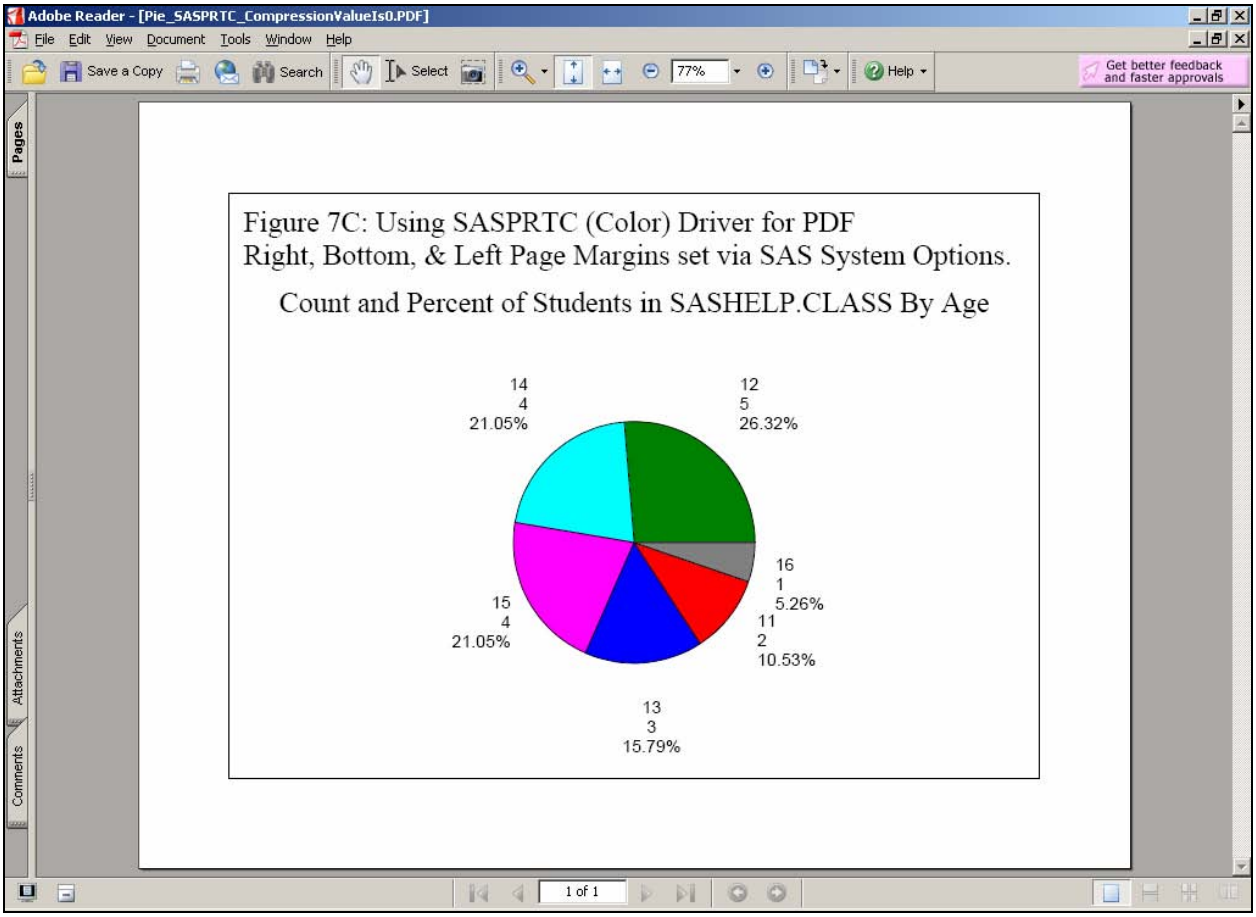
```
proc template;
edit Styles.Minimal AS Styles.Minimal_CXFFFF00;
  style body / background = CXFFFF00; /* browser-safe yellow */
end;
run;

ODS NORESULTS;
ODS LISTING CLOSE;

ODS HTML PATH="C:\" (URL = NONE)
         FILE="LiveAreaTestSlide_991_BY_581_NOBORDER.html"
         (TITLE=" Live Area Test Using 991 BY 581 Pixels and SAS/GRAPH NOBORDER")
         STYLE=Styles.Minimal_CXFFFF00
         GTITLE GFOOTNOTE;

goptions reset=all;
goptions device=GIF;
goptions noborder;
goptions cback=CX00FF00; /* browser-safe green */
goptions XPIXELS=991 YPIXELS=581;

proc gslide;
title1 . . . ;
run; quit;

ODS HTML CLOSE;
ODS LISTING;
```

You should design your web pages for what you expect to be the typical worst-case situation of your intended web audience. During development, you should configure your own screen resolution and web browser to match those of your target.

NOTE: Though the example above shows how to fill the live space almost completely, it is prudent to instead leave a reasonable amount of extra space unused, if the resulting smaller graph is still readable. Then web users with less live space than your expected target configuration may be able to see the whole picture without scrolling.

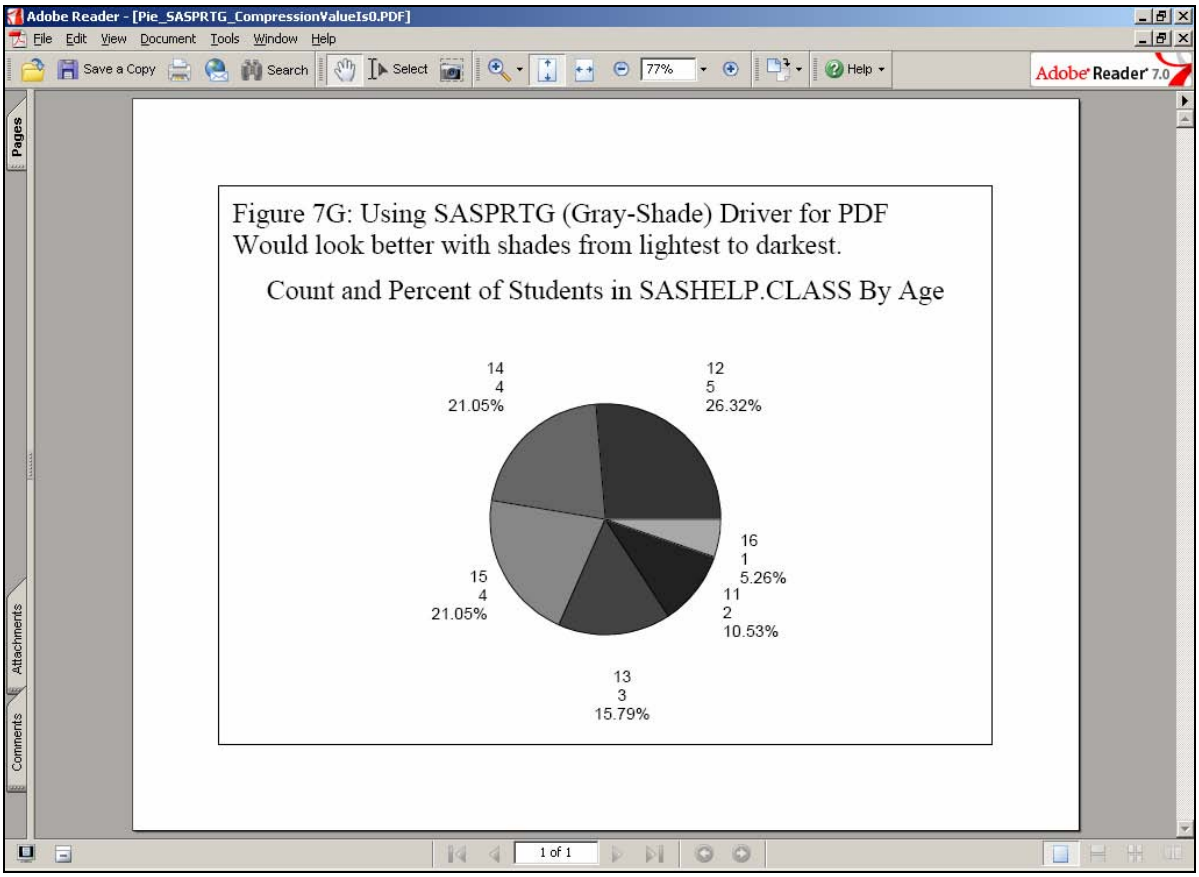
ODS Destination PDF

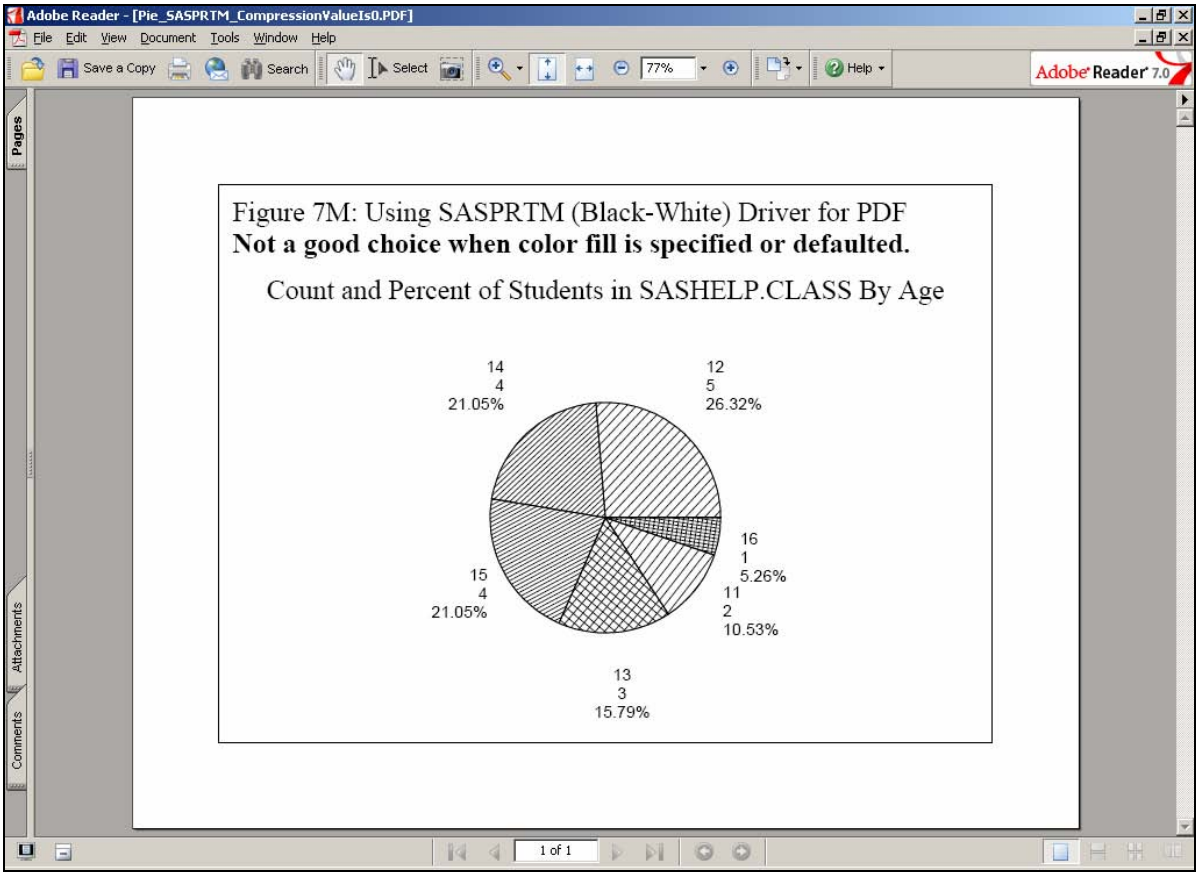
The key points shown are: (a) driver selection; (b) ODS packaging; (c) margin controls; (d) how to turn off the Table of Contents; and compression. Note: for PDF you are limited to three fonts: Times, Helvetica, and Courier, plus their bold, italic, and bold italic versions, coded, e.g., as 'Times/Bold/Italic'. SAS software fonts are also available, but not recommended by me.




```
OPTIONS ORIENTATION=LANDSCAPE;
OPTIONS RIGHTMARGIN=1 IN BOTTOMMARGIN=1 IN LEFTMARGIN=1 IN; /* Top Margin automatic */
ODS NORESULTS;
ODS LISTING CLOSE;
ODS PDF FILE="C:\AnyFolder\AnyFileName.PDF" NOTOC /* turn off Table of Contents */
  TITLE="Custom Pie Chart Using SASPRTC Driver for PDF" AUTHOR="LeRoy Bessler PhD"
  SUBJECT="Demonstrate PDF Pie Chart with SASPRTC Driver"
  KEYWORDS="PDF SAS/GRAPH ODS 'SASPRTC Device Driver'"
  COMPRESS=0; /* No Compression. Otherwise, assign value in the range 1-9. I have
              observed little or no improved compression for values beyond 1 or 2. */
  /* accepting default ODS style Styles.Printer */
goptions reset=all;
goptions device=SASPRTC;
goptions border;
goptions ftext='Helvetica' htext=3 PCT;
title1 font='Times' height=5 PCT color=CX000000 justify=LEFT "  Figure 7C: Using
SASPRTC (Color) Driver for PDF" justify=LEFT height=2.5 PCT " ";
title2 font='Times' height=5 PCT
  'Count and Percent of Students in SASHELP.CLASS By Age';
proc gchart data=sashelp.class;
pie age / name="SASPRTC0" discrete noheading descending percent=outside;
run; quit;
ODS PDF CLOSE;
```

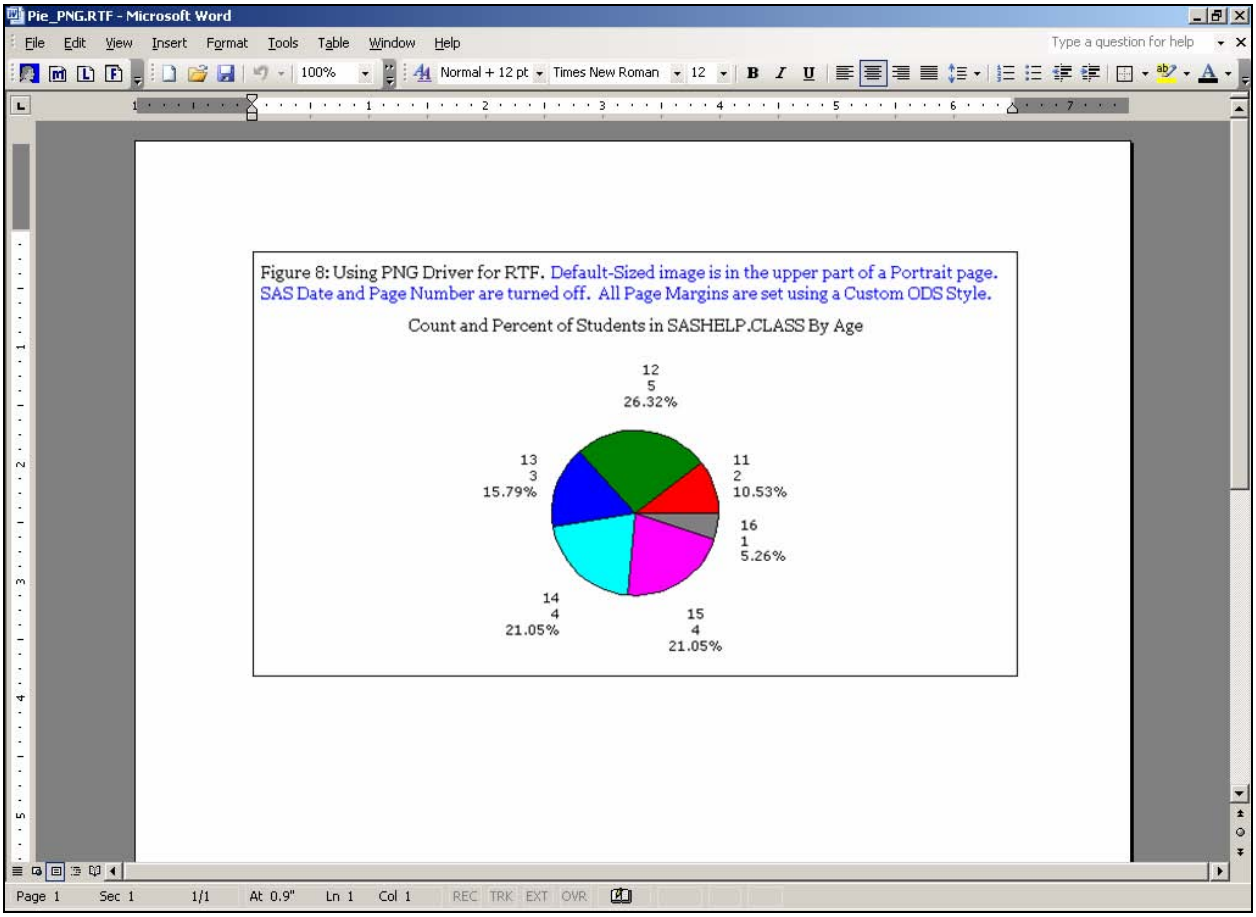
Samples Using the UnColor PDF-compatible Graphic Device Drivers





ODS Destination RTF

The key points shown are: (a) driver selection; (b) ODS packaging; and (c) margin controls;



For the image in the RTF page above, the right, bottom, and left margins are set to 1 inch, but the top margin is set to only 0.75 inch because 0.25 is already reserved for the SAS date and page number displays that have been turned off with OPTIONS NODATE NONUMBER.

The image is created at the default size by the PNG driver, for which HSIZE=6.474 INCHES and VSIZE=3.631 INCHES. This is a fortuitously convenient width for imbedding in a Portrait document with a USA Letter Size page width of 8.5 inches with standard 1-inch margins. But the default height is rather skimpy and makes poor use of the available space.

When you open an RTF file with Microsoft Word, you can right-click on the image and use the Size tab of Format Picture function to resize the image. However, you cannot make the image longer to fill the page without changing the aspect ratio, in which case the text (and any circle) in the graph will be distorted.

For a should-be distortion-proof programmatic solution to produce a PNG image with a larger size, higher resolution, and thicker lines, please see Technical Note TS-674 at support.sas.com, for its discussion of macro %PNGSZ.

Here is the code used to create the RTF document with imbedded PNG image:

```
proc template; /* Create a Custom Style based on the default style designed for RTF */
define style styles.RTFwithOneInchMargins;
parent=styles.rtf;
style body from document /
  topmargin=0.75 in
  rightmargin=1 in
  bottommargin=1 in
  leftmargin=1 in;
end;
run;

OPTIONS ORIENTATION=PORTRAIT;
OPTIONS NODATE NONUMBER;

ODS NORESULTS;
ODS LISTING CLOSE;

ODS RTF
  FILE="C:\AnyFolder\AnyFileName.RTF"
  TITLE="Custom Pie Chart Using PNG Driver for RTF"
  AUTHOR="LeRoy Bessler PhD"
  GTITLE GFOOTNOTE
  STYLE=Styles.RTFwithOneInchMargins;

goptions reset=all;
goptions device=PNG;
goptions border;
goptions ftext='Verdana' htext=3.75 PCT;

title1 height=2.5 PCT ' ';
title2 font='Georgia' height=5 PCT color=CX000000 justify=LEFT
  "  Figure 8: Using PNG Driver for RTF.  "
color=CX0000FF "Default-Sized image is in the upper part of a Portrait page."
justify=LEFT
  "  SAS Date and Page Number are turned off.  All Page Margins are set using a Custom
ODS Style."
justify=LEFT height=2.5 PCT " ";
title3 font='Georgia' height=5 PCT
  'Count and Percent of Students in SASHELP.CLASS By Age';

proc gchart data=sashelp.class;
pie age / discrete noheading percent=outside name="Pie_PNG";
run; quit;

ODS RTF CLOSE;
```

Accessibility

Accessible output on the web is communication-effective for people with impairments—i.e., it is designed and constructed to try to reach all of the possible audience. For more expansive information on Accessibility than I can provide here, please consult Lisa.Pappas@sas.com, do a search at support.sas.com, and do a web-wide search with, say, Google.

To achieve web accessibility of SAS/GRAPH output, one of the solutions, the ActiveX graphic device driver, requires the presence of special software on the web viewer's PC. So, even if a user has no impairments, that special software is required in order to display ActiveX output. Thus, to meet the possible needs of some of your audience with ActiveX, a definite extra software requirement is imposed on all of your audience. On three different PCs, I had three different experiences with ActiveX output, and only one of the experiences was satisfactory.

Nevertheless, there is good news:

(1) There are things that you can do with respect to fonts, colors, and backgrounds to promote accessibility without ActiveX, and most of them also benefit people with no visual impairment.

(2) Some of the ActiveX automatic deliverables can instead be created by explicit coding. These opportunities include provision of ALT text and use of the DES= (DESCRIPTION=) parameter on the PIE statement, BAR statement, PLOT statement, etc. The benefit of coding these yourself is that you can control and enhance the content, rather than take the default. See Reference 1 for details and examples of those design principles and implementation methods.

(3) If you wish to use ActiveX with certainty that you can reach all of your web audience (i.e., users with or without ActiveX support on their PCs), you could build your web-delivered application with a home page that allows the web user to select an ActiveX branch or a GIF branch for viewing the rest of the web package. It would require creating two sets of graphs and two sets of web pages, but that can be worth the effort.

This tutorial made much use of the GIF driver. Other device drivers may have their admirers, but, except for the JPG (aka JPEG) image format, which is really appropriate only for photos, the GIF image is the most widely used format in the world and on the web. The probability of meeting a PC that cannot handle GIF is probably zero. If a programmer can still find a way to meet the needs of impaired users, then GIF qualifies as the most accessible web graphic format.

If your definition of "Accessibility" includes presenting data and information via the web or PDF in ways other than the usual, please see Reference 3, my paper on Multi-Media Wizardry with ODS. You can add more channels and features to information delivery, and even have some fun.

Accessibility of Fonts and Colors

For people with normal or near-normal vision, a designer's or programmer's use of fonts, colors, and backgrounds can limit the communication-effectiveness of data presentation. Please see Reference 1 for guidelines. For an extensive discussion of communication-effective color, please see Reference 4.

Conclusion

Here is a summary of my device driver findings.

- For PDF, I agree with the SAS recommendation of drivers SASPRTC and SASPRTG for color and gray shades. (Formerly, SAS recommended PDFC and PDF, respectively.) I also found the GIF driver to work for PDF, but, not having done exhaustive testing, I can not recommend it over SASPRTC and SASPRTG.

- For RTF, I got acceptable results only with PNG (but not PNG generated with the Java Image driver or the ActiveX Image driver).

- For everything else within the scope of this tutorial, the GIF driver performed either better than, or as well as, any other driver.

Web output can be made accessible with ease, and benefits all users, as well as any impaired.

Let me close by reminding you to beware of the SIMULATE font, and to know how to recognize it (Reference 1 explains how to create a sample) when SAS/GRAPH substitutes it for your requested font, possibly without any notification in the SAS log. Readable text is essential to effective communication. Elegant text is an asset to the visual impression made by any graphic communication. The SIMULATE font can sometimes be hard to read and is never elegant.

References (Related Work by this Author)

1. Get the Best out of SAS/GRAPH and ODS, *Proceedings of the SAS Global Forum 2007.*

2. Communication-Effective Pie Charts, *Proceedings of the SAS Global Forum 2007.*

3. Multi-Media Wizardry: How To Make ODS Outputs That “Dance and Sing”, *Proceedings of the Thirty-First Annual SAS Users Group International Conference*, 2006.

4. Communication-Effective Use of Color for Web Pages, Graphs, Tables, Maps, Text, and Print, *Proceedings of the Twenty-Ninth Annual SAS Users Group International Conference*, 2004.

Acknowledgments

I am grateful to my colleague Alix Riley for her helpful review of this paper, and to SAS Global Forum 2007 Data Presentation Section Chair Tyler Smith for the opportunity to share my ideas with other SAS users.

Contact Information

Your comments, questions, and suggestions are welcome. I am always interested in design ideas or construction solutions that enhance graphic communication.

LeRoy Bessler PhD
Email: bessler@execpc.com
Phone: 1 414 351 6748 (evenings and weekends—time is six hours earlier than Greenwich Mean Time)