PharmaSUG 2016 - Paper TT03

Controlling Colors by Name; Selecting, Ordering, and Using Colors for Your Viewing Pleasure

Arthur L. Carpenter
California Occidental Consultants, Anchorage, AK

ABSTRACT

Within SAS® literally millions of colors are available for use in our charts, graphs, and reports. We can name these colors using techniques which include color wheels, RGB (Red, Green, Blue) HEX codes, and HLS (Hue, Lightness, Saturation) HEX codes. But sometimes I just want to use a color by name. When I want purple, I want to be able to ask for purple not CX703070 or H03C5066. But am I limiting myself to just one purple? What about light purple or pinkish purple. Do those colors have names or must I use the codes? It turns out that they do have names. Names that we can use. Names that we can select, names that we can order, names that we can use to build our graphs and reports.

This paper will show you how to gather color names and manipulate them so that you can take advantage of your favorite purple; be it 'purple', 'grayish purple', 'vivid purple', or 'pale purplish blue'. Much of the control will be obtained through the use of user defined formats. Learn how to build these formats based on a data set containing a list of these colors.

KEYWORDS

Color-naming Schemes, ARRAY, SAS Registry, PROC REGISTRY, PROC FSLIST, user defined formats, PROC FORMAT

INTRODUCTION

If you want to know virtually anything about the colors used by SAS, you will want to explore Technical Support document TS-688. This document carefully describes the valid color-naming schemes that are available in SAS, and gives examples of each type. The naming schemes include:

- RGB (red green blue)
- CMYK (cyan magenta yellow black)
- HLS (hue lightness saturation)
- HSV (hue saturation brightness), also called HSB
- Gray scale
- SAS color names (from the SAS Registry)
- SAS Color Naming System (CNS)

Table 1 Examples of specifying colors taken from TS-688

Color-Naming Scheme	Example
RGB	COLORS=(cx98FB98 cxDDA0DD cxFFDAB9 cxDB7093 cxB0E0E6)
CMYK	COLORS=("FF00FF00" "00FFFF00" "FFFFFF00")
HLS	COLORS=(H14055FF H0F060FF H0B485FF H07880FF)
HSV	COLORS=(V0F055FF v010FFFF v03BFFFF v12C55E8)
Gray Scale	COLORS=(GRAY4F GRAY6D GRAY8A GRAYC3)

SAS Registry Colors	COLORS=(palegreen plum peachpuff palevioletred powderblue)
CNS Color Names	COLORS=("very light purplish blue" "light vivid green" "medium strong yellow" "dark grayish green")

This paper describes how to obtain and use just one of these color list types – the SAS Registry Colors. This color naming scheme selects and names a hundred and fifty or so of the millions of available colors. These are the colors (like BLUE and RED) that you can use by name in your color lists in SAS/GRAPH and ODS.

The basic steps of this process will include:

1. Retrieve the list of color names from the SAS Registry
2. Store the list of names in a text file
3. Manipulate the list by changing the order of the colors and by removing unwanted colors
4. Convert the manipulated color list into a SAS data set
5. Make use of the color list

RETRIEVING AND STORING THE COLOR LIST

The color list is stored in the SAS Registry and you can extract this list from the registry by using PROC REGISTRY.

❶ name a text file that will receive the list of colors
❷ unless you have created a color list in SASUSER this option is probably not necessary
❸ names the registry key of interest

```
proc registry export= "colornames.txt" ❶
              usesashelp ❷
              startat='COLORNAMES' ❸;
   run;
```

Not that you necessarily need to do so, but the resulting text file can be viewed in a number of ways. If you are running SAS interactively (such as through the Display Manager), one of the easier is through the use of the FSLIST procedure. In this usage the list of colors will be written to the FSLIST window where it can be viewed. The text file shown here contains not only the color name, but the hex color representation as well. Since we are only interested in the color name in this application a DATA step will be used to eliminate all but the color names.

```
proc fslist file= "colornames.txt" ;
   run ;
```

```
#--- File updated at 08JUN2015 08:45:08
#--- Exporting SASHELP registry starting
at key [COLORNAMES]

[COLORNAMES]
"Active"="HTML"

[COLORNAMES\HTML]
"AliceBlue"= hex: F0,F8,FF
"AntiqueWhite"= hex: FA,EB,D7
"Aqua"= hex: 00,FF,FF
"Aquamarine"= hex: 7F,FD,D4
"Azure"= hex: F0,FF,FF
"Beige"= hex: F5,F5,DC
"Bisque"= hex: FF,E4,C4
```

We can separate the list of colors by reading in and parsing each text line. If we notice that each color has a corresponding hex code with the hex code and color name separated by an equal sign, we can eliminate the lines without colors.

```
data colors;
   infile 'colornames.txt' ❹
          dlm='=' dsd truncover;
   length colorname $35.;
   input colorname $ otherstuff $;
   if otherstuff=:'hex'; ❺
   put colorname; ❻
   run;
```

❹ The DLM and DSD INFILE options are used to control how the file is read into the DATA step.
❺ Only the data lines with a hex color code are used.
❻ Optionally the list of colors can also written to the LOG.

A PROC PRINT of the first few colors in this data set is shown here.

Obs	colorname
1	AliceBlue
2	AntiqueWhite
3	Aqua
4	Aquamarine
5	Azure
6	Beige
7	Bisque

Although the list of available colors can now be used by name, further manipulation of the list is needed to make the use of the list practical.

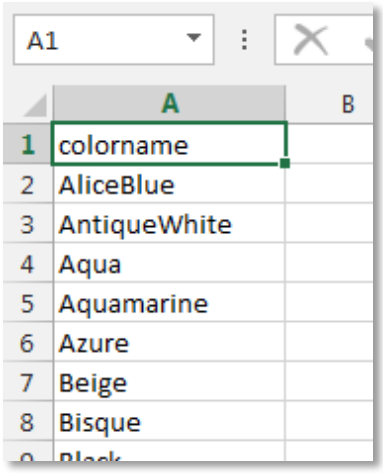
MANIPULATING THE COLOR LIST

In the application for which this color list was developed, the user needed to be able to see and manipulate the color list manually. They needed the ability include or reorder the color list so that their application would take advantage of only certain colors and in a prescribed order. The manipulation of the color list was to be a onetime manual operation, however it turned out that continual adjustments were required, and the process for color selection and ordering had to be made as easy as possible. We experimented with several approaches. I am going to show you alternate approaches here, so that you can choose which will work best for you.

Using a CSV File

A CSV text file can be edited using either Excel® or a text editor. In this approach the list of colors in WORK.COLORS is exported to a CSV file using a simple PROC EXPORT step. In this PROC EXPORT example the CSV file is located in the \TEMP directory.

```
proc export data= work.colors
            outfile= "c:\temp\colors.csv"
            dbms=csv replace;
     putnames=yes;
run;
```



Once the CSV file has been created it can be opened and manipulated using Excel (shown here) or through the use of a text editor such as NOTEPAD. Initially the colors are sorted alphabetically, however since we also need to be able to control the availability of a color as well as the order, individual colors can be deleted or the order of the colors can be changed.

Using the EXCELXP Tagset to View the Colors

Color selection and visualization is easier if the user can actually see the colors in use. Since the final application is to be displayed in Excel, the EXCLXP tagset can be used to both export the colors to Excel as well as to visualize the colors as they will appear in the final table.

```
ods tagsets.excelxp file='c:\temp\colors.xls';
* Show colors in use;
proc report data=work.colors nowd;
  column  colorname colorname=clr; ❶
  define colorname/ 'Color Name';
  define clr  / 'Color';
  compute clr /char length=35; ❷
 ❸ call   define(_col_,'style','style={background='||colorname||'}');
  endcomp;
run;
ods tagsets.excelxp close;
```

PROC REPORT is used to display both the name of the color, and the color itself through the use of the CALL DEFINE.

❶ An alias for COLORNAME is established with the name CLR.
❷ The style attributes (in this case the background color) will be established in a compute block.
❸ The CALL DEFINE routine is used to establish the background color for this cell of CLR by using the current value of the color name.

	A	B
1	Color Name	Color
2	AliceBlue	AliceBlue
3	AntiqueWhite	AntiqueWhite
4	Bisque	Bisque
5	BlanchedAlmond	BlanchedAlmond
6	Beige	Beige
7	Aqua	Aqua
8	Aquamarine	Aquamarine
9	Azure	Azure
10	Black	
11	Blue	Blue
12	BlueViolet	BlueViolet
13	BR	BR
14	Brown	Brown
15	Burlywood	Burlywood

After even some minor manipulation of the color order, it is apparent that some colors are too similar to each other to be useful (at least for this application). You can also see that BLACK is not a good color choice as a background color when the foreground color is also BLACK.

Because the human eye cannot distinguish many more than one or two dozen colors, and since the list starts with 150 colors, there are plenty of colors to choose from to create a pallett with sufficient variation and contrast.

Editing the List Directly in the Enhanced Editor

Because the list originally exists as a text file, the list itself can be copied directly into a DATA step for editing. Ultimately we are going to need the modified list in a SAS data set anyway, so we can save a step by doing the editing of the color list within the Program Editor.

```
data colors;
infile cards truncover;
input colorname $35.;
datalines;
RoyalBlue
AliceBlue
Yellow
Green
Aquamarine
Bisque
BlueViolet
IndianRed
run;
```

For demonstration purposes, only a few of the colors in the list are shown in this DATA step. Eventually when the list is used, we will want to be able to maintain the new color ordering supplied by the user edits. A PROC REPORT step that is essentially the same as the one shown above can be used to show the selected colors. The color list selection shown in the DATA step to the right is the one used in the remaining examples.

CONVERTING THE COLOR LIST INTO A SAS DATA SET

This step is not needed if the list is manipulated directly within the DATA step. However if you choose to use Excel to manipulate the list of colors, you will need to convert either the CSV file or the Excel file into a SAS data set. A straight forward PROC IMPORT can be used in both instances.

```
proc import out = CSVcolors
            file= "c:\temp\colors.csv"
            dbms=csv replace;
     getnames=yes;
run;
```

```
proc import out = XLScolors
            file= "c:\temp\colors.xls"
            dbms=excel replace;
     getnames=yes;
run;
```

USING THE COLOR LIST

Now that the color list is in a SAS data set, we can use this list in a number of ways. Typically I find it easiest to use this list to build a format and then to apply the format using traffic lighting techniques. Formats are built from a data set by creating a specialized data set that can be used as a control file by PROC FORMAT. This control file can have up to a couple of dozen specific variables (PROC FORMAT expects specific variable names), however there are only three that are required and the examples that follow only use a few of these specialized variables.

Basing the Colors on Ranks

The colors were saved in a specific order and in this example we are going to use that order to associate colors with the rank of a level of a classification variable.

```
data controlcolor(keep=start label fmtname hlo); ❶
      set colors(rename=(colorname=label)) end=eof; ❷
      retain fmtname 'RnkColor'; ❸
      start = _n_; ❹
      output controlcolor;
      if eof then do; ❺
         hlo='o';
         start=' ';
         label='white';
         output controlcolor;
      end;
      run;
proc format cntlin=controlcolor; ❻
   run;
```

❶ The four variables needed by PROC FORMAT are kept in the data set.
❷ The color name is assigned to the LABEL variable. This is the value that we will map into.
❸ The format name is specified in the variable FMTNAME. Since this name is constant, a RETAIN statement is used.
❹ The data value that is to be mapped from is stored in the START variable. This will be the rank value in this example.
❺ Values that are not otherwise covered (there should not be any) are mapped to the color white, which is often the default for a number of ODS styles.
❻ The control data set is passed to PROC FORMAT using the CNTLIN= option. PROC FORMAT then builds the format defined in the data set (RNKCOLOR.).

In the example that follows I would like to rank REGION by total sales. The complication is that I want the color in the report to be based on the rank of the region, but I want the regions themselves to be specified in alphabetical order. The SUMMARY and RANK procedures are used to summarize the data.

```
proc summary data=sashelp.shoes nway;
   class region;
   var sales;
   output out=totalsales sum=total;
   run;
proc rank data=totalsales
          out  = rnktot;
   var total;
   ranks ranktotal;
   run;
```

The data set to be reported (RNKTOT) has a column containing the value of the rank (RANKTOTAL), which we will use to tie to the RNKCOLOR. format that was created in the previous PROC FORMAT step.

PROC REPORT will again be used with the CALL DEFINE routine to build the style attribute for the background color, however this time a format is used to select the background color.

```
title 'Ranked Total Sales';
proc report data=rnktot;
   column region total ranktotal;
   define region / order;
   define total / format=dollar10.;
   define ranktotal / display;
   compute ranktotal;
      call define(_row_,
                  'style',
                  'style={background='||put(ranktotal,rnkcolor40.)||'}'); ❼
   endcomp;
   run;
```

❼ The PUT function is used to translate the rank value in RANKTOTAL to a color by using the RNKCOLOR. format.

The resulting table will have known and preselected colors for each of the ranks in the table, regardless of the order of the regions.

Ranked Total Sales

Region	Total Sales	Rank for Variable total
Africa	$2,342,588	3
Asia	$460,231	1
Canada	$4,255,712	7
Central America/Caribbean	$3,657,753	6
Eastern Europe	$2,394,940	4
Middle East	$5,631,779	10
Pacific	$2,296,794	2
South America	$2,434,783	5
United States	$5,503,986	9
Western Europe	$4,873,000	8

Associating the Colors with Levels of a Classification Variable

You may want to associate specific colors with specific values of a classification variable. Using the same data as in the previous example I now want the same color to be associated with the same region throughout all of my reports. Again a format will be built, however this time the colors will be tied to specific levels of a classification variable. This allows us to always have the same color associated with say 'South America' regardless of what other regions are in the report.

```
proc sql noprint;
create table unique as
   select distinct region
      from sashelp.shoes;
quit;
```

For this example we want to tie a specific color to a specific value of REGION. The first step is to build a list of the unique values of REGION, and this can be easily done in a simple PROC SQL step. This list of unique values will then be paired with the list of unique colors. By default the list generated by SQL will be sorted alphabetically, so the first region will be associated with the first color in our color list.

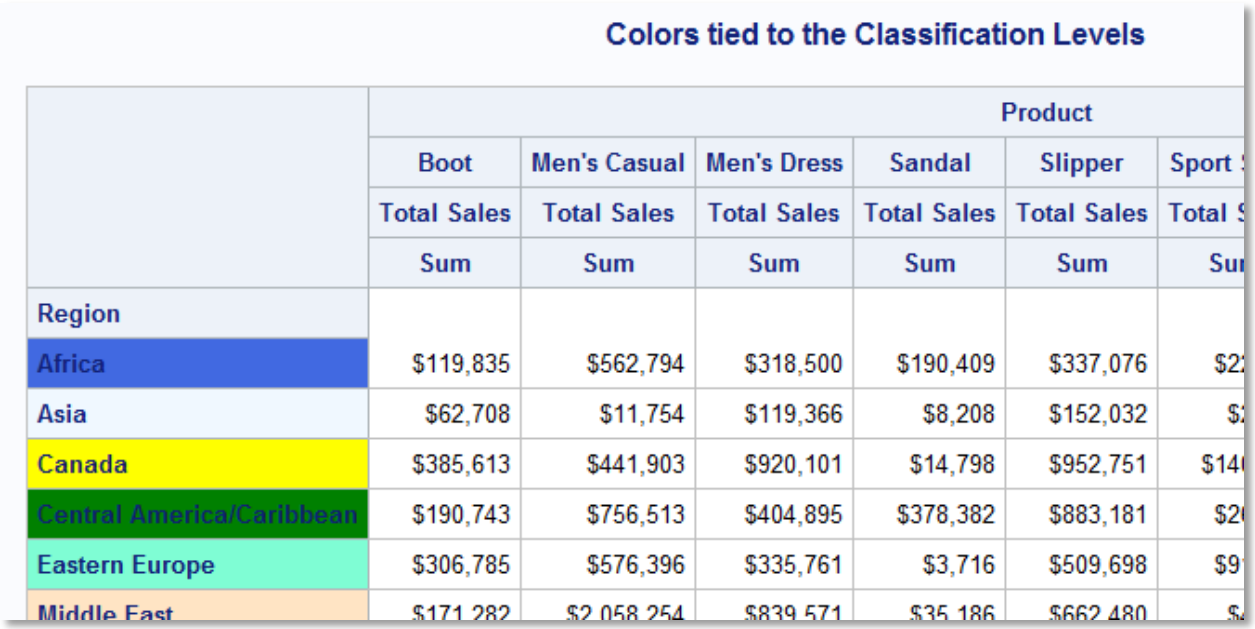
We again build a format using a control data set, however this time we merge the unique region values with the unique colors. The merge itself is performed with two SET statements. This allows us to detect the last value of REGION (we are assuming that we have more colors than regions).

```
data controlcolor(keep=start label fmtname hlo);
   set unique(rename=(region=start)) end=eof;
   set colors(rename=(colorname=label));
   retain fmtname '$RegColor';
   output controlcolor;
   if eof then do;
      hlo='o';
      start=' ';
      label='white';
      output controlcolor;
      stop;
   end;
   run;
proc format cntlin=controlcolor;
   run;
```

This time we want to summarize using PROC TABULATE so the background color attribute is assigned using the STYLE= option in the CLASSLEV statement. Notice that the syntax for the STYLE= option on the CLASSLEV statement is essentially the same as was used in the CALL DEFINE routine in the PROC REPORT step above. Other attributes such as links, font, font size, and foreground color can also be controlled by the STYLE option, and each of these can also be tied to a format.

```
title 'Colors tied to the Classification Levels';
proc tabulate data=sashelp.shoes;
   class product region;
   var sales;
   table region,product*sales*f=dollar10.;
   classlev region /style={background=$regcolor.};
   run;
```

A portion of resulting report shows that the region values have received the correct colors. Want different colors? All you have to do is reorder the colors in the color list and rebuild the format.



Colors tied to the Classification Levels

	Product					
	Boot	Men's Casual	Men's Dress	Sandal	Slipper	Sport
	Total Sales	Total Sales	Total Sales	Total Sales	Total Sales	Total
	Sum	Sum	Sum	Sum	Sum	Su
Region						
Africa	$119,835	$562,794	$318,500	$190,409	$337,076	$2
Asia	$62,708	$11,754	$119,366	$8,208	$152,032	$
Canada	$385,613	$441,903	$920,101	$14,798	$952,751	$14
Central America/Caribbean	$190,743	$756,513	$404,895	$378,382	$883,181	$2
Eastern Europe	$306,785	$576,396	$335,761	$3,716	$509,698	$9
Middle East	$171,282	$2,058,254	$839,571	$35,186	$662,480	$

Associating the Colors with Values in the Table

When the colors are to be tied to specific ranges of values the process is similar, however the DATA step that creates the format control data set is slightly more complicated.

The breakdown of the sales ranges can be specified in a data set with variables that note the lower and upper bounds of the ranges.

START	lower bound of the range
END	upper bound of the range
EEXCL	make the upper bound of the range exclusive
HLO	Used with a range value contains a keyword HIGH, LOW, OTHER

Notice that even though we are building a numeric format, the range variables (START and END) are character. The type of these two variables to not matter to PROC FORMAT and in this case, since the END variable contains a HIGH, it is necessary.

```
data ranges;
   infile datalines truncover;
   input start $ end $ eexcl $ hlo $ ;
   datalines;
      0  100000 Y
 100000  250000 Y
 250000  500000 Y
 500000 1000000 Y
1000000 2000000 Y
2000000    high N  H
run;
```

Once the data set containing the ranges has been built, it can be merged with the colors data to create the control file.

```
data controlcolor(keep=start end label eexcl fmtname hlo type);
   set ranges end=eof;
   set colors(rename=(colorname=label));
      retain fmtname 'SalesColor' type 'N';
      output controlcolor;
      if eof then do;
         hlo='o';
         start=' ';
         end=' ';
         eexcl='N';
         label='white';
         output controlcolor;
         stop;
      end;
      run;
proc format cntlin=controlcolor;
   run;
```

Because exclusive ranges are being created, more variables have been added to the control file. The variable TYPE is not really needed in this example, as it is designating the SALESCOLOR. format to be numeric. This variable allows you to create character formats without specifying the $ in the format name during the building process.

In our TABULATE example we want to use traffic lighting techniques to highlight sales ranges using colors. You can see that the second level ($100,000 to <$250,000) matches to the color named 'AliceBlue'. For the default (SASWEB) style used here, this may not be a distinctive enough color. Fortunately if we want to adjust the colors used, all we need to do is go back and edit our color list and recreate the formats.

Colors tied to Total Sales

	Product				
	Boot	Men's Casual	Men's Dress	Sandal	Slippe
	Total Sales	Total Sales	Total Sales	Total Sales	Total Sa
	Sum	Sum	Sum	Sum	Sum
Region					
Africa	$119,835	$562,794	$318,500	$190,409	$337
Asia	$62,708	$11,754	$119,366	$8,208	$152
Canada	$385,613	$441,903	$920,101	$14,798	$952

In the previous example the STYLE= option was used on the CLASSLEV statement so that the background color could be associated with a classification variable. Here we want the color to be associated with the derived values (total sales). To do this we nest (using the *) the STYLE option under SALES. Notice that the option not only uses interior curly braces, but that the whole option is also surrounded by curly braces [square braces could also have been used].

```
title 'Colors tied to Total Sales';
proc tabulate data=sashelp.shoes;
   class product region;
   var sales;
   table region,
         product*sales
                *{style={background=salescolor35.}}
                *f=dollar10.;
   run;
```

SUMMARY

It is possible to retrieve the list of standard color names that SAS stores in the SAS Registry, and then to use these names to build user defined lists. These lists can then be used to create formats that can in turn be used to standardize your reports and tables. Now when you need a color, you can call it by name.

POSTSCRIPT ON COLOR NAMES

Although the identification of a color by name seems to be easy and straightforward, especially for the standard names that we use on a daily basis (red, blue, green), selecting colors by name can have some severe repercussions. Primarily the problem is that there are millions of colors and the names that we use are limited. The problem is exacerbated by some of the color adjectives like: 'light', and 'pinkish'. These names do not take into consideration such things as hue and saturation and can result in issues such as 'pink' appearing to be darker than 'light pink'.

If you want precise control over colors; if you want to select colors that will be reliably rendered; you will most likely need to select your colors by using one of the color coding systems other than by selecting them by name. LeRoy Bessler (2012, 2013) has done a great deal of work with the selection and presentation of colors. If you want definitive discussions on the merits and selection of colors, his papers and presentations should be your starting point.

Regardless of whether you use color names or one of the other ways of making your color list, the techniques described in this paper can still be applied.

ABOUT THE AUTHOR

Art Carpenter's publications list includes; five books, two chapters in *Reporting from the Field*, and numerous papers and posters presented at SAS Global Forum, SUGI, PharmaSUG, WUSS, and other regional conferences. Art has been using SAS since 1977 and has served in various leadership positions in local, regional, and national user groups.

Art is a SAS Certified Advanced Professional Programmer, and through California Occidental Consultants he teaches SAS courses and provides contract SAS programming support nationwide.

Recent publications are listed on my sasCommunity.org Presentation Index page. SAS programs associated with this paper can be found at:
http://sascommunity.org/wiki/Presentations:ArtCarpenter_Papers_and_Presentations

AUTHOR CONTACT

Arthur L. Carpenter
California Occidental Consultants
10606 Ketch Circle
Anchorage, AK 99515

(907) 865-9167
art@caloxy.com
www.caloxy.com

ACKNOWLEDGEMENTS

When I needed to figure out how to find and extract the colors that SAS uses by name, Peter Crawford at Crawford Software Consultancy Ltd, knew that they were in the SAS Registry, and he showed me how to use PROC REGISTRY to find and access the list of colors.

REFERENCES

Papers

More on creating user defined formats can be found in:

Carpenter, Arthur L., 2003, "Building and Using User Defined Formats", Proceedings of the 11th Annual Western Users of SAS Software, Inc. Users Group Conference, Cary, NC: SAS Institute Inc. Also in the proceedings of the Twenty-ninth SAS User Group International Conference (SUGI), 2004, Cary, NC: SAS Institute Inc., also in the proceedings of the Mid West SAS User Group Conference (MWSUG), 2005, Cary, NC: SAS Institute Inc. Also presented at the 16th Annual Western Users of SAS Software, Inc. Users Group Conference (WUSS), Universal City, CA in 2008.

Use of the techniques in this paper are further described in:

Carpenter, Arthur L., 2015, "Color, Rank, Count, Name; Controlling it all in PROC REPORT", presented at the 2015 MWSUG conference. http://www.mwsug.org/proceedings/2015/BB/MWSUG-2015-BB-06.pdf. Also Presented at the 2016 SAS Global Forum Conference.

Identifying colors other than by name and other uses of color in reports and graphs:

Bessler, LeRoy, 2013 "Using Color to Communicate, Not to Decorate", http://www.wiilsu.org/EQzxioeazdFYT/SUSNov2013/Proceedings/Papers/Bessler%20-%20Using%20Color%20to%20Communicate,%20Not%20to%20Decorate.pdf

Bessler, LeRoy, 2012 "Data Visualization Tips, Techniques, and Tools: Bessler's Best for SAS Graphs, Web, and Color", http://www.wiilsu.org/igfsldkjfngrsel9883/SUSNov2012/Proceedings/Papers/Bessler%20-%20Data%20Visualization%20Tips,%20Techniques,%20and%20Tools%20-%20Bessler's%20Best%20for%20SAS%20Graphs,%20Web,%20and%20Color.pdf

Support Documents

Technical Support document TS-688

https://support.sas.com/techsup/technote/ts688/ts688.html#predefined

Color naming schemes

http://support.sas.com/documentation/cdl/en/graphref/63022/HTML/default/viewer.htm#colors-specify-color.htm

TRADEMARK INFORMATION