Paper TT04

SAS/GRAPH ® Colors Made Easy

Max Cherny, GlaxoSmithKline, Collegeville, PA

ABSTRACT

Creating visually appealing graphs is one of the most challenging tasks for SAS/GRAPH users. The judicious use of colors can make SAS graphs easier to interpret. However, using colors in SAS graphs is a challenge since SAS employs a number of different color schemes. They include HLS, HSV, RGB, Gray-Scale and CMY(K). Each color scheme uses different complex algorithms to construct a given color. Colors are represented as hard-to-understand hexadecimal values. A SAS user needs to understand how to use these color schemes in order to choose the most appropriate colors in a graph.

This paper describes these color schemes in basic terms and demonstrates how to efficiently apply any color and its variation to a given graph without having to learn the underlying algorithms used in SAS color schemes. Easy and reusable examples to create different colors are provided. These examples can serve either as the stand-alone programs or as the starting point for further experimentation with SAS colors.

SAS also provides CNS and SAS registry color schemes which allow SAS users to use plain English words for various shades of colors. These underutilized schemes and their color palettes are explained and examples are provided.

INTRODUCTION

Creating visually appealing graphs is one of the most challenging tasks for SAS users. Tactful use of color facilitates the interpretation of graphs by the reader. It also allows the writer to emphasize certain points or differentiate various aspects of the data in the graph.

SAS/GRAPH is a very powerful application for creating many kinds of colorful graphs. However with this power comes a certain degree of complexity. The programming of colors in SAS/GRAPH is not an easy task and requires a good knowledge of various color schemes that exist in SAS. The real or perceived complexity impedes some users from trying to create visually appealing SAS graphs.

There are many different color schemes available to SAS users. The main color schemes include: RGB (red green blue), HLS (hue lightness saturation), CMY (cyan magenta yellow), HSV (hue saturation value) or HSB (hue saturation brightness), Gray-Scale, SAS Registry and CNS (color naming scheme). RGB and HLS are frequently used color schemes; however they are not easy to use because the colors are represented in the form of hexadecimal numbers. Colors can be constructed by varying values of hue, lightness or saturation or by varying percentages of other colors. For example, most colors may be constructed by changing the percentages of red, blue and green colors. These percentages are in turn converted into hexadecimal formats. This process is quite complex and not intuitive. It is not easy for a SAS user to figure out how various percentages will construct a desired color.

Instead of using numerical percentages of the color components, CNS and SAS registry schemes use plain English names for various colors and their variations.

THE HLS (HUE LIGHTNESS SATURATION) COLOR SCHEME:

HLS scheme is based on the Tektronix Color Standard. Any color in the HLS scheme can be constructed by specifying values for hue, lightness and saturation. The Tektronix Color Standard is presented in the Figure 1. The hue values are seen in the middle circle on the diagram and range from 0 to 360. For example, the hue value for blue is 50 and the hue value for yellow is 180.

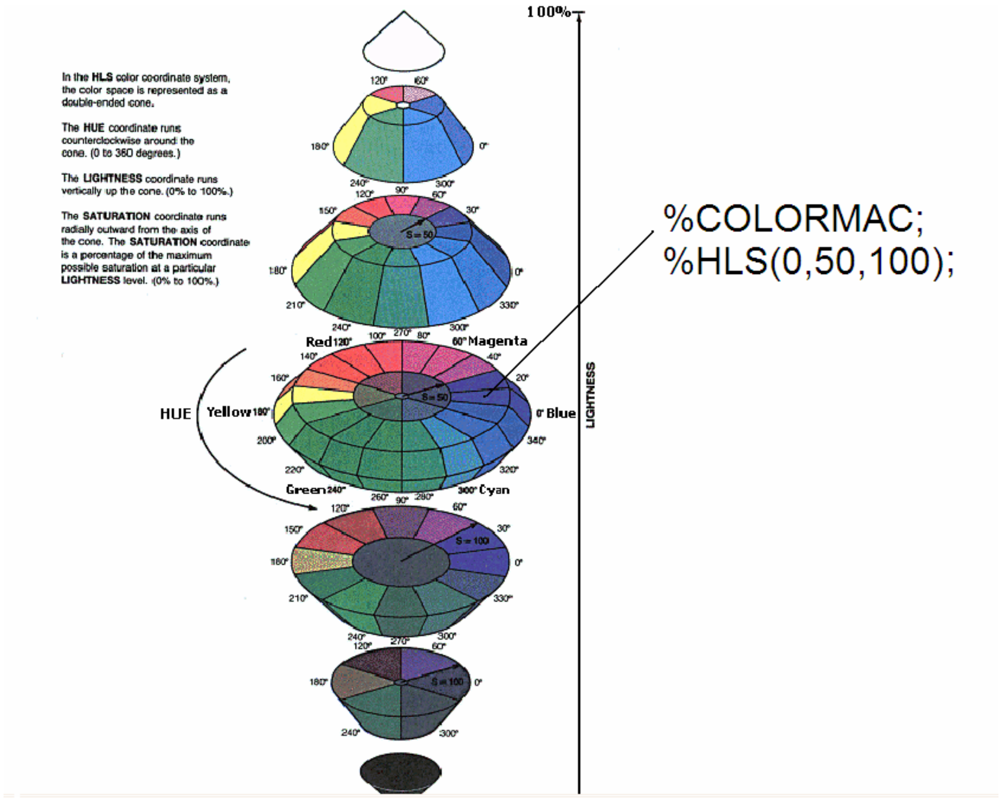

In the HLS color coordinate system, the color space is represented as a double-ended cone.
The HUE coordinate runs counterclockwise around the cone. (0 to 360 degrees.)
The LIGHTNESS coordinate runs vertically up the cone. (0% to 100%.)
The SATURATION coordinate runs radially outward from the axis of the cone. The SATURATION coordinate is a percentage of the maximum possible saturation at a particular LIGHTNESS level. (0% to 100%.)
100%
LIGHTNESS
HUE
Red 120°
60° Magenta
Yellow 180°
0° Blue
Green 240°
300° Cyan
S = 50
S = 100
%COLORMAC;
%HLS(0,50,100);

Figure 1. The Tektronix Color Standard (SAS Help and Documentation)

Lightness and saturation are based on percentages and range from 0 to 100. As seen in the figure above, 0 lightness forms the darkest color and 100 forms the brightest.

These values are actually represented in hexadecimal format from 00 to FF. For example, in order to use blue color in HLS scheme a SAS user will need to specify the color as H00080FF. Thus the following statement will make the color of pattern1 blue:

```
       PATTERN1 color= H00080FF ;
```

In the hexadecimal format, H00080FF represents 0 value for hue, value of 50 for lightness and value of 100 for saturation. Obviously the use of hexadecimal format is not intuitive and indeed rather complex for an average SAS user. The %COLORMAC macro utility was created specifically to help SAS users work with such a scheme. In order to use these macros, the `%COLORMAC` statement must be first used. The `%HLS` statement converts values of hue, lightness and saturation into the corresponding HLS code and circumvents the need for hexadecimal representations of colors. The following statement will convert the combination of hue, lightness and saturation components to H07880FF which is the hexadecimal code for red color:

```
        %colormac;
       PATTERN1 color= %hls(120,50,100);
```

The first number is used to indicate hue component. The table below indicates the hue numbers of some basic colors:

Blue	50
Magenta	60
Red	120
Brown	160
Yellow	180
Green	240
Cyan	300

The second component in the %HLS statement is the lightness defined in percentages. 50% is the medium, 100% is the lightest and 0% is the darkest. The third component is the saturation percentage. Its value is usually set at 100%.

Now let's create a simple program using HLS codes. The program will create a chart of several colors and their values. It can be used to identify some colors in a HLS macro call (i.e. the code for medium red is 120-50-100).

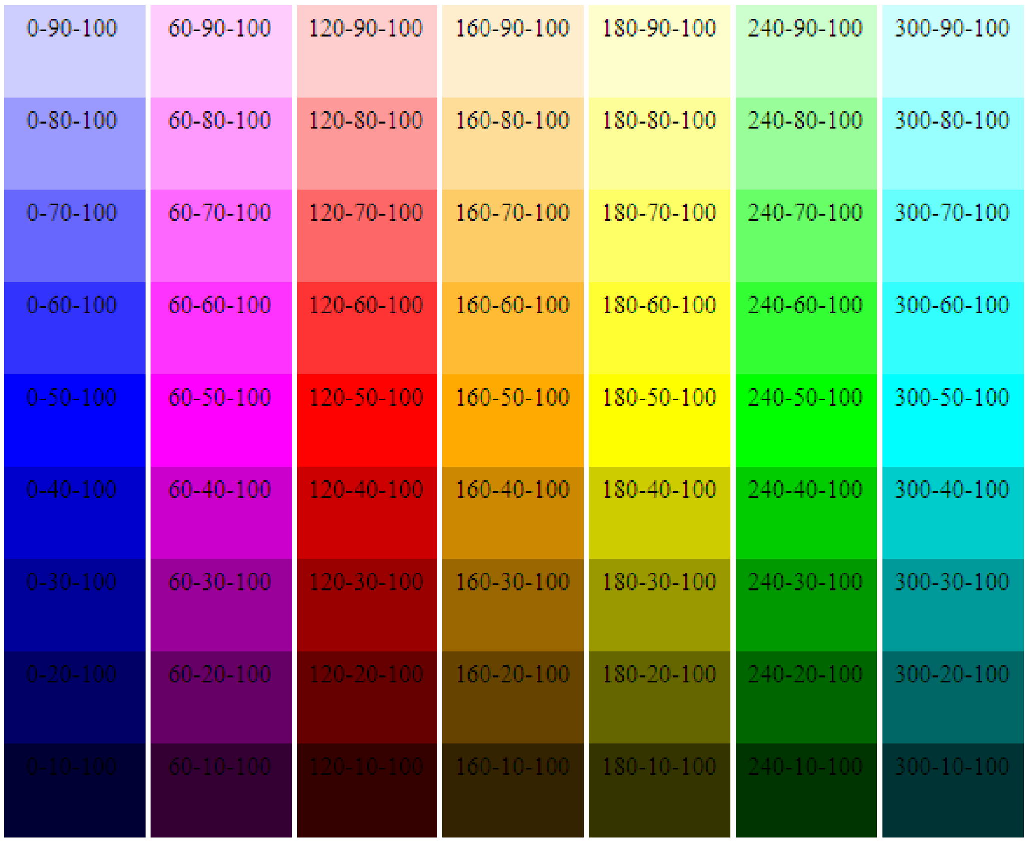


Figure 2. HLS Examples

The code below demonstrates how each color is created by converting values of hue, lightness and saturation into HLS codes by employing the %HLS macro:

```
       goptions reset=all ftitle="Times New Roman";
       %macro color_demo (color_scheme=HLS);
         /* create format for hues */
         proc format;
           invalue hls
           1=0
           2=60
           3=100
           4=180
           5=240
           6=280
           7=320;
         run;

         options mprint;

         /* invoke COLORMAC macros */
         %COLORMAC;

         /* Create dataset to display colors in proc gchart */
         data colors;
           constant=1;
           do main_color=1 to 7 ;
             do i=1 to 8;
               output;
             end;
           output;
           end;
         run;

         /* Create numbers for color components */
         data colors (drop=i);
           set colors (keep=main_color i);
```

```
      constant=1;
      saturation=100;
      lightness=i*10;
      hue=input(main_color, &color_scheme..);
    run;

    /* Store color components as macro variables */
    data colors;
      set colors;
      n=_n_;
      call symput ('colors', _N_ );
      call symput ('hue' || left (put (_n_, 8.)),
         compress(trim (hue)));
      call symput ('lightness' || left (put (_n_, 8.)),
         compress(trim (lightness)));
      call symput ('saturation' || left (put (_n_, 8.)),
         compress(trim (saturation)));
      call symput ('main_color_number' || left (put (_n_, 8.)),
         compress(trim (_N_)));
    run;

    /* Create pattern colors for vertical bar */
    %do i=1 %to &colors;
      %put &&hue&i &&lightness&i &&saturation&i;
      pattern&i color=%&color_scheme(&&hue&i,&&lightness&i,&&saturation&i);
    %end;

    /* Use data annotate to indicate component percentages in the color bars */
    data anno;
      retain color 'Black' when 'a'
      xsys ysys '2' ;
      set colors;
      position ='E' ;
      hsys='3';
      midpoint=main_color;
      subgroup=n;
      style="'Times New Roman'";
      size=3;
      text=cat(hue,"-",lightness,"-",saturation);
    run;

    title1 "&color_scheme color scheme";

    /* Display different colors as vertical bars  */
    proc gchart data=colors;
      vbar main_color/ annotate=anno type=sum sumvar=constant subgroup=n
                       width=10 space=0.5 nolegend noframe noaxis;
      run;
    quit;
  %mend color_demo;
  %color_demo(color_scheme=HLS);
```

This program uses a loop to create a dataset where each main hue (i.e. red) is represented by the HUE variable. Lightness and saturation are other variables in the dataset. Each component is converted into a macro variable which is later used to create pattern statements for each color in a loop. Each bar represents a color grouped by its lightness. The annotate data set is employed to create text indicating values for each component of a given color.

This program above and Figure 2 may be helpful to SAS users trying to create various colors. The author advises experimenting with various values of color components. Note that each bar represents an entire spectrum of a hue going from the brightest to darkest. The author only applied 9 shades of each hue, however the same program can be easily modified to derive finer variations of different hues based on the Tektronix Color Standard. For example, modifying the program by changing the number of shades of each color from 9 to 100 will display a very gradual change in color.

THE RGB (RED GREEN BLUE) COLOR SCHEME

The RGB scheme is another major color scheme. It is based on the following concept: any given color can be defined as a combination of red, green and blue colors. For example, 100% of red and 0% of green and blue colors creates the red color; whereas, 100% of red, 100% of green and 0% of blue creates the yellow color. Note that the RGB color scheme used in Microsoft Office is different from the SAS RGB scheme. It applies a slightly different scale. The

Figure 3 displays the color percentages of red, blue and green used in the SAS RGB color scheme and may be used to construct any given color.

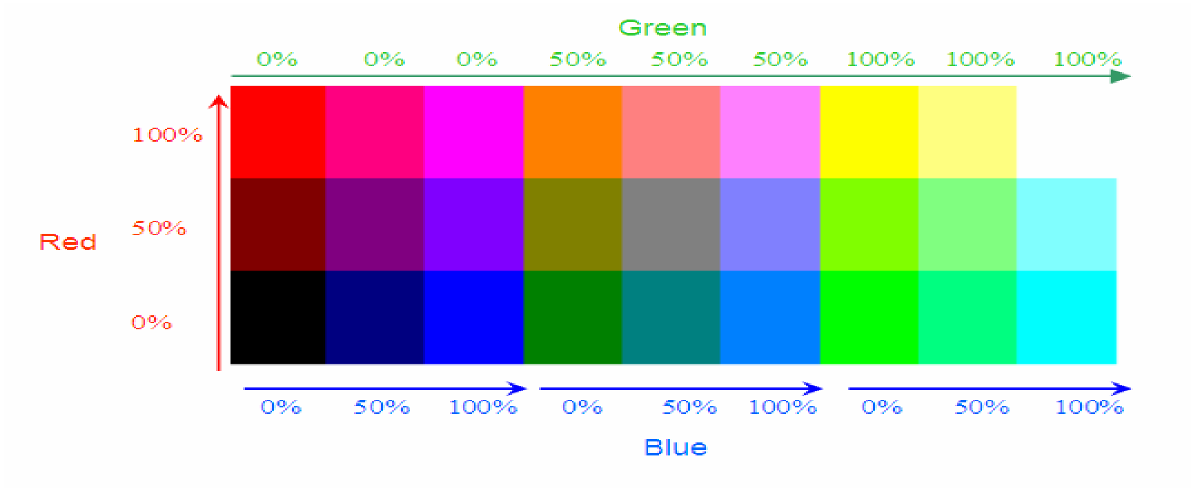


Figure 3. RGB color scheme

For example, we can see from this chart that the RGB combination for basic dark magenta color is 100% of red, 0% of green and 100% of blue. Just like HLS, RGB uses a hexadecimal representation of color combinations. To avoid working with hexadecimal numbers, the %RGB macro should be used. This macro converts percentages to a hexadecimal number. The following statement converts the (100, 0, 0) percentage combination to the RGB hexadecimal value of CXFF0000 which is the red:

```
        PATTERN1 color= %RGB (100,0,0);
```

Now let's create a graph using the RGB color scheme.

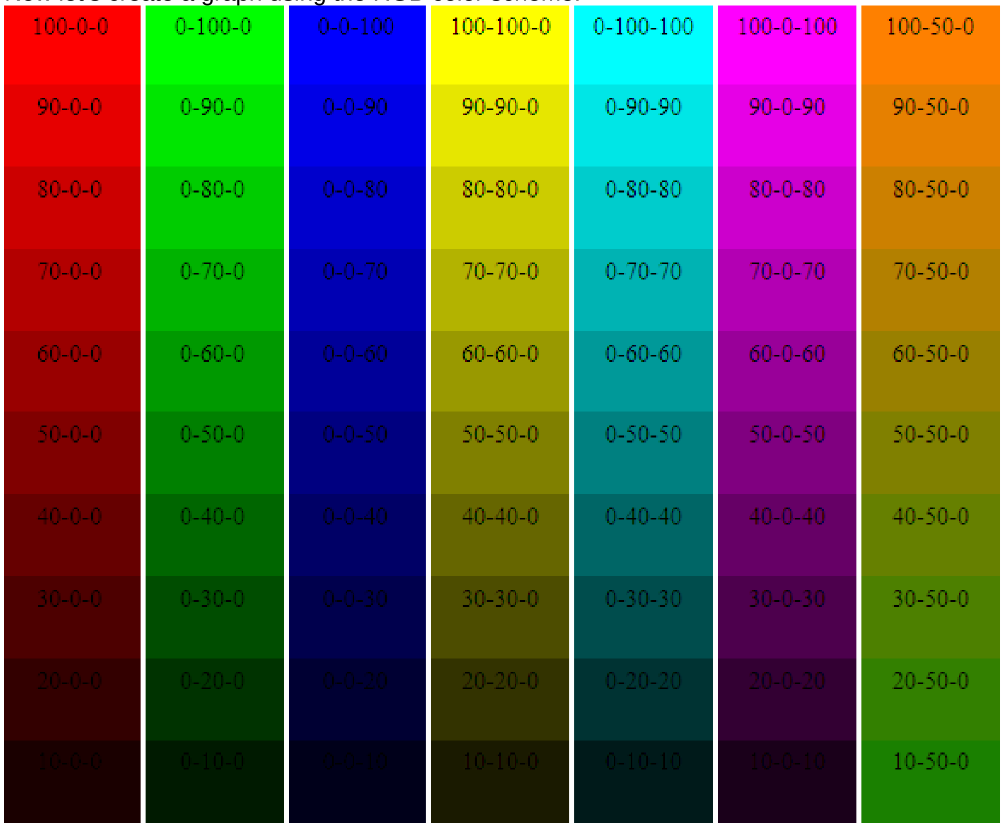


Figure 4. RGB Examples

Each bar in Figure 4 represents shades of a given color using RGB scheme. Each color bar is grouped by its shade. It goes from medium shade to its darkest shade for the first 5 color bars. The author used the %RGB macro to convert the RGB color percentages to RGB hexadecimal values. The code for the Figure 4 is shown below:

```
        goptions reset=all ftitle="Times New Roman";
        %macro rgbdemo(colorscheme=, black=);
```

```
options mprint;

data rgbcolors;  delete;  run;

/* create numbers representing percentages of red, blue and green */
%macro makergb (r=, g=, b=, r0=0, g0=0, b0=0,maincolor=);
  data newcolor;
    retain r &r0 g &g0 b &b0;
    do i=1 to 10;
      r=r+&r;
      g=g+&g;
      b=b+&b;
      main_color=&maincolor;
      constant=1;
      output;
    end;
  run;

  data rgbcolors;
    set rgbcolors newcolor;
  run;
%mend makergb ;

%makergb (maincolor=1, r=+10, g=0, b=0);
%makergb (maincolor=2, r=0, g=+10, b=0);
%makergb (maincolor=3, r=0, g=0, b=+10);
%makergb (maincolor=4, r=+10, g=+10, b=0);
%makergb (maincolor=5, r=0, g=+10, b=+10);
%makergb (maincolor=6, r=+10, g=0, b=+10);
%makergb (maincolor=7, r=+10, g=0, g0=50, b=0);

/* Store color components as macro variables */
data rgbcolors;
  set rgbcolors;
  n=_n_;
  call symput ('colors', _N_ );
  call symput ('r' || left (put (_n_, 8.)), compress(trim (r)));
  call symput ('g' || left (put (_n_, 8.)), compress(trim (g)));
  call symput ('b' || left (put (_n_, 8.)), compress(trim (b)));
run;

/* Use data annotate to indicate component percentages in the color bars */
data anno;
  retain color 'Black' when 'a'
  xsys ysys '2' ;
  set rgbcolors;
  size=3;    position ='E' ;    hsys='3';    midpoint=main_color;
  subgroup=n;
  text=cat(r,"-",g,"-",b);
  %if &colorscheme eq CMYK %then %do;
    text=cat(r,"-",g,"-",b,"-",&black);
  %end;
  style="'Times New Roman'";
run;

%COLORMAC;

%do i=1 %to &colors;
  /* Create pattern colors for vertical bar */
  %put &i} &&r&i &&g&i &&b&i;
  %if &colorscheme ne CMYK %then %do;
    pattern&i color=%&colorscheme(&&r&i,&&g&i,&&b&i);
  %end;

  %if &colorscheme eq CMYK %then %do;
    pattern&i color=%&colorscheme(&&r&i,&&g&i,&&b&i, &black);
  %end;

%end;
title1 "&colorscheme color scheme";
/* Display different colors as vertical bars  */
```

```
    proc gchart data=rgbcolors;
      vbar main_color/ annotate=anno type=sum sumvar=constant subgroup=n width=10
                      space=0.5 nolegend noframe noaxis;
    run;
    quit;
  %mend rgbdemo;

  %rgbdemo(colorscheme=RGB);
  %rgbdemo(colorscheme=CMY);
  %rgbdemo(colorscheme=CMYK, black=50);
```

The program uses a loop to create a dataset where the percentage of red, green and blue colors is stored as a variable. Each percentage is then converted to a macro variable which later creates pattern statements for each color. Each color bar represents a color grouped by its shade. The annotate DATA step is employed to create text indicating RGB percentages inside of each color. The program also contains some code for CMY and CMYK color schemes and the resulting output will be demonstrated in the corresponding section. Like the previous example, this program and the resulting output may be helpful to SAS users trying to create colors of various shades with RGB scheme. We advise experimenting with various RGB percentages.

Note that the COLORMAC macro may also be used to convert HLS codes to RGB (%HLS2RGB macro) and vice versa (%RGB2HLS macro). For example `%RGB2HLS(CX9F5F8F)`will translate a color code from RGB to HLS.

THE HSV (HUE SATURATION VALUE) COLOR SCHEME

The HSV scheme uses a combination of hue, saturation, and brightness. The HSV macro may be applied to convert the component combinations to hexadecimal values. HSV is also sometimes called HSB (hue saturation brightness). The hue component ranges from 0 to 360 where 0 defines red. Both saturation and brightness components range from 0 to 100 if used with %HSV macro. A saturation value of 0 forms white. A brightness value of 0 forms black.

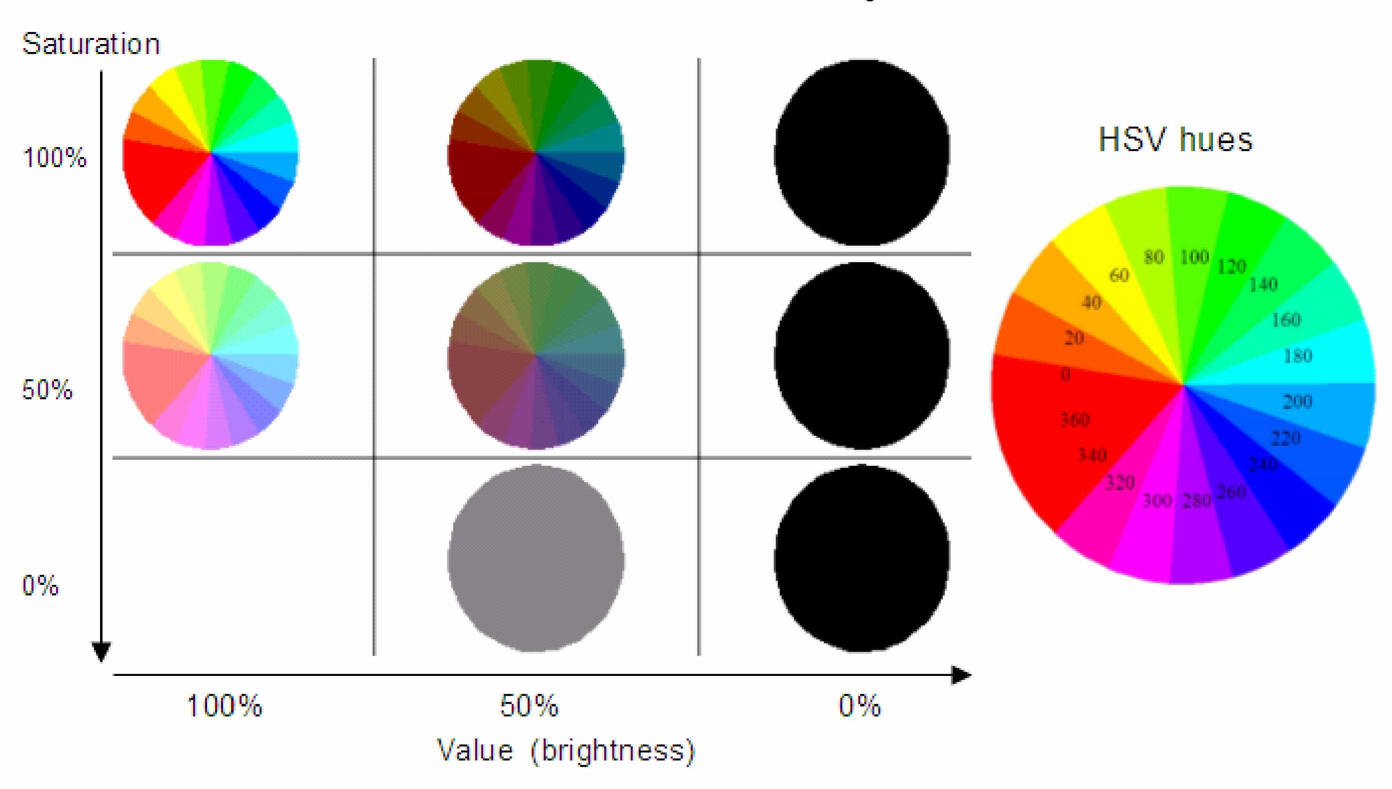


Figure 5. HSV color scheme (%HSV macro)

Figure 6 and Figure 7 demonstrate the range of hues in HSV with 100 and 60 values of brightness. The values of each component for a given color can be seen in the color bars.

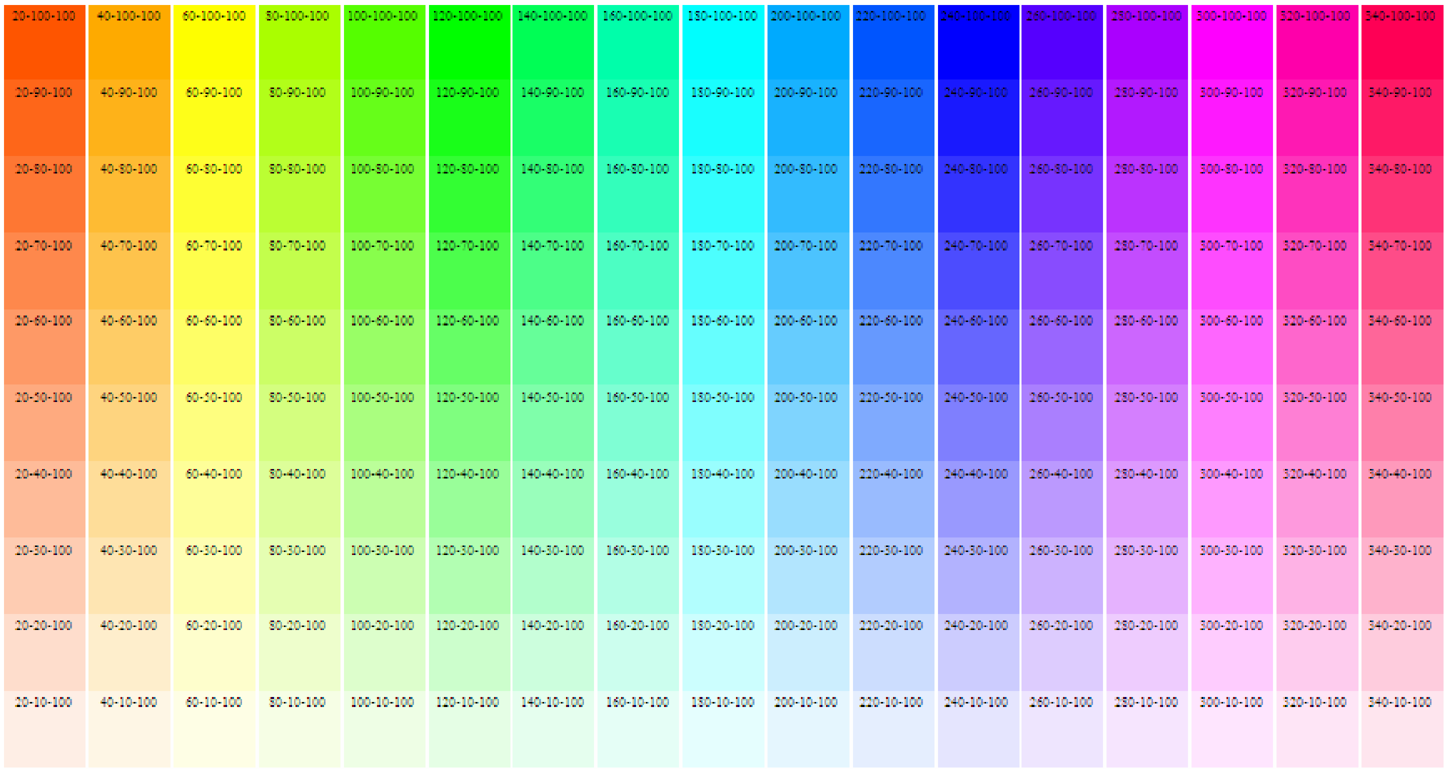


Figure 6. HVS Color Chart with Brightness Value of 100

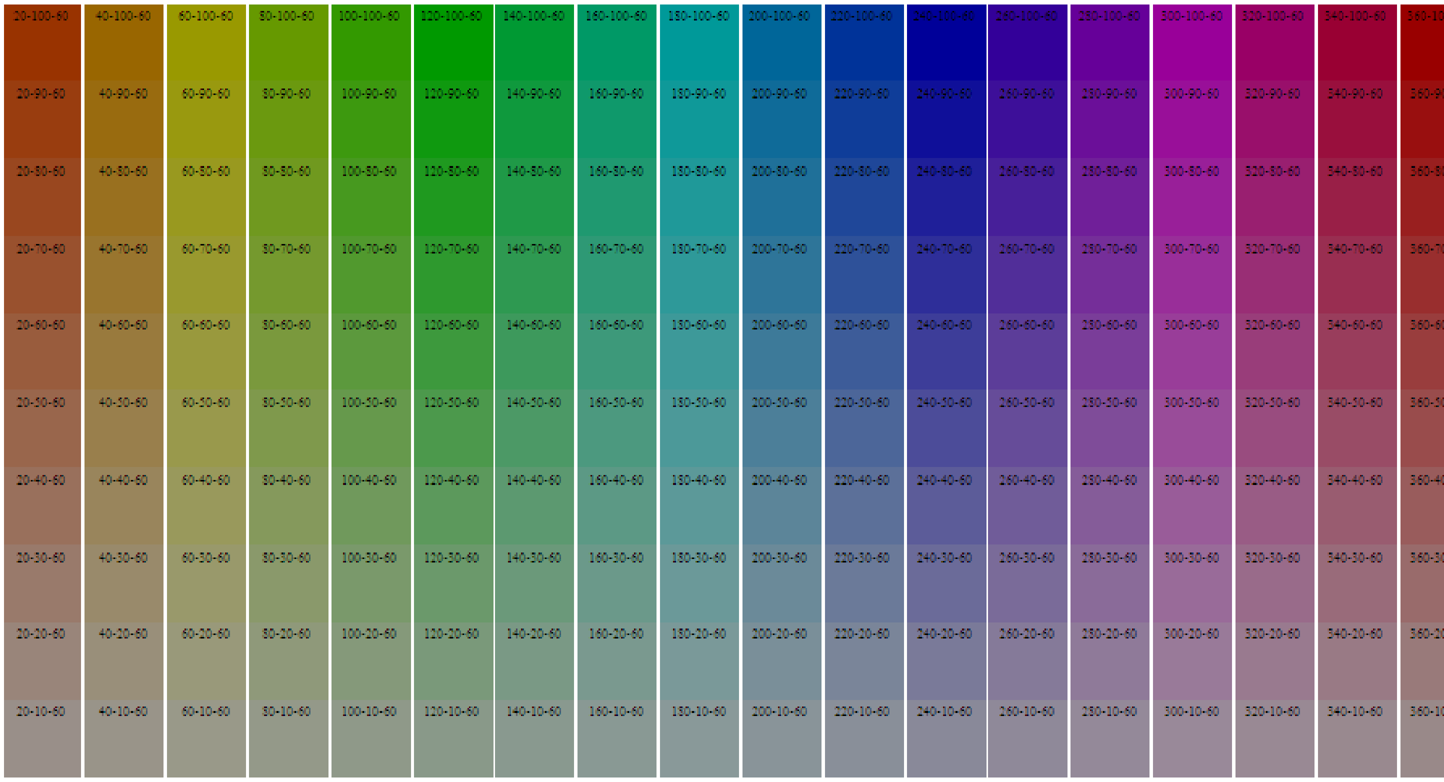


Figure 7. HVS Color Chart with Brightness Value of 60

The program which created these charts is shown here:

```
  %macro hsvdemo (brightness=100);
    option mprint;  goptions reset=all ftitle="Times New Roman";
    %colormac;
    /* create numbers representing color components */
    data hsvcolors;
      main_color=1;
      do c=1 to 18;
        hue=c*20;
        do saturation=10 to 90 by 10;
          output;
        end;
        output;
      end;
    run;
    /* Store color components as macro variables */
    data hsvcolors;
      set hsvcolors;
      n=_n_;
      call symput ('colors', _N_ );
      call symput ('hue' || left (put (_n_, 8.)), compress(trim (hue)));
      call symput ('saturation' || left (put (_n_, 8.)), compress(trim (saturation)));
    run;
    /* Create pattern colors for vertical bar */
    %do i=1 %to &colors;
      pattern&i color=%hsv(&&hue&i,&&saturation&i,&brightness);
    %end;
    /*Use data annotate to indicate component percentages in the color bars */
    data anno;
      retain color 'Black' when 'a'
      xsys ysys '2' ;
      set hsvcolors;
      position ='E' ;
      hsys='3';
      midpoint=c;
      subgroup=n;
      style="'Times New Roman'";
      size=1.8;
      text=cat(hue,"-",saturation,"-",&brightness);
    run;
    title "HSV color scheme with &brightness % brightness";
    /* Display different colors as vertical bars  */
    proc gchart data=hsvcolors;
      vbar c/ subgroup=n anno=anno discrete nolegend noaxis
      noframe width=6 nospace;
    run;
    quit;
  %mend hsvdemo;
  %hsvdemo (brightness=80);
  %hsvdemo (brightness=100);
```

THE CMY (CYAN MAGENTA YELLOW) AND CMYK (CYAN MAGENTA YELLOW BLACK) SCHEMES

The CMY color scheme is based on the concept that each color can be defined as combination of cyan, magenta, yellow, and black components.

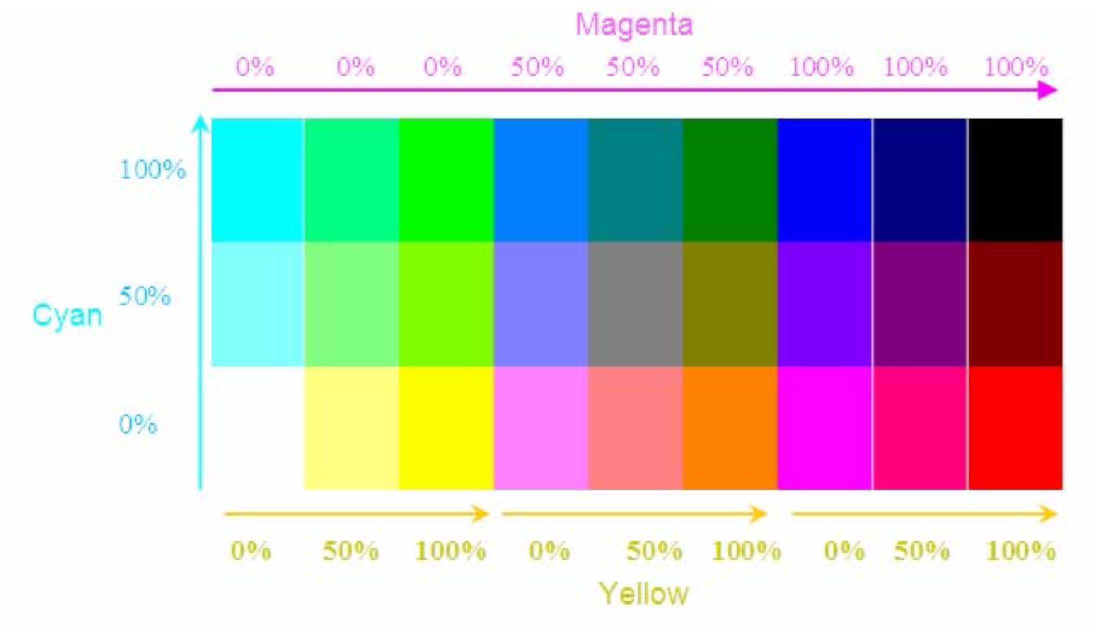
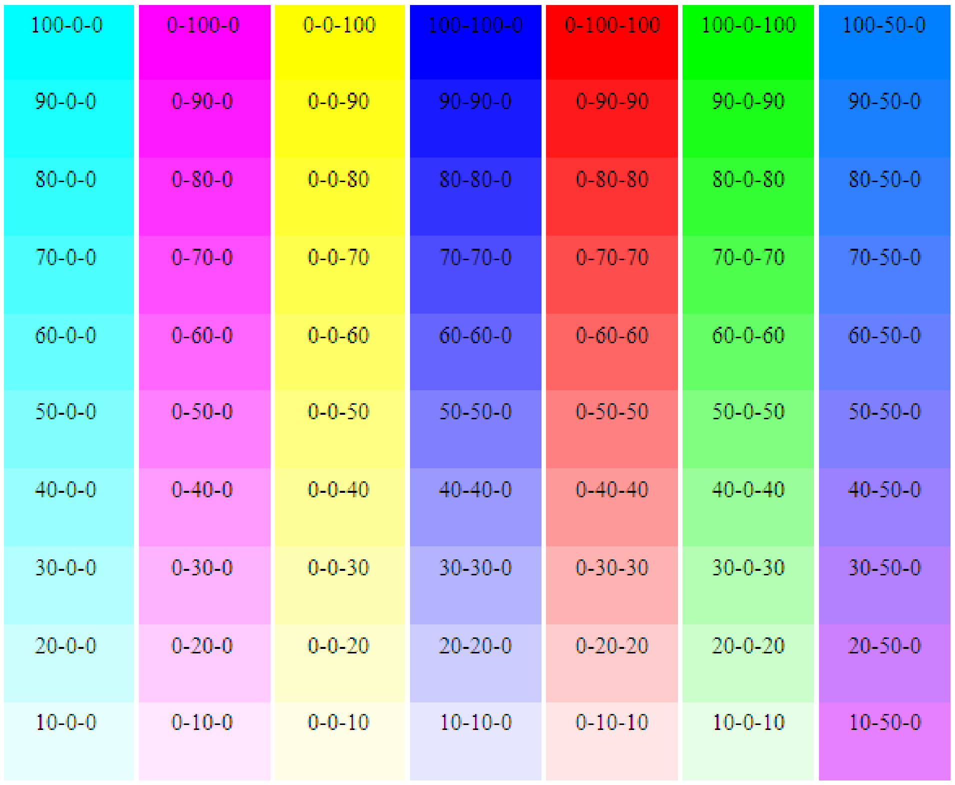


Figure 8. CMY color scheme

%CMY macro can be used to convert percentages of each component colors to its hexadecimal representation. To create a blue color in CMY use the following syntax: `%CMY(100,100,0);`



100-0-0	0-100-0	0-0-100	100-100-0	0-100-100	100-0-100	100-50-0
90-0-0	0-90-0	0-0-90	90-90-0	0-90-90	90-0-90	90-50-0
80-0-0	0-80-0	0-0-80	80-80-0	0-80-80	80-0-80	80-50-0
70-0-0	0-70-0	0-0-70	70-70-0	0-70-70	70-0-70	70-50-0
60-0-0	0-60-0	0-0-60	60-60-0	0-60-60	60-0-60	60-50-0
50-0-0	0-50-0	0-0-50	50-50-0	0-50-50	50-0-50	50-50-0
40-0-0	0-40-0	0-0-40	40-40-0	0-40-40	40-0-40	40-50-0
30-0-0	0-30-0	0-0-30	30-30-0	0-30-30	30-0-30	30-50-0
20-0-0	0-20-0	0-0-20	20-20-0	0-20-20	20-0-20	20-50-0
10-0-0	0-10-0	0-0-10	10-10-0	0-10-10	10-0-10	10-50-0

Figure 9. CMY examples

Figure 9 demonstrates some of the colors created by using CMY scheme. It was generated by running the same program which created the RGB chart and replacing %RGB macro call with %CMY macro call. The CMYK scheme is the same as CMY but it has black color as the fourth component. The higher the percentage of black color, the darker the color.

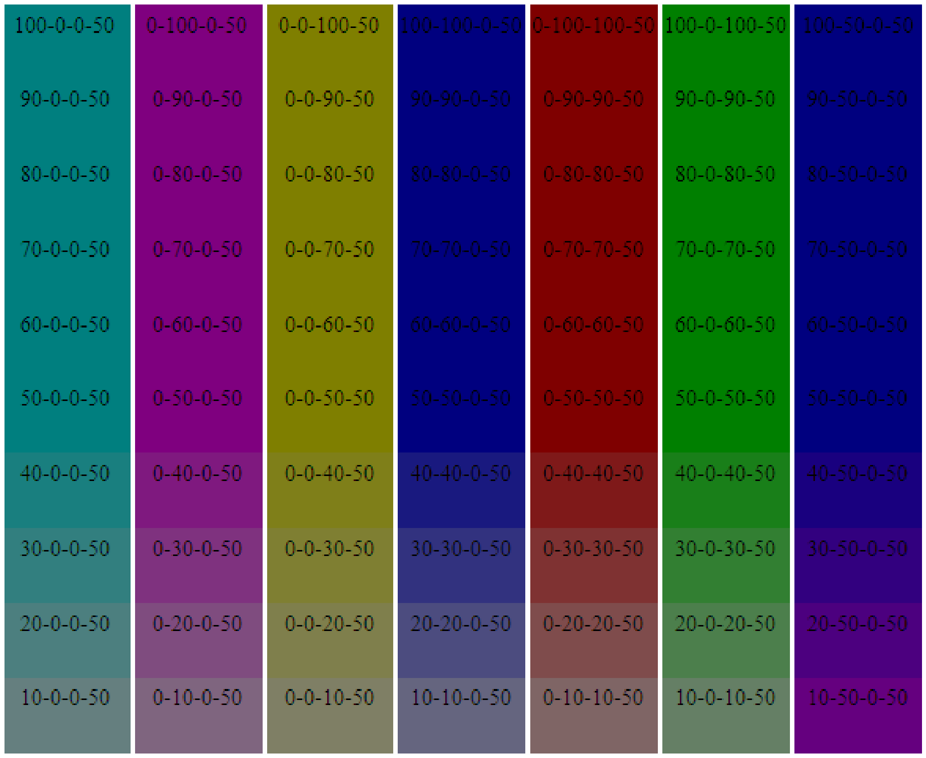


Figure 10. CMYK examples

Figure 10 is the same as Figure 9 but it now has the fourth component: 50% of black color.

THE GRAY-SCALE COLOR SCHEME

The Gray-scale color scheme may be used to create various shades of gray. Gray-scale shades are formed by combining the word GRAY with the hexadecimal value of a color from 0 to 255. The value of 0 is the darkest shade and value of 255 is the lightest. In order to convert a decimal number to a hexadecimal value use the `hex2.` format. For example, the statement `put(10, hex2.);` converts number 10 to its hexadecimal representation which is OA.

Figure 11 shows decimal values on the top row and their corresponding hexadecimal values of different shades of gray on the bottom row. In order to specify a shade of gray, the hexadecimal number must follow the string GRAY. For example, the following statement will form a dark gray:

```
        PATTERN1 color=GRAYOA;
```

Where as, the following statement will form a medium gray:

```
        PATTERN1 color=GRAYA0;
```

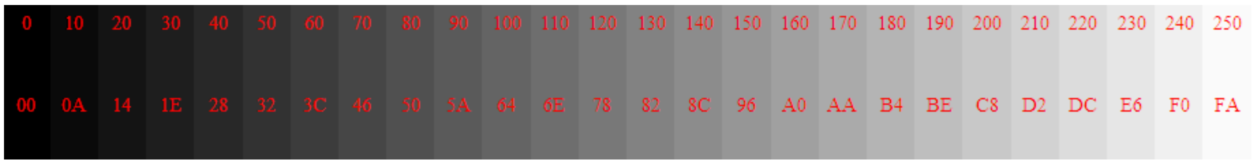


Figure 11. Gray-Scale Color Chart (scale is from 0 to 250)

Note that gray color can also be created by applying another color scheme such as HLS or RGB. Therefore, gray – scale color scheme is not the only option for working with gray color.

CNS (COLOR NAMING SCHEME)

The color schemes introduced above allow SAS users to create various colors and their shades. However these color-schemes are not intuitive and require at least some understanding of numeric color combinations used to create SAS colors. Many SAS users would find it difficult to determine the right combination for a given color.

Thus, the easiest way to create colors is to simply use plain English words for these colors. Figure 12 displays basic colors commonly used in graphs. These colors may be specified simply by using the English names for them. For example, to make pattern1 use green color the SAS code should be something like this:

```
        pattern1 color=green;
```

Some of these colors may be specified by using their abbreviations, for example BR for brown. The color abbreviations are shown in the parentheses.

Most SAS graphs do not require variations of the same color and can be created just by using these main colors. In fact, employing many different colors is probably not desired for most graphs as they may only confuse and overwhelm the intended audience.

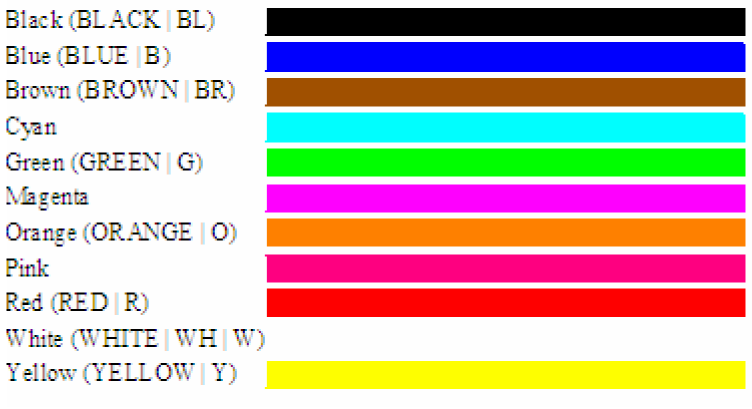


Figure 12. Main SAS Colors

Figure 13 displays some additional basic colors.

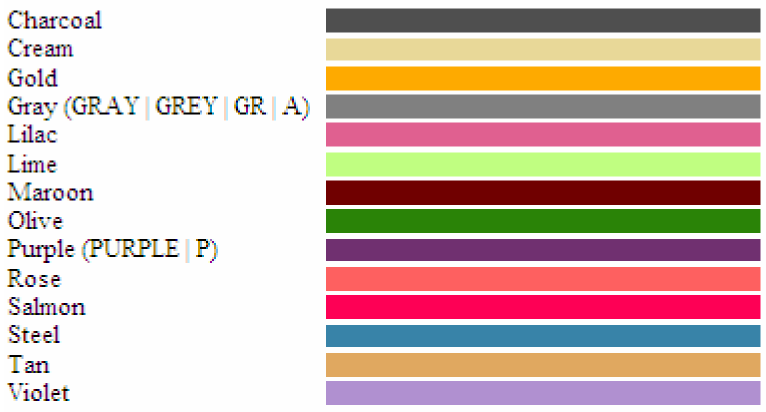


Figure 13. Additional Basic SAS Colors

These colors may be applied to a graph which needs a larger number of colors. The various shades of colors can also be specified by using words for lightness, saturation and hue. This is done as follows:

Lightness	Saturation	Hue
Black	Gray	Blue
Very	Dark	Grayish Purple
Dark	Moderate	Red
Medium	Strong	Orange/Brown
Light	Vivid	Yellow
Very Light	Green	

For example the following statement will create a lighter variation of red:

```
        PATTERN1 color= LightRed;
```

Figure 14 below shows some basic variations of red in plain English.

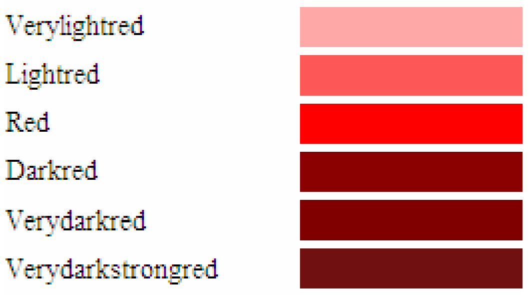


Figure 14. Red Colors

This semantic approach is much more intuitive then using component percentages or hexadecimal representations of colors.

The list of such names exists at http://www.uwm.edu/IMT/Computing/sasdoc8/sashtml/gref/zgscheme.htm. The list contains English names for each color along with their corresponding RGB and HLS codes. Note that this list does not contain all possible combinations of SAS color names. The author created a program which converts this list into a SAS dataset and then creates a list of lines representing colors. Figure 15 represents various versions of pink. Note that English names may work differently or not even work at all depending on hardware characteristics and device properties.

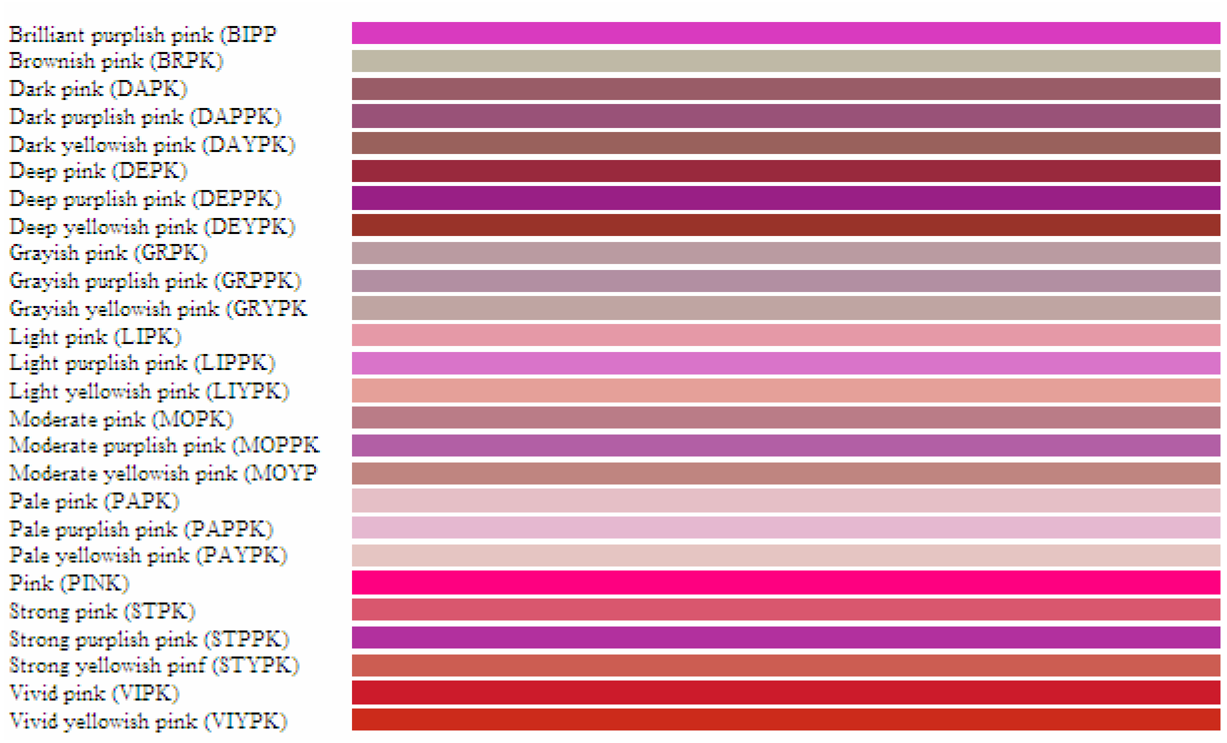


Figure 15. Variations of Pink Color in CNS Scheme

The CNS macro can be used to convert English names to HLS color codes as follows:

```
%COLORMAC;
data _null_;
put "%CNS(very dark red)";
run;
```

returns value of H07840FF.

SAS COLOR NAMES DEFINED IN SAS REGISTRY

Another way to use plain English is to use SAS color names already defined in the SAS registry. Some of the names are different from those of CNS. To display this list, the following statement must be executed:

```
proc registry list
 startat='COLORNAMES';
```

```
        run;
```
The list of colors will be produced. Figure 16 displays all the colors contained in SAS registry. SAS users can refer to this figure to choose colors:

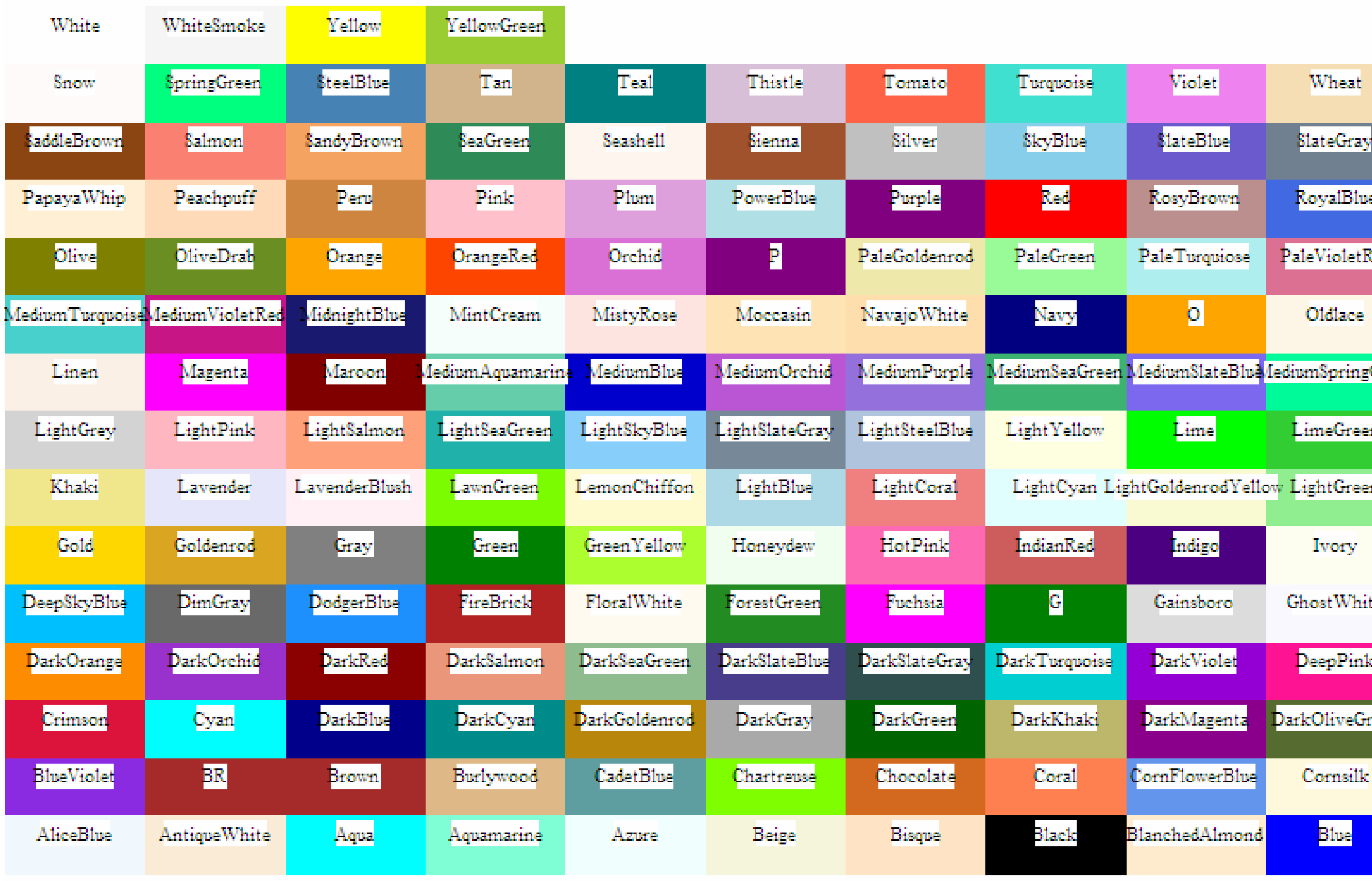


Figure 16. SAS Registry Colors

The chart above was constructed by converting the SAS registry color names into a dataset. This program may be provided upon request.

COMPARISON OF SAS COLOR SCHEMES

CNS is clearly the most intuitive and easy to understand SAS color scheme because it uses plain English to specify colors. It does not require any knowledge of how colors are constructed in various color schemes. The author advises using this approach whenever possible. Since this scheme also allows for creating various shades of SAS color, it should be sufficient for most types of graphs. SAS registry scheme is helpful because it uses English words for various colors as well. Additional experimentation with SAS color names may be needed.

The disadvantage of these color schemes is that they cannot be used to specify many specific variations of a given color. The obvious reason is that there are not enough English words for all of the colors. CNS can only help to specify no more then five or six shades of the same color.

To create more specific colors, SAS users must use other color schemes which allow for creating basically any variation of any color. The author suggests using HLS or RGB as they are commonly used color schemes. HLS is a rather intuitive color scheme. While it is not very easy to determine how to construct a certain color, it is certainly not too difficult to do it with some experimentation. The tables and programs used in this paper can be used as the starting point for such experimentation.

In the author's opinion, HLS is easier to understand then RGB. SAS users can refer to the Tektronix Color Standard chart to determine a color code of a major hue. Changing the percentage of lightness will yield various colors. HVS scheme is another intuitive scheme.

RGB is more difficult to understand because a SAS user needs to know how the percentages of red, blue and green form a desired color. The right percentages can be determined from the tables and programs used in this paper. RGB is a popular color scheme and is widely used in other applications such as Microsoft Word. Keep in mind that Microsoft Office RGB color scheme is different from that of SAS. CMY, HSV (or HSB), and Gray-Scale can also be used to construct colors. Like HLS and RGB, they all present the same challenge: SAS users must understand how colors are constructed numerically, either in decimal or hexadecimal notations.

PROPER USE OF COLORS

The use of colors in a graph must be carefully considered. Most graphs do not need to have many different colors. Using basic colors such as red, blue, yellow, green, black, brown, pink or magenta may be enough for most graphs. In fact, using too many colors may make graphs appear "busy" and possibly even confuse the audience. However some graphs may indeed require use of many different colors. For example let's consider Figure 17 which displays US states by population. The default SAS colors in the example are the basic SAS colors: red, green, blue, cyan and magenta.

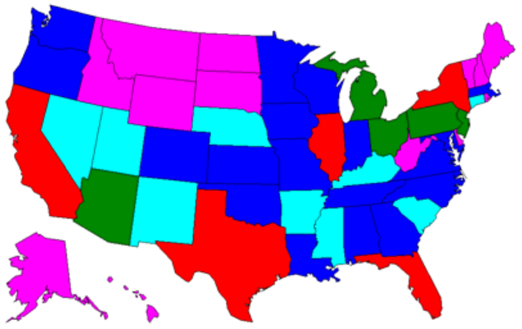


Figure 17. Example of SAS Default Color Scheme

Figure 17 is a reasonably good graph. However a SAS user may wish to exercise a finer control over the colors. For example, instead of using different colors to represent populations, lighter shades of green may be used to represent the sparsely populated states and darker shades of green used to represent more populous states. Such a graph is shown in Figure 18 and is easier to interpret.

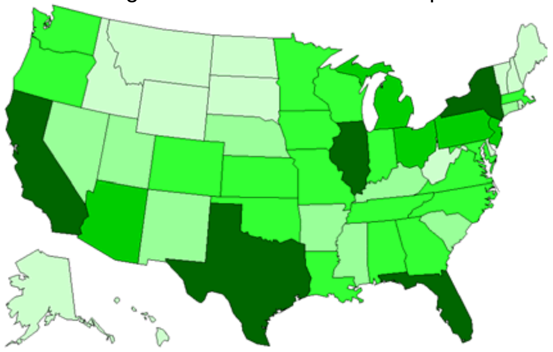


Figure 18. Example of Green Color Scheme

SAS users must carefully consider whether applying many different colors to graphs helps or hurts the overall effect of data presentation. It is important to keep in mind the intended audience. If a graph designer anticipates that a graph will be printed, then using colors may not be the most suitable approach. Additionally, colors may look different depending on operating systems, monitors and devices or some people may be color-blind. In such cases, the author suggests using different fill patterns instead of colors. For example, the population map in Figure 18 may be redesigned as shown in Figure 19:

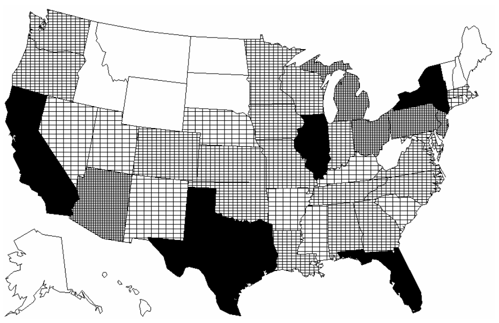


Figure 19. Example of SAS Fill Patterns

This figure is not relying on colors and may be printed by almost any printer. Figure 19 will most likely look the same on any operating system.

CONCLUSION

SAS/GRAPH provides a number of ways to create colorful graphs. This paper demonstrated several approaches to add colors to a graph using various color schemes. SAS users only need to reference the color codes displayed in the charts above. If the desired color is not on the chart then the sample programs can be easily modified to experiment with other colors.

SAS color scheme using plain English (CNS) is the most intuitive scheme because colors and some of their variations are defined in plain English. The CNS color scheme may be sufficient in most of the commonly used graphs. If CNS cannot be used to specify all of the required colors, then other color schemes must be used. SAS users can create virtually any color using SAS color schemes which do not use English words. HLS color scheme is the least difficult to master. COLORMAC macros can help SAS users easily convert color components to color codes. The tables and programs described in this paper can help with choosing the right color code for a given color.

ACKNOWLEDGMENTS

Author would like to thank Perry Watts, Patricia Coyle, Mark Jones and Greg Cicconetti, Ph.D. for their help with this paper.

REFERENCES

Watts, Perry. Working with RGB and HLS Color Coding Systems in SAS® Software. Proceedings of the Twenty-Eighth SAS® User Group International Conference, Cary, NC: SAS Institute Inc., 2003, paper #136.

Watts, Perry. Using ODS and the Macro Facility to Construct Color Charts and Scales for SAS Software Applications. Proceedings of the Twenty-Seventh SAS® User Group International Conference, Cary, NC: SAS Institute Inc., 2002, paper #125.

Watts, Perry. New Palettes for SAS 9 Color Utility Macros. Proceedings of the Twenty-Ninth SAS® User Group International Conference, Cary, NC: SAS Institute Inc., 2004, paper #162.

SAS Institute Inc. (2005), SAS ONLineDoc®, Version 9.1.3, Cary, NC, USA: SAS Institute Inc.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Max Cherny
GlaxoSmithKline
Email: chernym@yahoo.com