Paper DV-03-2014

SAS® Graphs with Multiple Y Axes — Some Useful Tips and Tricks

Soma Ghosh, UnitedHealth Group, Minneapolis, MN

ABSTRACT

SAS/Graph is a very powerful feature that helps programmers, analysts to provide a very high standard view of data. Without having this feature, report writers had to use some other applications to create graph manually every time, with every new datasets. But this helps to automate the design view of many analytical data available in SAS. With SAS graphics, program needs to be created only once and graphs will be created automatically for all similar datasets. This paper will explain how to create graph with double values in the Y axes and also combining Bar and Line graph together in one Chart. To define the entire graphical process, GPLOT & GCHART, PROC GREPLAY are used. These procedures will help in merging two different graphs together and saving file in jpeg format. Also, this paper includes the usage of attributes like COLOR, STYLE, FONT and ANNOTATE DATA step with graphs.

INTRODUCTION

Throughout this paper sample code snippets are provided to elaborate the entire process of creating SAS Graph with double Y axes.

Dataset referred is “sample_demo”, which is a copy of sashelp.demographics. Columns used to plot the graphs are NAME (Country Name), popAGR, MaleSchoolpct, FemaleSchoolpct.from sashelp.demographics.

Following graphical output will be generated with the help of examples provided in this paper.

1. **A bar graph will be created for maleschoolpct, femaleschoolpct using GCHART.**
   Bar graph will be plotted with left Y axis and X axis

2. **A line graph will be created for popagr using GPLOT.**
   Line graph will be an overlay chart, will be plotted with Bar chart with right Y axis.

Bar and line chart will be merged using GREPLAY.

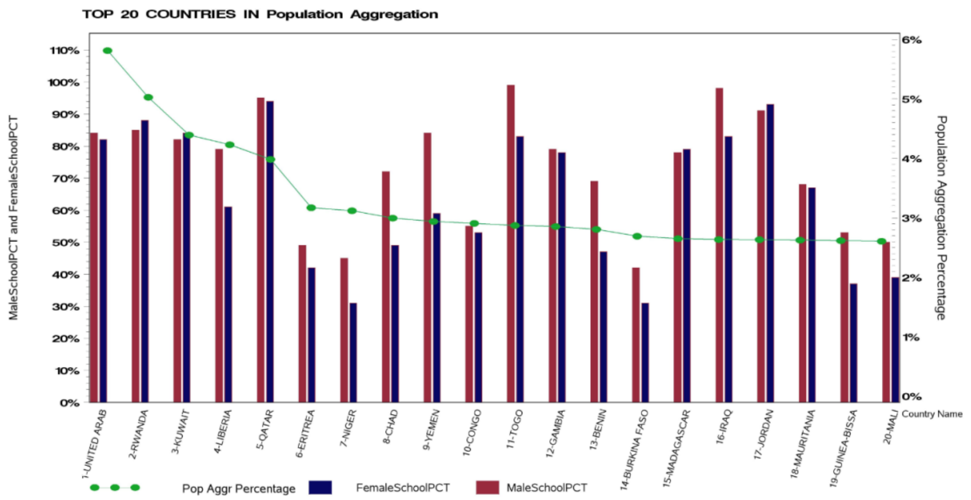

Figure 1. Diagram of output generated through SAS graphics – Multiple Y axes

SOME KEY PROCEDURE IN SAS GRAPHICS

GPLOT:

The GPLOT procedure plots the graph of multiple values based on X and Y axes. It can plot graphs with different pictorial view and gives meaning to underlying data. It reads the value from different observations from SAS data set based on two or more columns. This procedure has features of creating all axes uniformly scaled and can utilize ANNOTATE data set.

Some other features are:

- Multiple Y axes/plots with a second vertical axis
- Scatter plots
- Overlay plots, in which different data points are displayed on one set of axis
- Bubble plots
- logarithmic plots

GCHART

Using GCHART procedure different pictorial formats can be represented. Those are: block charts, horizontal and vertical bar charts, pie and donut charts, and star charts. It can represent both number and character variables. This is the best way to show values based on statistical calculation like, sum, average, mean, percentage, frequency. It can produce chart based on the values of one or more chart variables.

Creating SAS graph with multiple Y Axes

To produce the graph shown in **Figure 1**, code snippets are provided in different steps. There are 16 steps and each steps explains the functionality used in the code and then the code snippet is included.

The code below shows step by step process of creating SAS graph with multiple Y axes and combination of Line and Bar graph together.

Step 1: Pulling data from the SAS dataset sashelp.demographics.

Following columns are used to plot the graph:

c_name [plotted on x axis] (NAME from sashelp.demographics is replaced to c_name)

popagr [plotted as line graph through right y axis]

maleschoolpct [plotted as bar graph through left y axis]

femaleschoolpct [plotted as bar graph through left y axis]

```
proc sql;
create table sample_demo as select substr(name,1,12) as
c_name,popagr,maleschoolpct,femaleschoolpct from sashelp.demographics
where maleschoolpct ne . and femaleschoolpct ne .;
run;
```

Step 2: Pulling top 20 rows

Data is pulled from dataset created in the above step “sample_demo”. It is creating a new dataset called “sorted_sample”. Top 20 [highest] numbers are pulled for “popagr”

```
proc sql outobs=20;
 create table sorted_sample as select * from sample_demo
 order by popagr desc;
 run;
proc sql;
create table sample as select
  (put(monotonic(),8.) || "-" || c_name) as c_name
  ,popagr
  ,maleschoolpct
  ,femaleschoolpct
from sorted_sample;
run;
```

Step 3: Finding the highest value for Y axis

Finding the highest value among the variables “MaleSchoolpct” and “FemaleSchoolpct”

This would help in determining the highest number in the Y axis (for “MaleSchoolpct” and “FemaleSchoolpct”) and can set the length of the Y axis dynamically instead of hardcoding. To pull the highest number, code is checking maximum number from both the variables (“MaleSchoolpct” and “FemaleSchoolpct”). Whichever variable has the highest value is getting assigned to another variable called “hst”.

```
proc sql;
  create table highest as select
  case when max(maleschoolpct)> max(femaleschoolpct)
  then max(maleschoolpct)*1.15 else
  max(femaleschoolpct)*1.15 end as hst from sample  ;
run;
```

Step 4: Storing the highest value

Highest number that is pulled by comparing two variables in the previous step, is stored in a variable. This is used later in the graph. Among two variables, it will find out the highest value and this will decide the length of the Y Axis.

```
proc sql;
  select hst into : ht from highest;
run;
```

Step 5: Determining the length of Y axis.

This section of the code is dividing the “Height/Length” identified in the previous step with 10 and finding the gaps between the major ticks in the Y axis. Variable “htby” will used in the axis definition. An example is shown here:

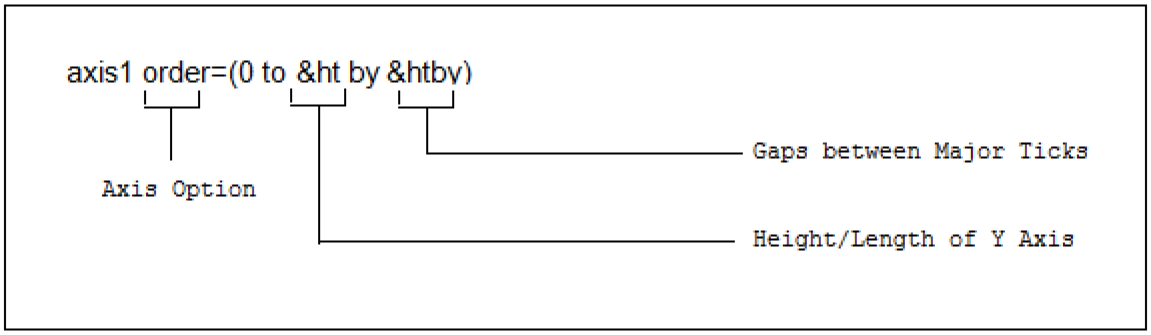

Figure 2. Diagram is showing the usage of variables ht and htby with SAS Graphics- Axis.

CALL SYMPUT can store value in a variable.

```
data _null_;
  set highest;
  by_no = round(hst,10**(int(log10(hst)) )) / 10;
  call symput('htby',by_no);
run;
```

Step 6: Finding the highest value for plotting line graph using right Y axis

This would help in determining the highest number in the right Y axis based on the variable called “popagr" and can set that dynamically instead of hardcoding.

```
proc sql;
   select max(popagr)*1.15 into : lnht from sample;
run;
proc sql;
   create table tlnht as
   select max(popagr)*1.15 as lnhtt from sample;
run;

data _null_;
   set tlnht;
   by_no = round(lnhtt,10**(int(log10(lnhtt)) )) / 10;
   call symput('lnhsby',by_no);
run;
```

Step 7: Grouping the variables.

This step will show how the data is processed. Data is sorted based on Country Name[c_name], which will appear on X axis.

Also this step is showing the creation of two different groups for two different bars.

```
options nocenter;
data gr_c_name1;
    set sample;
run;

proc sort data=gr_c_name1 sortseq=linguistic(numeric_collation=on);
by c_name;
```

```
data gr_c_name3;
set gr_c_name1;
if maleschoolpct ne . then maleschoolpct=maleschoolpct ;
if maleschoolpct ne . then gr='p';
output;
if femaleschoolpct ne . then maleschoolpct=femaleschoolpct ;
if femaleschoolpct ne . then gr='t';
output;

keep c_name popagr maleschoolpct gr;
run;
proc sql;
create table gr_c_name2 as select c_name,popagr,maleschoolpct as result , gr
from gr_c_name3;
run;
proc sort data=gr_c_name2 sortseq=linguistic(numeric_collation=on);
      by c_name;
goptions reset=all device=jpeg htext=8pt nodisplay;
```

In the above statement to initiate the graphics, **GOPTIONS** is used. Along with GOPTIONS, other options used are: RESET, DEVICE AND HTEXT. Functionalities are defined below based on their usage in this paper.

Table 1. Describing the options used with GOPTIONS.

Option	Description
RESET =	This is used to reset/clear all the graphical options. It kills all previously stored graphical options in memory. Once RESET ALL is used, graphical features can be defined using SAS.
DEVICE =	This Is used to select a desired driver to redirect the graphics [output file]. This paper shows an example of a SAS graphical output file, which save the graphics in JPEG format. If no device is defined, SAS by default select driver to show the output
HTEXT =	Determines graphics text height
NODISPLAY=	NODISPLAY to generate graphics which are directed to a file such as a GOUT dataset, which can be replayed later with the DISPLAY option.

Step 8: Defining Axis

This step is demonstrating, how different axis are defined for the line graph, their position, color, label. With the definition of **Axis**, different parameters for graphs can be allocated. Those are:

- Axis Scales
- Axis Position
- Axis Appearance
- Axis Label

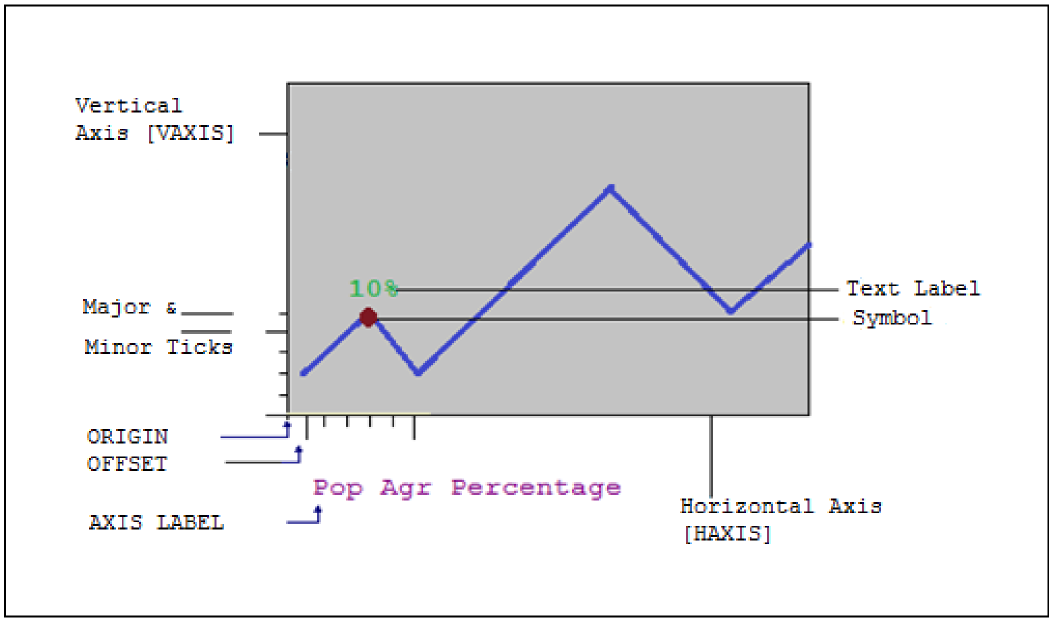

Figure 3. Pictorial view of different parts of graph.

Options, which can be used with Axis:

Option	**Description**
ORDER =	Sets the value of the origin and the value of the end position of axis and it is accompanied with BY to determine the Major Ticks.
ORIGIN =	Sets the origin point in respective of X & Y plot
LENGTH =	Defines the length of the vertical and horizontal axes in PCT [units]
MAJOR =	Point out the major ticks. Tick-mark with sub-option can define the appearance like size, color, etc. If it is NONE, it represents the values associated with each point.
MINOR =	Point out the minor ticks. Tick-mark with sub-option can define the appearance like size, color, etc. If it is NONE, it represents the values associated with each point.
OFFSET =	Sets the length from the first point and last point among major ticks or among the bars.

Using Labels with Axis:

LABELS works with Axis with options like:

None or any text: Given text will appear as label text. [Note] Label text cannot be more than 256 characters.

Text can be defined with other options like font, degrees, text height

Example: LABEL=(a=degree h=numberpt f=font_type "TEXT");

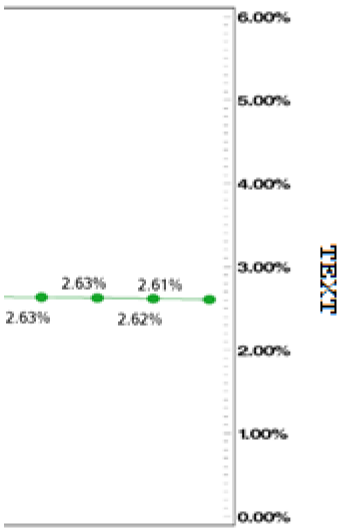

Figure 4. An example of LABEL.

Example of graph used in this paper, also uses the option **SYMBOL**:

SYMBOL statements create SYMBOL, which is used here with GPLOT. SYMBOL can also be used by GBARLINE and GCONTOUR PROC.

SYMBOL handles the appearance of plot symbol and lines, including bars, boxes, area.

In this paper, the example used to plot the graph uses symbol to connect the lines in line chart and also to show legend symbol with line join.

Example: symbol color='RED' INTERPOL=join LINE=1 WIDTH=1 POINTLABEL=(height=10pt '#TEXT');

There are several options, which can be defined with SYMBOL. Those are as follows:

Table 2. Describing the options used with SYMBOL.

Option	**Description**
LINE=	Sets type of line, e.g., solid, dashed, etc.
VALUE=	Sets the joining points, e.g., dot
WIDTH=	Sets the width of the line, default value is 1
INTERPOL=	This works with lot of options. In this example, JOIN is used, which connects data points with straight lines.
COLOR=	Sets the COLOR of the symbol
POINTLABEL=	Defines plot point labels.

The following code incorporates the different options with AXIS.

```
axis1 order=(0 to &lnht by &lnhsby) value=none origin=(10.8,20) pct
length=70 pct
label=none major=none minor=none;
axis2 order=(1 to 20 by 1) value=none origin=(10.8,21.1) pct length=80 pct
label=none major=none minor=none offset=(1.8,1.8);
axis3 order=(0 to &lnht by &lnhsby) origin=(90,20) pct length=70 pct
value=(f=swissb)
label=(a=270 h=11pt f=arial "Population Aggregation Percentage");
title f=swissb h=9pt move=(10,39) "top 20 countries in Population Aggregation";
symbol1 color='cx16a629' interpol=join /*sets color and joins markers in a
line*/
line=1 /*sets type of line e.g. solid, dashed, etc*/
width=1 value=dot; /*sets width and assigns unique symbol*/
symbol2 color='cx16a629' interpol=join line=1 width=1;
```

[Optional]Along with the Symbol2, option can be included to add plot point labels.. E.g. `symbol2 color='cx16a629' interpol=join line=1 width=1 pointlabel=(height=10pt '#pop agr');`

Step 9: Defining graph legends.

For the SAS Graphs, legend can be defined separately

Options which can be used with LEGENDS are as follows:

Table 3. Describing the options used with LEGENDS.

Option	Description
ACROSS =	Defines whether the legend should appear vertically or horizontally
DOWN=	Functions like ACROSS
POSITION =	Location of the legend can be defined. [example given in the below code snippet]
OFFSET=	Used to change the position of the legend. It can be defined as OFFSET=(X,Y) where X would be horizontal spacing and Y would be vertical spacing. Spacing can be defined in units with Legend offset. It could be in pct, cm.

In the code snippet provided below, there is no value for Y with offset, which means Y will be considered as 0. With option value, caption of the legend can be defined, with height of text, font, alignment and text label.

```
legend1 across=2
down=1
cborder=white
position=(bottom center outside)
mode=protect
label=none
offset=(-3cm,)
value=(h=10.2pt f=arial justify=center ' Pop Aggr Percentage’);
```

Step 10: Creating the Line Graph

With PROC GPLOT, dataset used for line graph is assigned and further statements like PLOT are defined to create the graphical structure.

Along with PROC GPLOT, GOUT is used to define the SAS catalog, which will be the physical location of the graphics output. The graphics output which is saved, can be redirected to a desired location.

This graphics output can be specified with LIBREF, if no LIBREF is defined, graphics will be saved in the work directory.

PROC GLOT with option DATA identifies the dataset which can be used to plot the graph. Further visualization can be provided to plot using axis statement.

While creating the Line Bar, the axes can be defined to incorporate data used in the graph. Axis definition provides meaning to the graph. If the code is defining the axis, then the default axis values will be overwritten. This can be achieved by modifying the PLOT statement [example can be found in the code given below]. With the PLOT statement VAXIS and HAXIS can be defined after adding a slash "/". VAXIS is the vertical axis, which means Y axis and HAXIS is the horizontal axis, which means X axis.

Along with PLOT, legends can also be included as options.

Legend can have options for position, font, color, alignment for the text [Code above has the definition for Legend-named as legend1]. Legend1 is used with PLOT statement followed by VAXIS and HAXIS.

```
proc gplot gout=work.gseg34 data=GR_C_NAME1;
format POPAGR percent6.0;
plot POPAGR* C_NAME / vaxis=axis1 haxis=axis2 legend=legend1;
plot2 POPAGR* C_NAME / vaxis=axis3 haxis=axis2 legend=legend1;
run;
quit;
```

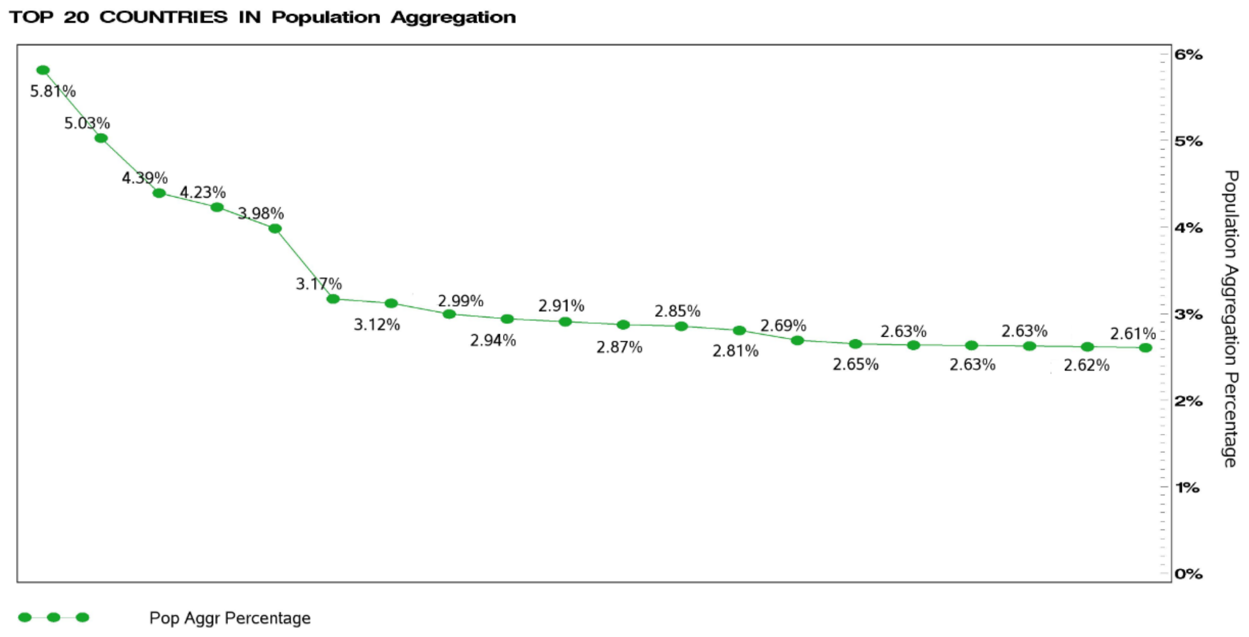

Figure 5. Sample of the Line Graph created by the code defined above

Step 11: Defining the legends for the Bar Graph

This section will add legends to the data related to bar graph. This is showing how Annotate data set can be used with SAS/Graph to enhance standard of the output. Annotate data set holds functional detail for the graph. It has various functionalities; however this paper shows an example of adding legends with overlay of line plot on a bar chart. Where there is overlay plots, annotate data set helps to define the exact position of the labels or the symbols. Annotation follows the technique of certain variables and those variables are defined using the length statement, those are:

FUNCTION, X, Y, COLOR, STYLE, TEXT

In general term this can be defined very easily as, annotate instructs:

- *What has to be done [Function]*
- *Where to apply [X, Y, XSYS, YSYS]*
- *Which Color, Size and Style*
- *What is the Label*

The variables defined in the Annotate data set, translates the information into the graphs. It's just like a decorative wedding cake, where presentation is a very important thing.

Function is a very key variable defined for annotate dataset. The character variable FUNCTION provides the information on *What has to be done*. This key variable is an important component defined with every observation.

In this paper, annotate data is created for the legend shown in the picture given below.

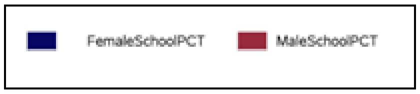

Figure 6. Diagram shows an example where data set "anno" is used to ANNOTATE

Below defined annotate dataset is defining the position, color, text for two boxes and two labels. Graphical view can be seen in Figure 6.

The XSYS and YSYS variables define the area enclosed by the frame. It defines the coordinates for any frame. To draw a frame that encloses the entire graphics output area, XSYS and YSYS are assigned with a value of 5. So, the frame is covering the entire graphical area and within the frame, different overlaying plots are printed and data annotate used for bar graph's legend.

It is not required to keep constant values for XSYS and YSYS; also both of these variables can have different values.

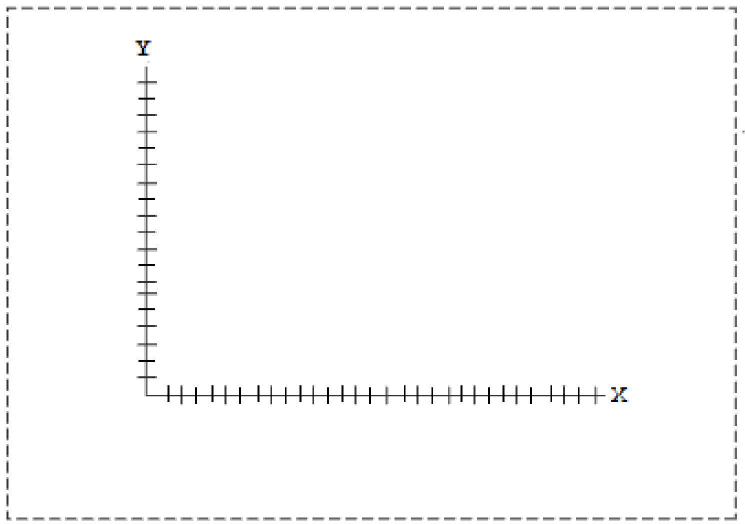

Figure 7. Above diagram is showing the FRAME created with XSYS=5 and YSYS=5

```
data anno;
length function color style $8 position $1 text $28;
retain xsys '5' ysys '5' when 'a';
function='move'; x=49; y=2.6; output;
function='bar'; x=51.2; y=5.1; color='CX99293D'; style='solid'; output;
function='move'; x=56; y=1.9;output;
function='label'; style='arial'; text='MaleSchoolPCT'; color='black'; size=.9;
position='2';output;
function='move'; x=32.5; y=2.6;output;
function='bar'; x=34.7; y=5.1;color='CX090766'; style='solid'; output;
function='move'; x=41.8; y=1.9;output;
function='label'; style='arial'; text='FemaleSchoolPCT'; color='black';
size=.9;
position='2';output;
run;

goptions reset=all device=jpeg htext=8pt nodisplay;
```

Step 12: Creating the axis for the Bar Graph

This section is defining the axes of Bar Graph. It is working with the following options:

ORDER: determines the length of the axis

ORIGIN: start and end position. Defined in **Figure 3**.

LENGTH defines the height or width of the axis.

The **LABEL**= option defines the options for just the label [text] of the axis. This text length cannot be more than 256 characters. The text for the label should be enclose with quotes. Example can be seen in the code snippet given below.

Details about the options related to AXIS are defined in **Step 8**.

Along with LABEL, appearance of the text can be enhanced with help of the following options:

COLOR=text-color

FONT=font | NONE

HEIGHT=text-height <units >

JUSTIFY=LEFT | CENTER | RIGHT

ROTATE=degrees

The **OFFSET**= defines how far from the lower left corner of the graph to start the first tick mark.

```
axis1 order=(0 to &ht by &htby) value=(f=swissb)origin=(10.8,20)pct
length=70 pct
label=(a=90 h=10pt f=arial "MaleSchoolPCT and FemaleSchoolPCT");
axis2 order=(1 to 20 by 1)  origin=(10.8,21.1)pct value=(angle=75)
offset=(-5,-5) length=80 pct
label=("country name");
axis3 order=(1 to 50 by 1) value=none origin=(10.8,26.9)pct minor=none
offset=(-5,-5) length=80 pct
label=none;
```

Step 13: Defining the color for the Bars in Graph

To categorize different parameters of the source dataset, which are portrayed in the form of different bars, color is a very important option.

SAS graphs allow applying colors through different ways.

A list of colors can be created with the GOPTIONS with option as COLORS. Name of the color can be defined directly or color code can be provided.

GOPTIONS COLORS=(red green blue);

GOPTIONS COLORS=(CXFF0000 CX00FF00 CX0000FF);

GOPTIONS COLORS=(medium_red medium_green medium_blue);

There are rules to define the color:

If it is the name of the color, then it should not exceed 64 characters
If it is color code, that should not exceed eight characters and a valid color code should be defined.
If the color list is not defined SAS uses the default colors list in the device driver.

To reset the color already specified with <<COLORS= graphics option>>, the existing color can be nullified and SAS will reset it back to the default color list.

GOPTIONS COLORS=();

Color can be specified in global statements which enhance the output are as follows:

1. AXIS
2. FOOTNOTE
3. LEGEND
4. PATTERN
5. SYMBOL
6. TITLE

Below code snippet is a part of Step 13. It is using GOPTION with option COLOR. This code is adding two different colors for two different bars as shown in **Figure 9.**

```
      goptions colors=( CX99293D,CX090766);
```

Step 14: Creating the Bar Graph

In this example an overlaying Bar chart is created [Bar and line will be plotted together]. SAS helps to create different types of Bar graphs, which helps to produce meaningful data. In the example given below, bar chart is created using PROC GCHART. Data is pulled from the dataset called "GR_C_NAME2" and gout is sending output to the work directory which can be later redirected to a desired location after merging with line graph, which is also sent to the work directory.

Once the dataset is defined with PROC GCHART, some other functionality can be added to enhance the output.

Next section of this paper creates vertical Bar Charts with PROC GCHART and the VBAR statement

With PROC GCHART, some options can be used like VBAR. VBAR is used to specify the column name. The column, whose value will be represented with the bar(s).

Bar graph is displayed with any aggregate function, like sum, mean, average, etc. as an option for the TYPE of VBAR.

Options, which can be used with VBAR are as follows:

Table 4. Describing the options used with VBAR.

Option	**Description**
SUMVAR =	Set the height of the bar graphs. In this example SUMVAR is having an option as result, which is the vale for the variable "maleschoolpct".
PATTERNID =	With option MIDPOINT is used to provide each bar a different color
SPACE =	Sets space between groups
GSPACE =	Sets space between bars
DISCRETE =	It is used to include the actual values of the variable to label each bar
ANNOTATE=	Annotate data set "ANNO" defined in **step 10** is called here as an option with option annotate. This will define the legends including all the featured applied to the dataset "ANNO"

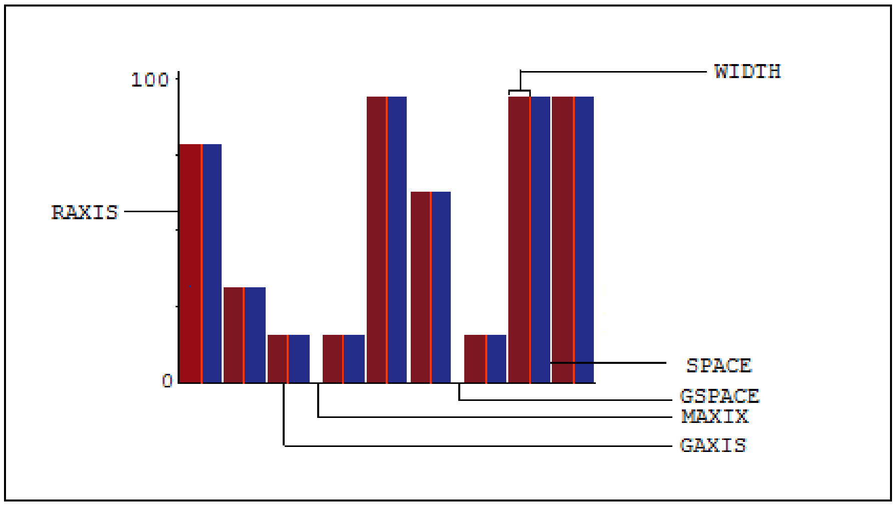

Figure 8. Diagram with a pictorial view defining RAXIS, MAXIS, GAXIS, SPACE, GSPACE and WIDTH

```
proc gchart data=gr_c_name2 gout=work.gseg34;
format result percent6.0;
vbar gr / group=c_name
type =sum
sumvar=result outside=sum
patternid=midpoint
space=2.3 gspace=2.3 width=2
raxis=axis1
space=2.3
maxis=axis3
space=2.3
gaxis=axis2
discrete
patternid=midpoint
annotate=anno; /*identifies the annotate dataset*/
run;
quit;
```

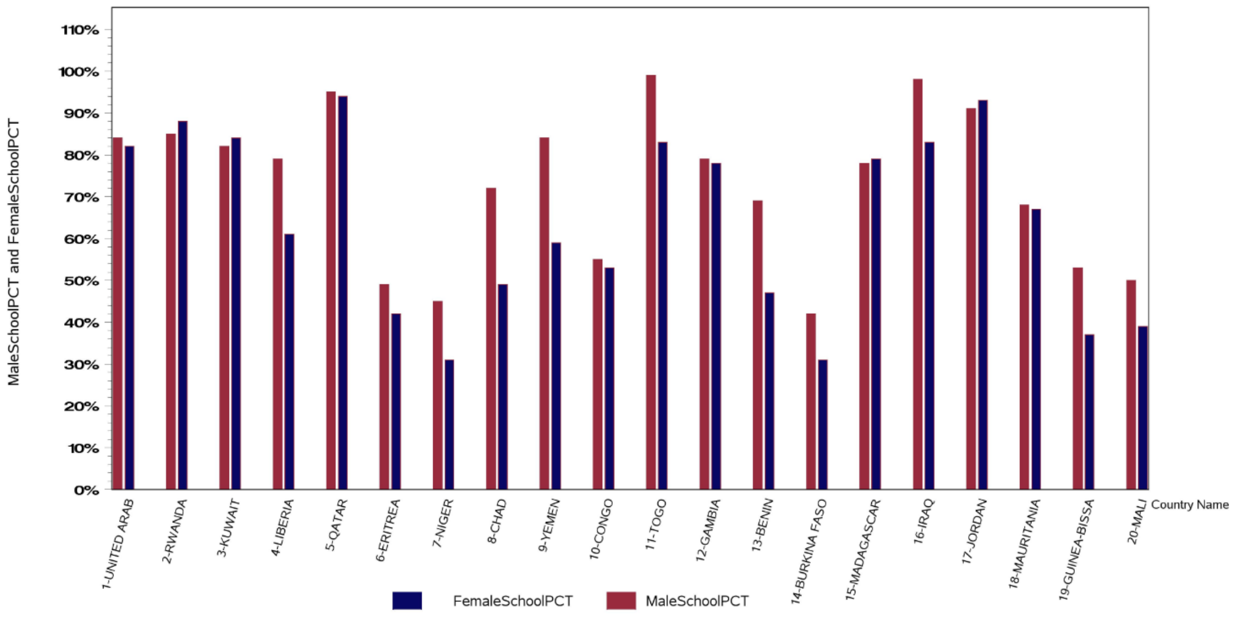

Figure 9. Sample of the Bar Graph created by the code defined above

Step 15: Saving the Graph as JPEG file

Below code shows how to save the graph as JPEG file in the desired location. File name and path should be provided. To save the graph through SAS code, FILENAME can be used along with GOPTION statement. Filename is used to define the name of the JPEG(.JIF can also be created) file along with the file path. GOPTION with GSFNAME can call the variable defined for FILENAME.

Defining the path and the file name, where Output graph - .JPEG file should be stored.

XPIXELS=, **XMAX**=, **YPIXELS**=, and **YMAX**= graphics options to set the resolution of the graphics. XPIXELS=, XMAX=, defines Horizontal dimension and YPIXELS=, and YMAX= defines vertical dimension.

LFACTOR=, defines the thickness of the line. It can range from 0 – 9999. LFACTOR=0 corresponds to 1.

```
filename figures "c:/Graph_Folder/SAMPLE_GRAPH_BY_C_NAME.jpg";
goptions reset=all device=jpeg gsfname=figures htext=8pt
xpixels=4600 ypixels=2400 lfactor=2
xmax=6.4in ymax=3.4in hsize=6.4in vsize=3.4in;
```

Step 16: Plotting Bar and Line graph together – with multiple Y Axes

Merging both the graphs together – Line is overlaying

Oftentimes we need to create multiple graphs on single plot. We can create several graphs to describe different variables from multiple tables. There are two ways that you can accomplish this in SAS using **PROC GREPLAY**.

- Using SAS code
- Using point-and-click interface

In the code snippet given below, the **IGOUT** with option for filename on the **PROC GREPLAY** statement is pointing towards the SAS code for the graphs defined above.

TC=Option **explains the SAS template layout**

TEMPLATE=Option indicates that template that to be used.

TREPLAY statement can indicate the cell where each graph layout template should appear.

In the below example all are in the same position: “1”, because of overlaying graphs. Otherwise, if they need to appear all together one after or side-by-side, they can be followed by numbers “1:”, “2:”, “3:”, “4:”.

```
proc greplay igout=work.gseg34 nofs tc=sashelp.templt template=whole;
treplay
1:gchart
1:gplot
;
run;
quit;
proc datasets library = work nolist;
     delete gsegs / mtype = data;
run;
```

CONCLUSION

SAS graph is the best way to represent an output in pictorial format. Multi axis graph defined in this paper can be used with different scenarios to elaborate the output data.

Apart from this, PROC GPLOT, GCHART elaborated in this paper could be very useful tips to SAS users.

REFERENCES

Eberhert, M.G., 2006. Presentation quality graphs with SAS/GRAPH®, Proceedings of NESUG 2006, Philadelphia, PA.

SAS Institute Inc. 2010. Specifying Colors in SAS/GRAPH Programs, *SAS/GRAPH® 9.2: Reference, Second Edition*. Available at http://support.sas.com/documentation/cdl/en/graphref/63022/HTML/default/viewer.htm#colors-specify-color.htm. © 2014, SAS Institute Inc., Cary, NC, USA. All Rights Reserved. Reproduced with permission of SAS Institute Inc., Cary, NC.

CONTACT INFORMATION

The author can be contacted at soma.ghosh@optum.com and ghosh.soma@gmail.com

TRADEMARK CITATION