Paper PS18

Effective Communication with Colour

LeRoy Bessler, Assurant Health, Milwaukee, USA, bessler@execpc.com

Abstract

Though it includes information and coding specific to SAS®, SAS/GRAPH®, and ODS, this paper is also a software-independent guide to using colour to communicate, rather than decorate. Colour does more than merely add visual excitement to your output. Some visual communication situations require colour—the human eye can reliably distinguish only five shades of gray (actually, five shades of *any* one colour). The commonest form of colour blindness cannot distinguish red from green—yet red, yellow, and green is (unnecessarily) the most popular form of colour-coding for information, the so-called "Traffic Lighting". Furthermore, it is not uncommon to encounter nearly unreadable black text on a medium-to-dark blue background, or even yellow on white. There is a best colour system for web publishing, and a small safe subset of that 16.7-million-colour palette. And there is a most-convenient-to-use colour system for print-only publishing. Colour swatches can be generated with just a few SAS statements, and sample charts for evaluating text-background colour combinations are not much more difficult. For high-volume samples, macro-based colour design tools are also provided. This paper was created with SAS 8.2. The intended audience is all levels of SAS users, and users of any other software that creates colour output.

Introduction

This paper is based on decades of my working with colour for communication, starting in the days when colour devices for computer output were uncommon, were not cheap, and did not deliver very good results. Besides being a user of colour technology, I was involved in evaluation, selection, and deployment of colour displays, colour printers and plotters, and colour copiers (including the first high-resolution, feature-and-function-rich digital copier). My interest in communication with colour is really just one aspect of a wide and deep interest in visual communication of computer-sourced information.

The macros and sample programs in Appendix B are available via email.

I regret that I cannot provide bibliographic citations for the research study results cited immediately below, nor for the remarks attributed later to various experts. Such information is drawn from notes taken from reading and listening many years ago. Work by these people, and others, can be found via web search. For a web search, try both the British spelling "colour", and the American spelling "color".

Note: In cases where the word is used in the context of references within software, and in macros and sample programs developed originally for an American audience, I adopted the spelling "color".

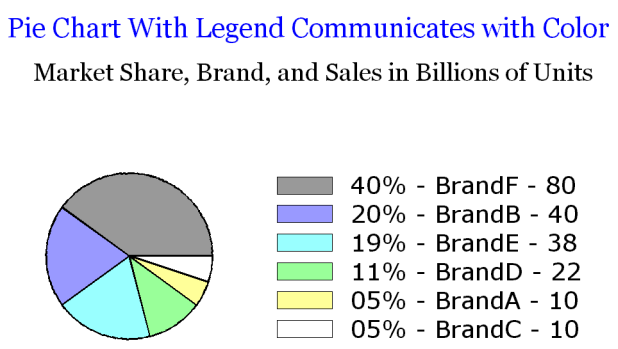
Benefits of Colour Reported in Various Studies

- increased readership
- increased reading speed and comprehension
- faster learning
- reduced error rates
- improved recognition, recall, and response

Using Colour for Communication

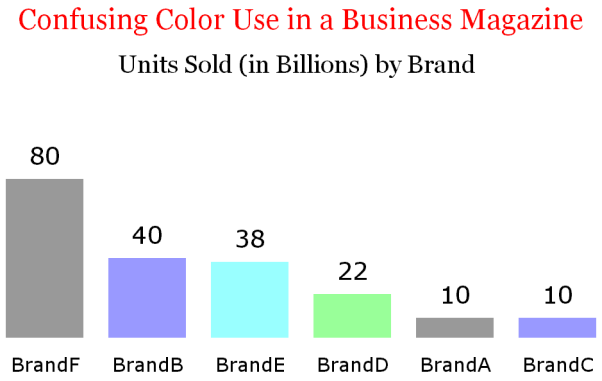
Colour Does Not Improve Bad Design

Use Colour To Communicate, Not To Decorate



This pie chart uses colour to communicate. If your visual communication has no need to distinguish response levels or categories, use Black and White, or some other colour pair for foreground and background. If you have a few levels or categories, gray shades may suffice. If you have many levels or categories, colour is necessary. It is impossible to reliably distinguish more than five shades of a single hue. Of course, you may be able to expand your palette with Black and/or White, depending on the application and the background colour.

Use of Colour Can Confuse, Rather Than Communicate



Viewers attribute significance/meaning to your use of colour, even when none is intended. So, be careful what you do, whenever you use colour. Use of colour without a design objective can disorient, confuse, or even mislead the viewer. Failed person-to-person communication is always the fault of the transmitter, not the receiver. The content of the example at the left is different from the magazine illustration I saw, but the misuse of colour is exactly parallel. There is NO relationship between BrandF and BrandA, and none between BrandB and BrandC.

For Those Who Can't See a Colour Difference, There Is None

The commonest colour blindness cannot distinguish red and green, a frequently used colour combination in the USA. Prof. Jay Neitz of the Eye Institute of the Medical College of Wisconsin reported that 8 to 10 percent of American males have some form of colour blindness, but, due to genetic differences, only about one-half percent of American females. (I cannot provide the bibliographic citation for this information. I read it in a local Milwaukee newspaper several years ago. I expect that you can verify it via a web search.)
*Figure 1 in Appendix A uses "Irish Flag Lighting", but with gray substituted for white (Reference 5).

Maximize Colour Contrast between Text and Background

Contrast between foreground and background is essential to communication. ODS opened the door to "enhancing" tables with colour. Besides the unfortunately popular Traffic Lighting, there are problems using Yellow with White, or Black (or other dark) text on dark or intense background colours. Evaluate the text-background combinations in the illustrations below. See also the contrast demonstration charts in Figures 6 & 7. It should be noted that adequate contrast for online display does not guarantee the same for hardcopy, which is not brightly backlit.

gray black
yellow orange red magenta
green
cyan blue medium blue dark blue

white gray
yellow orange red magenta
green
cyan blue medium blue dark blue

Make Coloured Text and Lines Thicker, Coloured Symbols Bigger

Use of Black and White for print in newspapers, magazines, and books is no accident. Their high contrast makes them the most readable foreground-background combination. Coloured text and lines are harder to see. Coloured lines should be thickened. SAS/GRAPH enables this with the W= option for plot lines in the SYMBOL statement, with the SHAPE=LINE(width-number) option in the LEGEND statement, and with the WOUTLINE= option for the statements used with the GCHART and GMAP PROCs. ODS lets you specify Bold for fonts, and SAS/GRAPH has bold versions of many of its own software fonts, as well as allowing you to use Windows TrueType fonts with Bold (e.g., as in `f='Georgia/Bold'`).

Coloured symbols also need to be bigger. Use `h=` on the SYMBOL statement.

Jan White on Colour Communication

- If everybody screams, all you get is noise. The less colour is used, the more effective it is.
- Colour consistency provides recognition.
- Use colour to sort and/or link information.
- Make large areas pale, small areas bright.
- Don't waste colour on titles—for emphasis, use large or bold print instead.

Michael Turton on Colour Communication

- Colour works better with space around it.
- Colour prioritizes information, whether it is meant to or not.

Aaron Marcus on Colour Communication

- Use blue for large areas, not text or lines. Blue-sensitive colour receptors are the least numerous in the retina's central focusing area.
- Use red or green in the center of the visual field. The edges of the retina are not very sensitive to these colours.

The Case for "UnColour", and How To Use It

When to Use or Not Use Colour on a Graph

- If you have no response levels/categories, use black & white.
- For a few levels or categories, gray shades may suffice.
- For many levels or categories, colour is necessary.

Benefits of Boring Black-and-White

Technology to print black and shades of gray is faster, cheaper, and more reliable.

Black, white, and shades of gray are easier to use. Not only is the equipment simpler, but also their use requires no agonizing over colour selection.

Finally, such output is more copyable. Regardless of the proliferation of cheap colour printers at work and at home, the copiers that you find in abundance in the workplace are still almost always black-and-white. Why does that matter? Well, good graphs, maps, and tables—if hardcopy—will get copied when people want to share them.

SAS/GRAPH Names for Grays

Light Gray, Medium Gray, and Dark Gray (e.g., CXCCCCCC, CX999999, and CX666666), even with White and Black, may not provide enough colours. If so, use colour names of the form GRAY*ll*, where *ll* is a hexadecimal code with range 00-FF. FF (hex for decimal 255) is 0% gray, i.e., WHITE. 00 (hex for decimal 0) is 100% gray, i.e., BLACK. 80 (hex for decimal 128) is 50% gray.

Here are other correspondences for your possible use:

D5: 17%, CC: 20%, C0: 25%, AA: 33%, 99: 40%, 66: 60%, 55: 67%, 40: 75%, 33: 80%, 2B: 83%

Unfortunately, however, the very dark shades of gray tend to be unusable.

How to Choose/Use SAS/GRAPH Grays

Gray shades too close together are difficult or impossible to distinguish.

Here is a theoretical algorithm for creating a gray colour palette. Decide how many grays, N, are needed for the chart, divide 256 by N - 1, and use the quotient (in hexadecimal) as the increment from 00 to FF for *ll* in GRAY*ll* assignments. Subsets of the values provided in the section above can produce equally spaced grays for sets of 3, 4, 5, 6, or 7 PATTERN statements.

As noted earlier, the human eye cannot reliably distinguish more than five shades of gray (or of any other colour), and dark grays are problematic. Hence, characterization of this algorithm as "theoretical".

Sometimes gray shades do not photocopy well. And black text on a gray background can be a problem.

NOTE: Grays with names of the form GRAYll are not browser-safe. See the next section.

For the Web, Use "Browser-Safe" or "Web-Safe" Colours

Why?

Unlikely as it may seem, many web users still have displays or video cards limited to 256 colours. Even when the display and the video card have a higher capability, the video card may be set to display only 256 colours.

To check or change the setting of your video card on a Windows computer, click
Start > Settings > Control Panel > Display > Settings > Colors.

You may wish to design your web pages for the "lowest common denominator". Here is why and how.

To deal with equipment diversity, web browsers determine the currently set limits of the display unit's video card, and, if needed, will remap unsupported colours. (Compare Figures 8 and 9.)

Video displays produce colours as combinations of Red, Green, and Blue, the RGB colour system. All web browsers agree on a universal common subset of 216 browser-safe RGB colours.

They are RGB colours with names, in SAS, of the form CXrrggbb. The web-safe RGB colours restrict rr, gg, and bb to the six values 00, 33, 66, 99, CC, FF, which correspond to 0%, 20%, 40%, 60%, 80%, 100% of red, green, and blue. (216 = 6 X 6 X 6.)

If a web browser detects a colour outside this set on a web page to be shown on a 256-colour display, it remaps the colour to a browser-safe one. Then, Web Designer Colour does not equal Web Viewer Colour. There are 16,777,216 RGB colours, but only 216 are browser-safe.

All of the SAS predefined colour names (see below) and all of the HTML colour names (see below) have RGB equivalents, but only seven of each are browser-safe.

SAS GREEN, contrary to the RGB value still listed in the Version 8 manual, was changed in Version 6.12, and is no longer browser-safe—even though Green is one of the three RGB primaries. The new SAS name, and the HTML name, for browser-safe green is "LIME". I agree that browser-safe green is perhaps not exactly what most people consider to be a "typical" green (but "typical" being vague, imprecise, and inherently subjective). Browser-safe colour CX009900 can serve well as a typical green.

See Figure 2 in Appendix A for 81 samples of browser-safe colours. The basic colours are Red (CXFF0000), Yellow (CXFFFF00), Green (CX00FF00), Cyan or Turquoise (CX00FFFF), Blue (CX0000FF), Magenta (CXFF00FF), Black (CX000000), and White (CXFFFFFF). The upper chart shows the only way for RGB colours to vary in lightness with constant hue.

If you study the full set of 216 browser-safe colours in Figure 5, you may conclude, as I have, that from the browser-safe palette it is difficult to select subsets of "related" colours, other than those in Figure 2. For how to add gray to each of the browser-safe primaries and secondaries, see Figure 3. Another selection of small sets of related browser-safe colours is presented in Figure 4.

NOTE: This seeming limitation can actually be a benefit. A palette of "only" 216 colours does reduce the opportunity for needless agonizing about which colours to use. Presentation of computer-sourced information or charts does not have the same palette requirements as painting a portrait or a landscape.

How To See the Effect of Browser-Unsafe Colours (Compare Figures 8 and 9)

You need a display unit and video card that can display more than 256 colours. As explained in the prior section, use the Control Panel to verify that your video card is set to display more than 256 colours.

Either by using the code in Appendix B, or with any other means that you like, create a web page that includes easily visible patches of SAS Predefined Colours BLUE, TAN, and CREAM. BLUE is one of the only seven web-safe SAS Predefined Colours.

Open the web page with your web browser. The colours will look OK. Close your web browser.

Use the Windows Control Panel to change your video card to display only 256 colours.

Now re-open the web page with your web browser. It will detect the video card's colour impoverishment. You will see that the browser has remapped TAN and CREAM, with the browser-safe colour subset.

Be sure to reset your video card back to its normal setting.

The Best Colour System for Doing Hardcopy Only

It is easy to vary lightness with constant hue by using the HLS colour system. When your target is hardcopy, HLS colours are an excellent choice, also providing easy tunability of transition in hue and "saturation". HLS colour names are of the form H*hhhllss*. Here is how they work:

- *hhh* is the hexadecimal code for Hue
- *ll* is the hexadecimal code for Lightness (also called "Luminance")
- *ss* is the hexadecimal code for Saturation
- *hhh*, *ll*, *ss* ranges are 000-168, 00-FF, 00-FF
- *hhh* = 000 - 168 defines a "wheel of hues", 0 - 360 degrees
- ll = 00 (0%) always produces black, regardless of hue or saturation
- ll = FF (100%) always produces white, regardless of hue or saturation
- ss = FF (100%) always produces the fully saturated hue
- ss = 00 (0%) always produces a gray, regardless of selected hue
- llss = 80FF is what I call the "true colour"

There are six/seven special hues (primary colours and their combinations) in the HLS colour wheel.

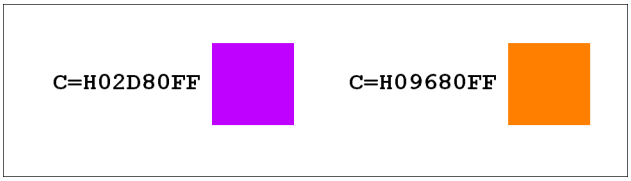
hhh	**colour**	**position**
000	Blue	0 degrees
03C	Magenta	60 degrees
078	Red	120 degrees
0B4	Yellow	180 degrees
0F0	Green	240 degrees
12C	Cyan (Turquoise)	300 degrees
168	Blue	360 degrees

In this scheme, Violet lies between Blue and Magenta, Orange between Red and Yellow, Yellow-Green between Yellow and Green, etc. To get to these other colours, and to adjust their precise hue, you have

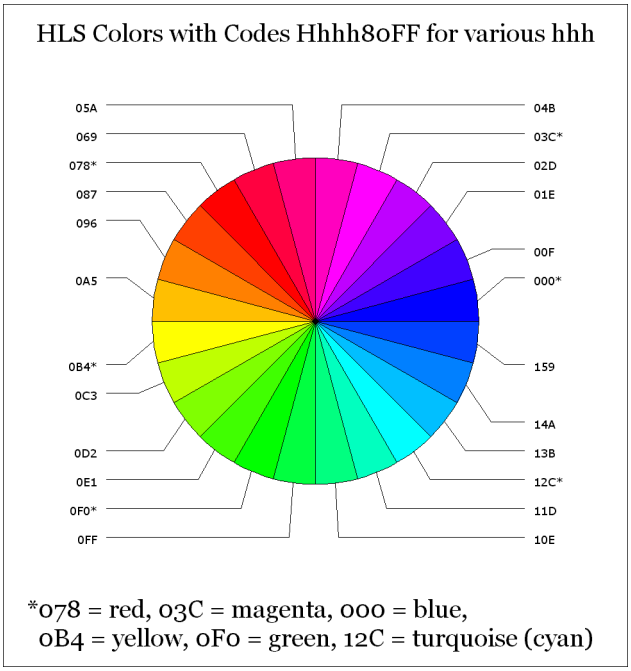
to “turn the dial” between the successive relevant pairs of *hhh* values listed above. Below is code to convert colour wheel degrees into their hexadecimal codes for HLS hues, and to create colour samples:

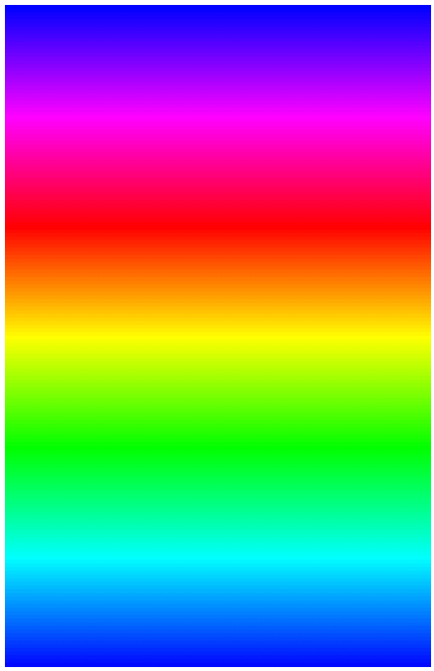
```
data _null_;
length HLScode $8.;
degrees = 45;
HLScode = 'H'||put(degrees,hex3.)||'80FF';
call symput('MyViolet',HLScode);
degrees = 150;
HLScode = 'H'||put(degrees,hex3.)||'80FF';
call symput('MyOrange',HLScode);
run;
goptions reset=all;
goptions device=PNG gsfname=anyname border;
goptions vpos=06 vsize=0.90 IN ymax=0.90 IN ypixels=270;
goptions hpos=34 hsize=3.25 IN xmax=3.25 IN xpixels=975;
filename anyname "YourDrive:\YourFile.png";
proc gslide;
title; footnote; note h=1 ' ';
note j=C                f='Courier New/Bold' h=1 c=H0000000  "C=&MyViolet"
     move=(+0.5,-1.25) f='Monotype Sorts'   h=4 c=&MyViolet '6E'X
     move=(+2,+1.25)   f='Courier New/Bold' h=1 c=H0000000  "C=&MyOrange"
     move=(+0.5,-1.25) f='Monotype Sorts'   h=4 c=&MyOrange '6E'X;
run; quit;
filename anyname clear;
```

Here are the colour samples for “My Violet” and “My Orange”:



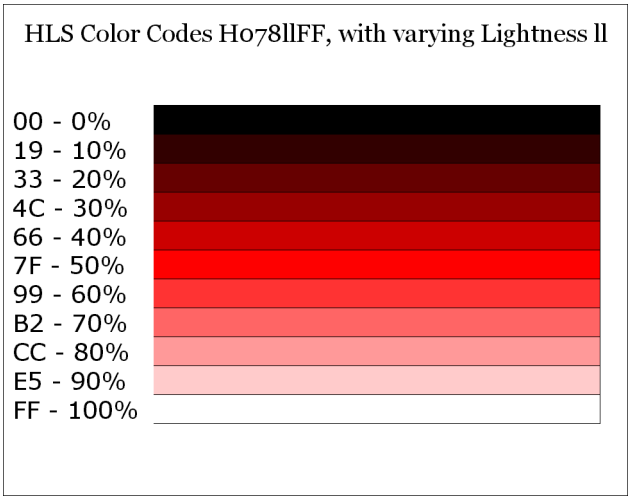
Here is what an HLS Colour Wheel and a much denser, linearized version of it look like:

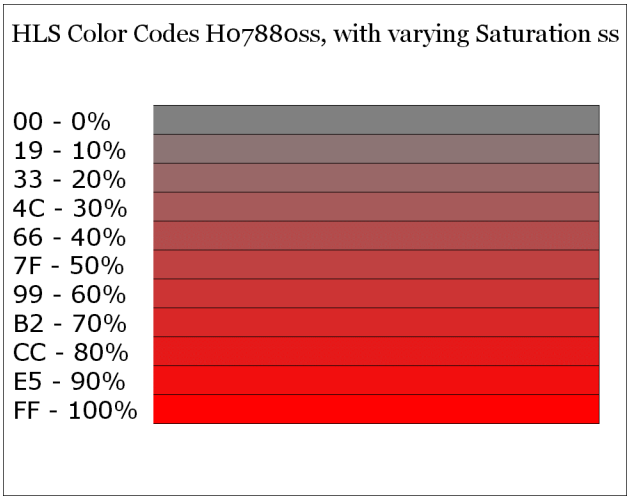




The narrow sharp “signals” at magenta, yellow, and turquoise also occur when the colour wheel is created with narrow slices. They are more pronounced when viewed on a monitor rather than printed.

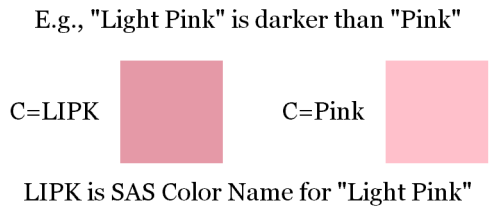
Varying the lightness or the saturation, while holding other parameters constant, is demonstrated below.





Verify What You Will Get from the Predefined Colour Names and HTML Colour Names

There are 292 “SAS Predefined Colour Names”, listed in Table 7.2 in the Version 6 and Version 8 SAS/GRAPH documentation. They have names such as “PINK”, or “LIPK” for “Light Pink”. However, many colours differ from what you would expect based on their name. If you display or print PINK and LIPK, you will find that SAS Light Pink is darker than SAS Pink. See the illustration below. There are other situations like this. Also, many colours are too dark to be useful. Always make colour samples.



Omitting GOPTIONS, here is the code for the above colour sample:

```
proc gslide;
note j=C             f='Georgia'         h=1 c=CX000000 ' C=LIPK'
     move=(+1,-1.5) f='Monotype Sorts' h=5 c=LIPK     '6E'X
     move=(+3,+1.5) f='Georgia'         h=1 c=CX000000 'C=Pink'
     move=(+1,-1.5) f='Monotype Sorts' h=5 c=PINK     '6E'X;
run; quit;
```

If you cannot use Windows TrueType fonts, use f=CENTX, or some other SAS software font, and replace '6E'X with '03'X.

Also, there is another list of 144 SAS colour names, in the SAS Color Registry. You can find them, and their RGB codes, using this click sequence in your Windows SAS session:

Solutions > Accessories > Registry Editor > Colornames > HTML

Of these colour names, only 140 are HTML colour names. Those 140 colours were originally developed for the Unix X Window system, and were later adopted as HTML colour names. (Four of the colours in the SAS registry are SAS abbreviations for HTML Brown, HTML Green, HTML Orange, and HTML Purple. The registry also includes the four corresponding unabbreviated HTML colour names.)

As with the previously discussed SAS predefined colour names, for HTML colour names, too, assume nothing. Make yourself a sample chart. The HTML names are not necessarily reasonable descriptions of the colours. E.g., NavajoWhite is not at all close to White. Though you could describe it as light orange, it is not a faintly orange-tinted white.

Only seven of the HTML colour names intended for web use are browser-safe. Actually there are ten browser-safe HTML colour names, but they include three pairs of colours that are two different names for the same RGB code.

Ironically, the HTML colour name standard palette is not web-safe.

Also, there are three anomalies in the set of SAS HTML colour names. PowderBlue, Turquoise, and PaleTurquoise are in the standard list of HTML colours, but are spelled differently in the SAS Color Registry. If you use the standard HTML name in an ODS program, you will get a message like the following in the SAS log:

```
WARNING: Possible unknown color: PowderBlue. Color will be passed
directly to output destination(s).
```

Fortunately, the resulting output does show the expected colour—HTML recognizes it, but not SAS.

Working with the SAS Color Registry

To get a printable listing in the SAS log, use this:

```
proc registry
startat='HKEY_SYSTEM_ROOT\COLORNAMES'
list;
run; quit;
```

To export the listing to a .txt file, use this:

```
proc registry
startat='HKEY_SYSTEM_ROOT\COLORNAMES'
export='C:\YourFolderName\YourFileName.txt';
run; quit;
```

Using the .txt file created with the code above as a template, you could create your own colour list and import it back into the SAS Color Registry with a different colour list name. That would enable you to use SAS software with your own custom palette, with RGB assignments that you like (e.g., browser-safe ones), and with your own names for them (presumably ones that you regard as reliably descriptive).

NOTE: The long HTML colour names can be used for ODS styles, or in the STYLE parameters with PROC PRINT, PROC REPORT, and PROC TABULATE. However, in SAS Version 8.2, they cannot be assigned with C= in SAS/GRAPH, nor in any other ways that colours are specified in SAS/GRAPH.

Tools for Generating Large Numbers of Sample Colour Combinations or Sample Colours

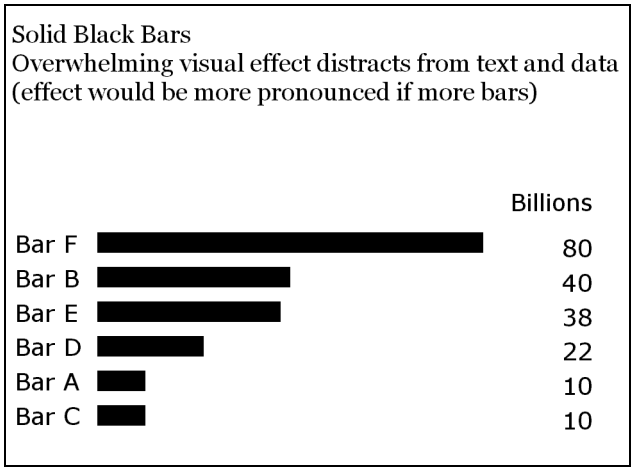
In Appendix B is laborsaving code that can be used to:
(a) evaluate a large number of text (foreground) / background colour combinations for readability; or
(b) create a large number of colour samples.
See Figures 6-9 for examples generated with these tools.

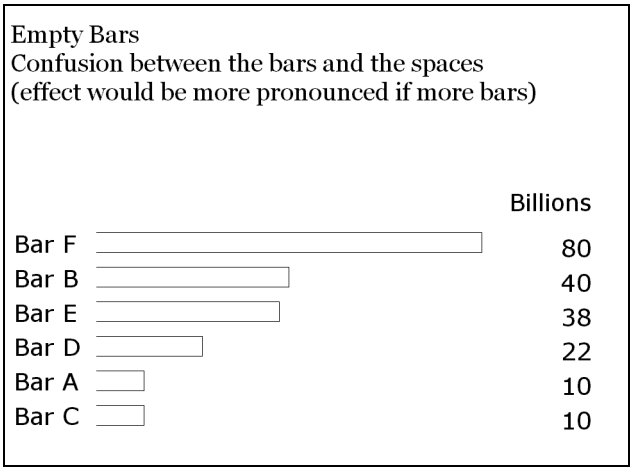
Except for the macro that is used to generate all 216 browser-safe colours, these tools (despite the fact that the macro parameters used to specify colours have the suffix "RGBcolor") can actually be used with any other colour that the SAS System recognizes: HLS colours, the SAS Predefined Colour Names (such as TAN, CREAM, etc.), and any of the long HTML colour names supported by SAS ODS.

The examples do use a custom ODS style preferred by the author. You could substitute any ODS style that you prefer, but your web page background should be white, so as to not affect the visual perception of the colours being evaluated/sampled.

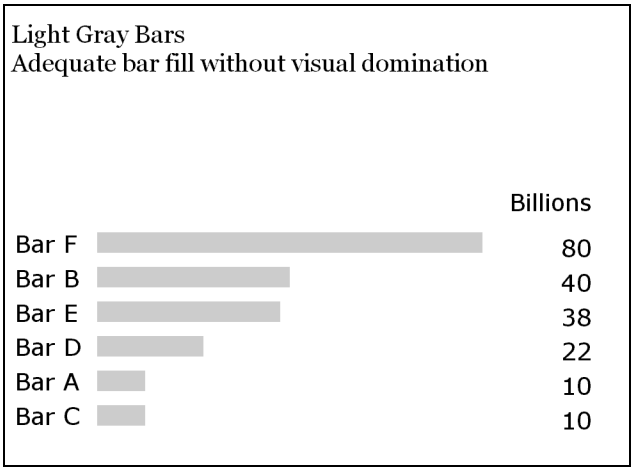
A Light Colour May Be the Right Colour

With many more bars in the example below, the effects of too much or too little colour would be worse than what you can see below.





Below, a light colour creates solid area fill without dominating the visual and information messages.



Colour Differs on Different Media. Do You See What I See?

My wife and I disagree on whether certain colours are green (what I see) or brown (what she sees). But there are more than mere differences in human visual perception. In addition to what can happen when using browser-unsafe colours on the web, and what are called "gamma differences" between PC, Mac, and Unix, there are other technology-related sources of variation. Here are some of them:

- CRT monitor colour and LED flat panel colour differ.
- On an LED panel, very light colours wash out to near-White.
- LED projector colour differs from colour on the presenter's PC or laptop that feeds the projector.
- CRT or LED colour differs from printer colour.
- Hardcopy colour varies from printer to printer.

Among my experiences in colour communication was to see an LED projector convert blue and red text on my PowerPoint slides into violet and orange. This is more than the colour difference phenomena mentioned above. The LED projector is probably the riskiest colour communication tool. Tuning a shared projector to suit one's own laptop is time-consuming, and may impair the usefulness of the projector for some other presenter.

Conclusion

Colour is something that we take for granted. However, without getting into details about the physiology of colour perception, optical illusions due to colour perception, and other arcane subtleties, this paper has shown that colour selection requires care if we want to get beyond mere decoration and into effective communication, and has provided guidelines as well as colour evaluation tools.

Commented List of References (and of Related Work on Data Presentation Design)

1. Bessler, LeRoy (1995), "Communicate Effectively in Color with SAS/GRAPH Software", *Proceedings of the Twentieth Annual SAS Users Group Conference*, Cary, NC, USA: SAS Institute Inc., 1995. The first SUGI paper on communication-effective use of colour.

2. Bessler, LeRoy (2003), "Easy, Elegant, and Effective SAS Graphs: Inform and Influence with Your Data", *Proceedings of the Twenty-Eighth Annual SAS Users Group Conference*, Cary, NC, USA: SAS Institute Inc., 2003. Communication-effective graph design and construction.

3. Bessler, LeRoy (2003), "Web Communication Effectiveness: Design and Methods to Get the Best Out of ODS, SAS, and SAS/GRAPH", *Proceedings of the Twenty-Eighth Annual SAS Users Group Conference*, Cary, NC, USA: SAS Institute Inc., 2003. Communication-effective web design.

4. Bessler, LeRoy and Pierri, Francesca (2001), "Your Graphs and Tables at Their Best on the Web with ODS", *VIEWS 2001 Proceedings*, London, UK: United Kingdom SAS Users Group, 2001. Numerous examples of communication-effective use of ODS, SAS/GRAPH graphs & maps, and PROC PRINT. Also includes how to create a better ODS Table of Contents, and how to create what we call web-page "CrossLinks" with a trick, in order to supplement the standard ODS drill-down hyperlinks. A very ambitious compendium. Page-count constraints limited what we later could publish for SUGI 2002.

5. Bessler, LeRoy (2000), "Show Them Where It's At: Data Mine the Earth's Surface for Locational Significance of What's Hidden in Your Data Warehouse", *Proceedings of the Eighteenth Annual SAS European Users Group International Conference*, Heidelberg, Germany: SAS Institute Inc., 2000. An illustration from the full-colour poster for that black-and-white SEUGI 18 paper has been republished here for SUGI 29 in Figure 1 of Appendix A.

Acknowledgments

I am grateful to the SAS Technical Support staff, who are always willing and able to help. I was originally motivated to become the in-house graphics expert for my employer at that time (Miller Brewing Company) by Alan Paller, who did graphics seminars back in the early days when computer graphics was a novelty rather than the commonplace utility that it is today. In the very early 90's, I was encouraged by Chris Potter, the SUGI Section Chair for what has evolved into the Data Presentation Section, to begin to share my ideas about effective graphic communication. SAS users Steven Subichin and Gary Plazyk taught me some very useful things about PROC GMAP. And it was my SAS user colleague Dr. Francesca Pierri, at Università degli Studi di Perugia, who introduced me to ODS for SAS/GRAPH, and participated with me in some very satisfying and productive transatlantic collaborations, whereby we demonstrated how to get the best out of the SAS System for web publishing of tables, graphs, and maps.

Notices

Author Information

Your questions, comments, suggestions, and other solutions are always welcome.

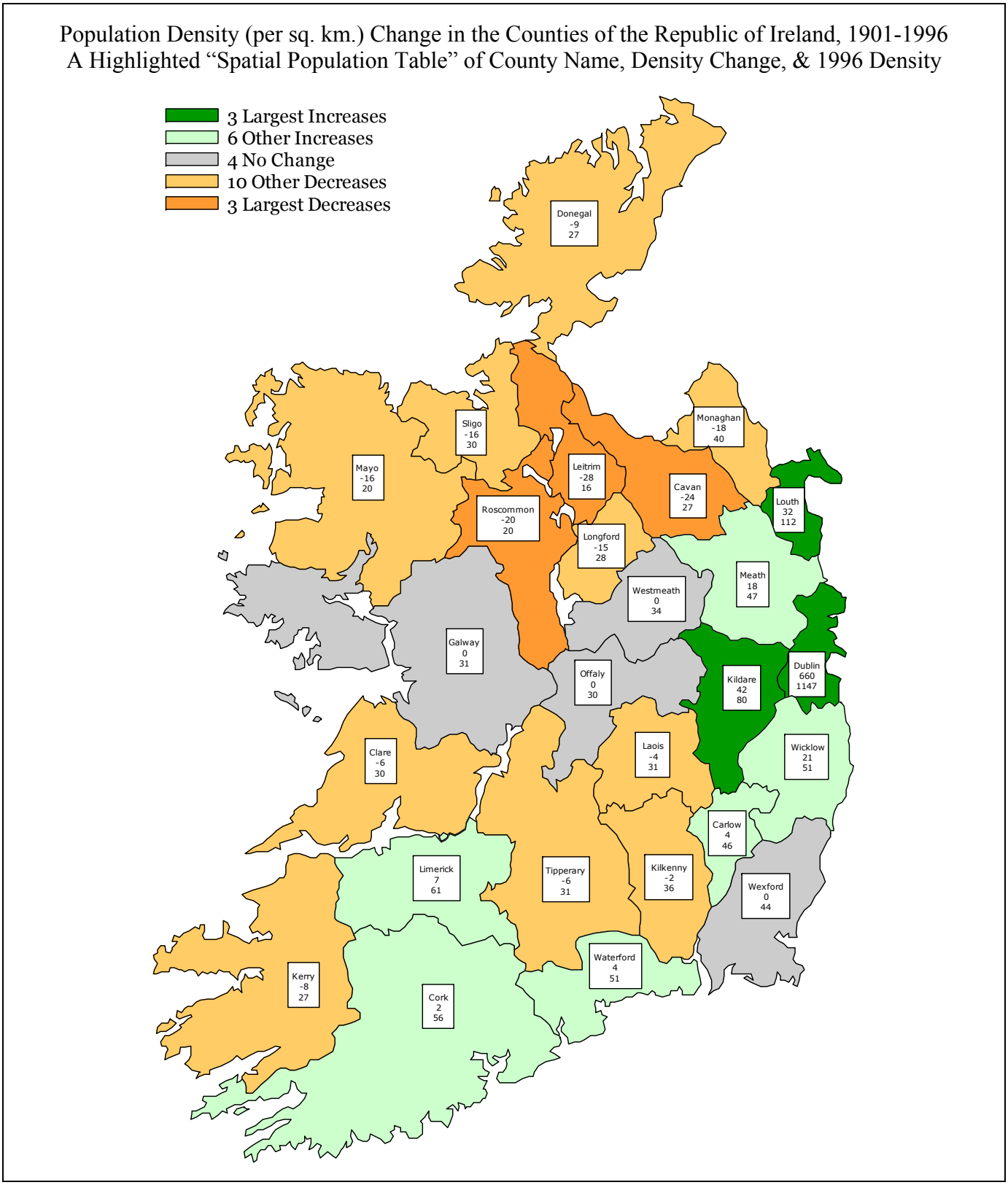
LeRoy Bessler PhD
Email: bessler@execpc.com
Phone: 1 414 351 6748 (evenings and weekends—time is six hours earlier than GMT)

Dr. LeRoy Bessler has special interests in: Software-Intelligent application development, which yields SAS solutions that are reliable, reusable, maintainable, and extendable; and in communication-effective design and construction of reports, tables, graphs, maps, spreadsheets, and web pages.

Please see Appendix A and Appendix B on following pages.

Appendix A.

Figure 1. An Alternative to Traffic Lighting: augmented with light shades of the signal colours.



Shown at the Eighteenth Annual SAS European Users Group International Conference, Dublin, 2000

Figure 2. Samples of Browser-Safe SAS/GRAPH Colours, with Their RGB Codes

Figure 3. Adding Gray to Browser-Safe Primaries and Secondaries

Figure 4. Some Other Sets of "Related" Browser-Safe Colours

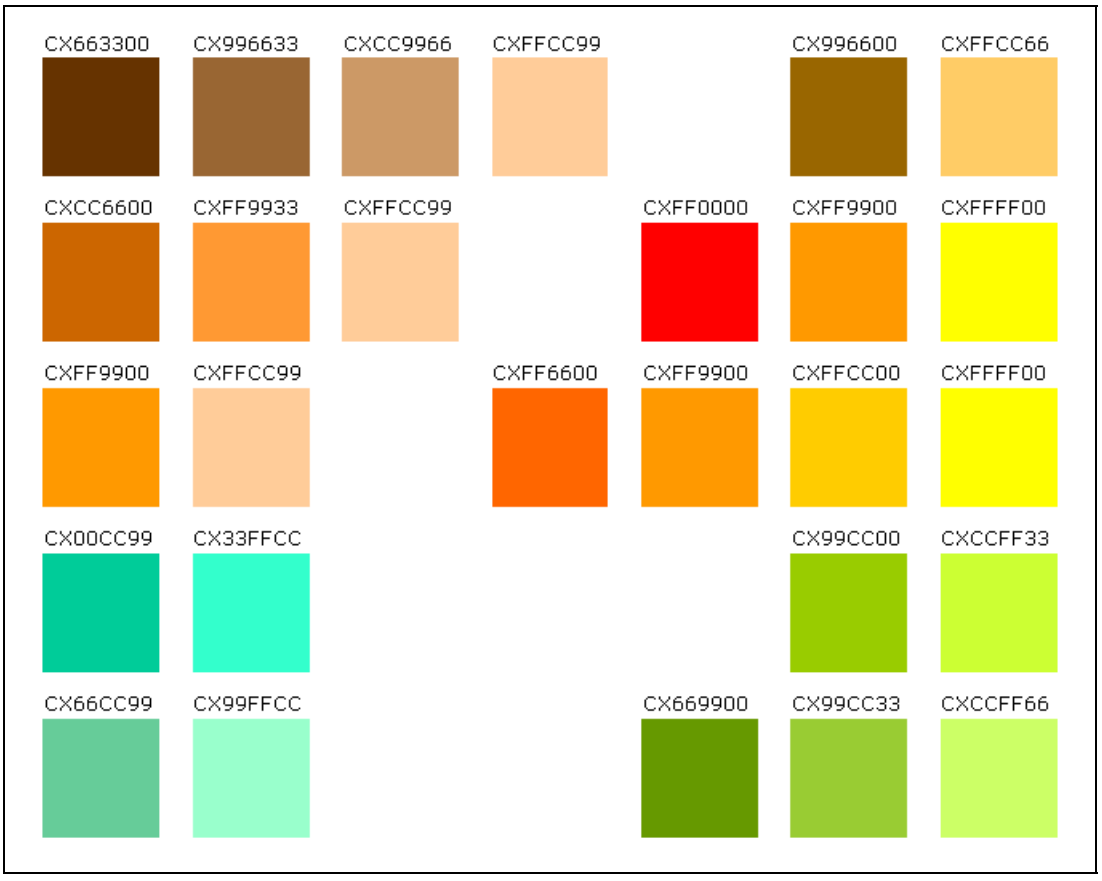


Figure 5. The 216 Browser-Safe Colours with Their RGB Codes

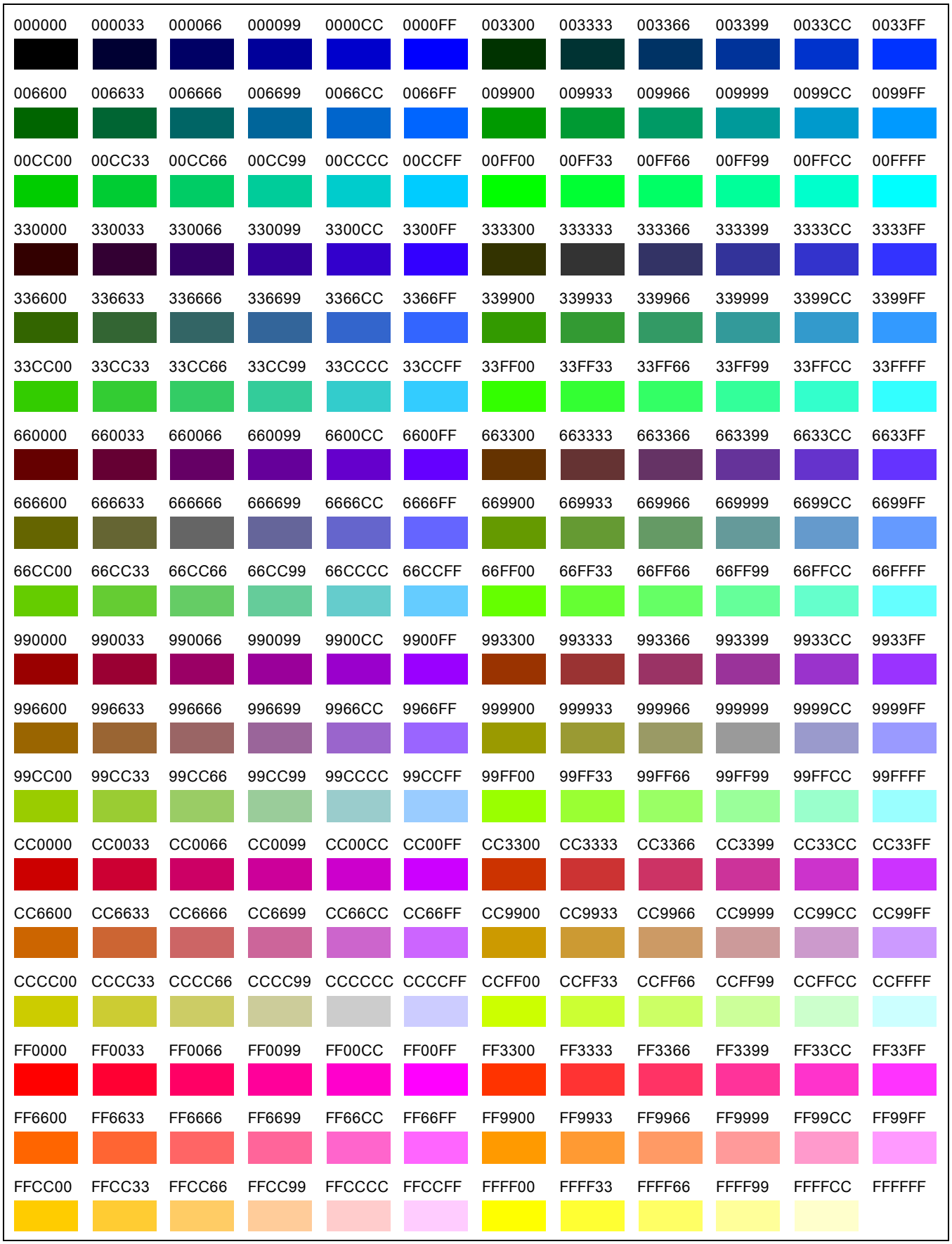


Figure 6. Bad Examples of Text-Background Colour Combinations

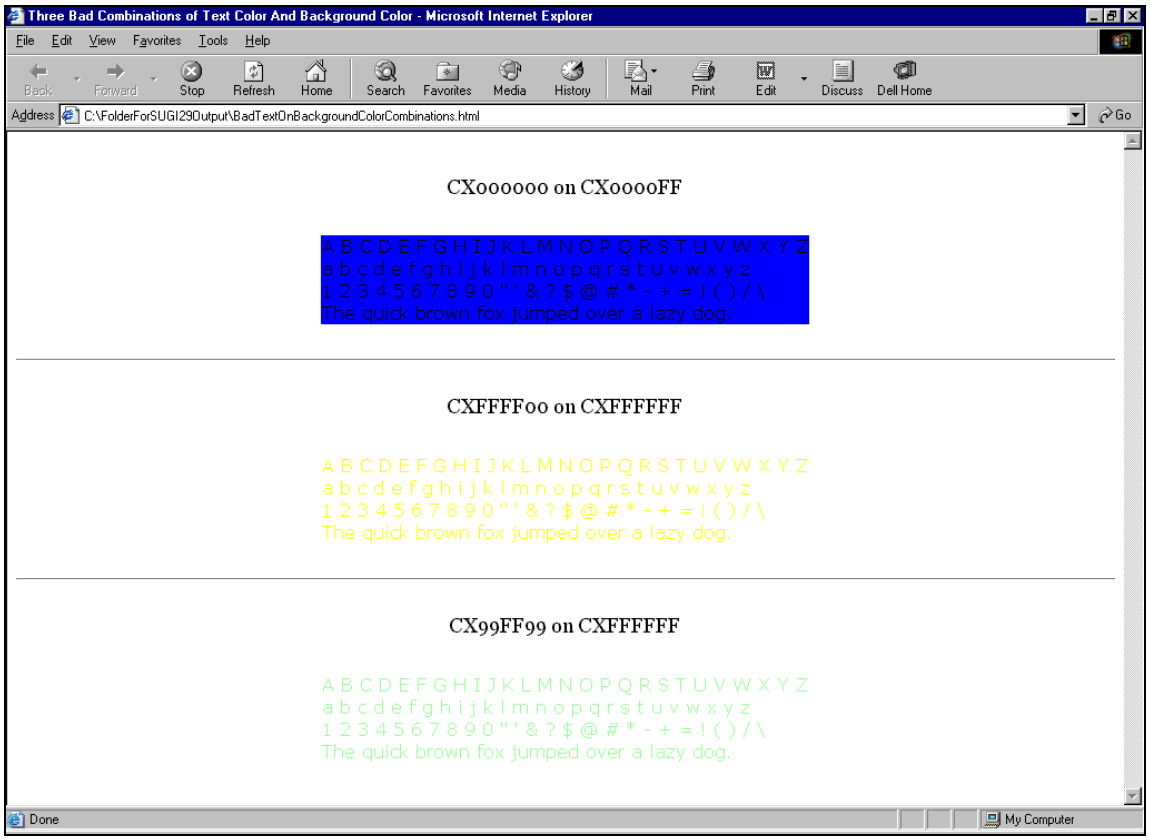


Figure 7. Good Examples of Text-Background Colour Combinations

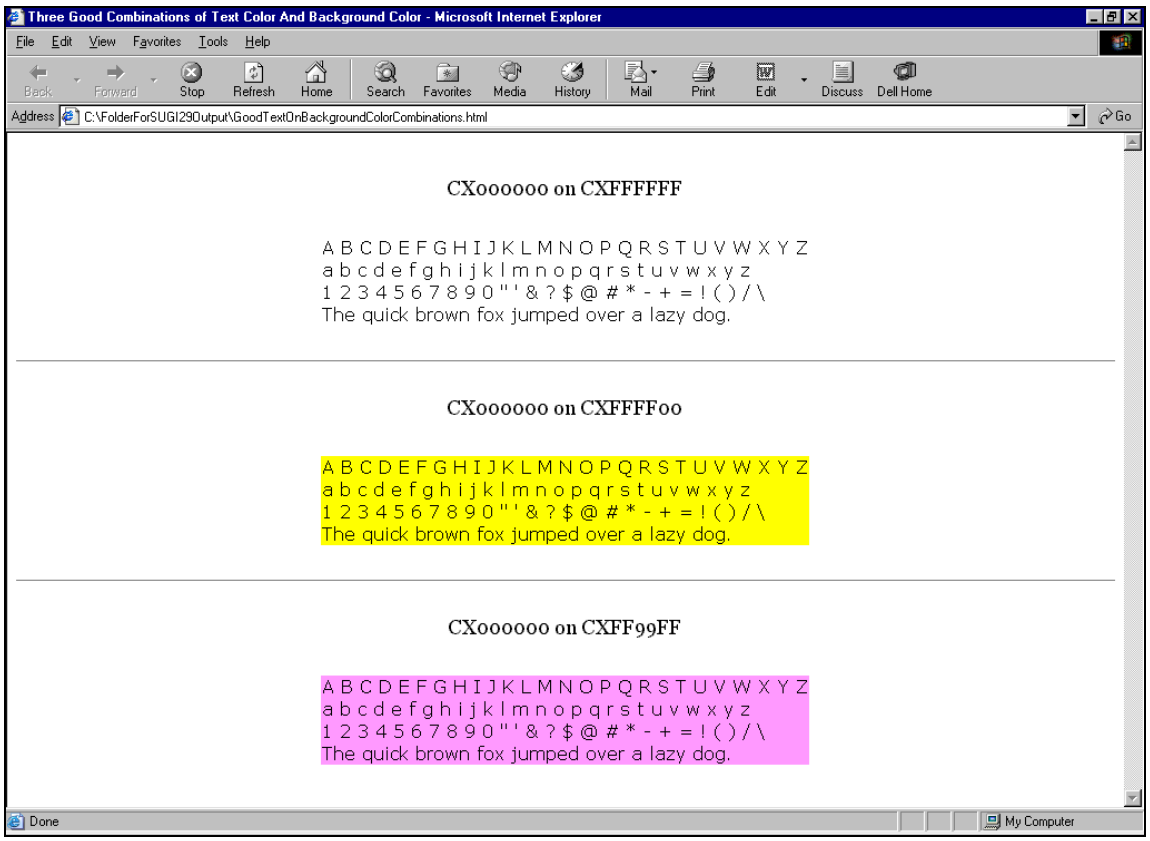


Figure 8. SAS Colours Blue, Tan, & Cream, with My Monitor Set to 32-bit Colour Mode

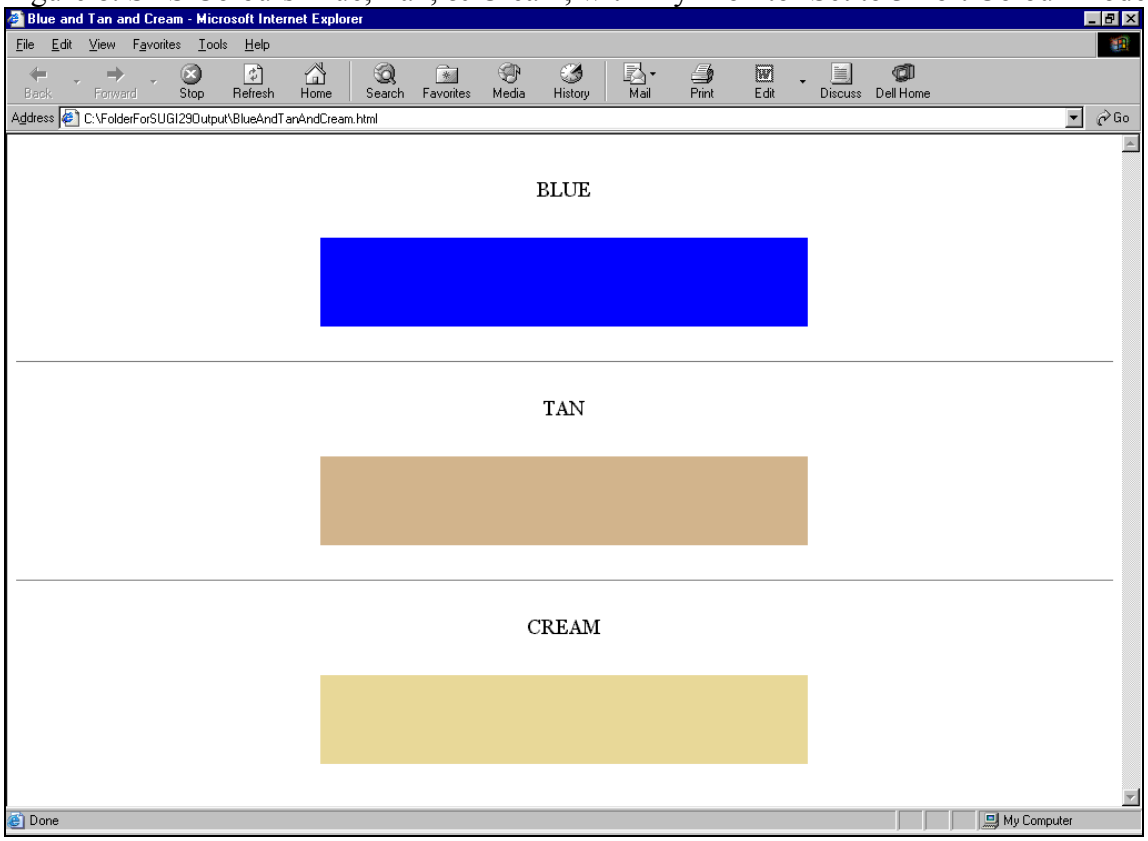
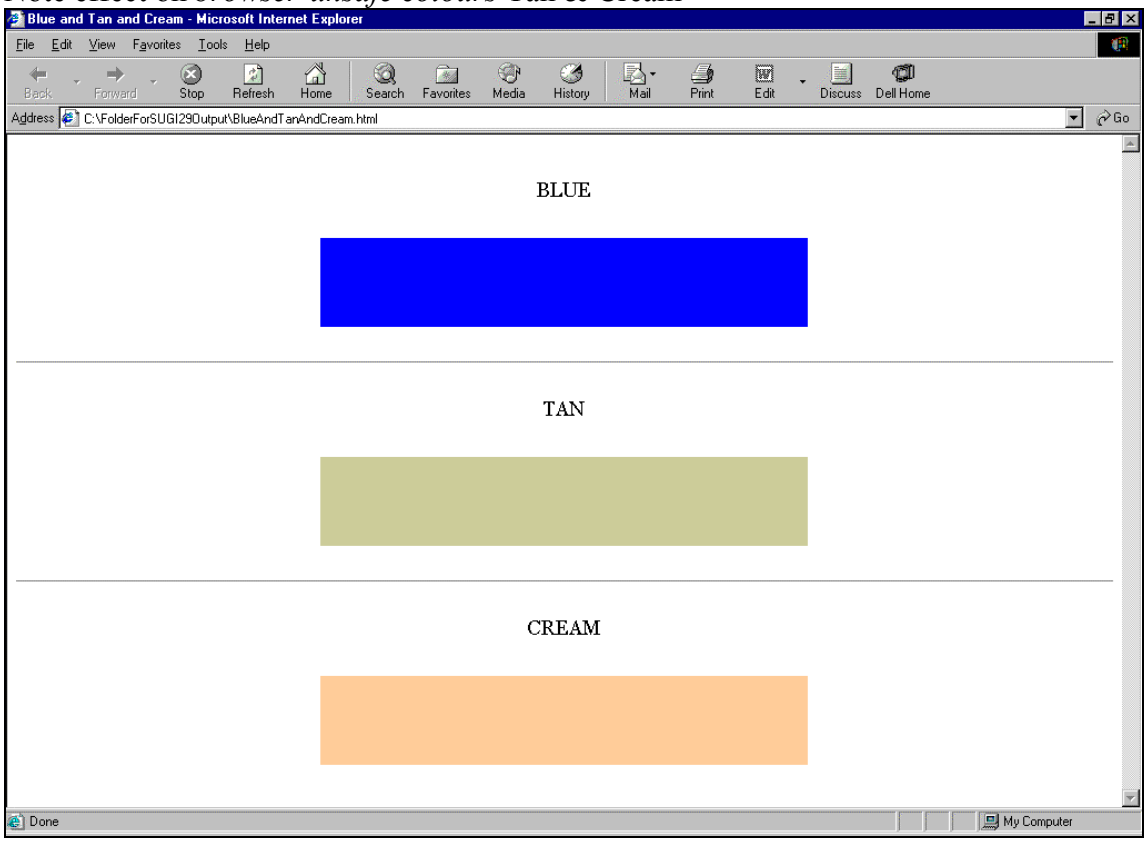


Figure 9. SAS Colours Blue, Tan, & Cream, with My Monitor Set to 256-Colour Mode
Note effect on *browser-unsafe colours* Tan & Cream



Appendix B. Macro Tools and Sample Programs to Evaluate Colours and Combinations

Macro To Create a Custom ODS Style

NOTE: The macro parameters used to specify colours, despite the fact that they have the suffix "RGBcolor", can actually be assigned any colour that the SAS System recognizes.

```
%macro CustomBaseStyleBuild(
StyleName=LeRBsugi29CustomStyle,
PROCoutputSeparators=NO, /* YES to put horizontal rule
  between successive PROC outputs in the same web page,
  but would have NO EFFECT if using NEWFILE = PROC. */
PROCoutputSepLineThickness=1, /* 2 is thicker,
                                 3 is Default */
WebPageBackgroundRGBcolor=CXFFFF99,
    /* CXFFFF99 is light Web-Safe yellow */
TitleFootnoteBackgroundRGBcolor=CXFFFF99,
TitleFootnoteBackgrdTransparency=NO, /* YES to let
  web page background show through */
TitleFootnoteRGBcolor=CX000000, /* Web-Safe black */
TitleFootnoteFont=Georgia,
TitleFootnoteSize=4,
TableBackgroundRGBcolor=CXFFFF99,
TableContentRGBcolor=CX000000, /* data and headings */
TableHeadingFont=Verdana,
TableHeadingSize=3,
TableDataFont=Verdana,
TableDataSize=3,
TableFrame=box, /* TableFrame=void to remove frame */
TableFrameRulesGridRGBcolor=CX9999FF,
                        /* light Web-Safe blue */
TableGrid=NO, /* YES for a grid between data cells */
TableSpacing=5); /* the SAS-shipped default is 7.
  This is space between cell data & cell boundaries. */

proc template;

edit styles.Default as styles.&StyleName;

  /* Create a modified style based
     on the ODS STYLES.DEFAULT.
     Anything not referenced or overridden here
     will be controlled by the ODS Default Style. */

style fonts /

  'TitleFont'   =
   ("&TitleFootnoteFont, Times New Roman, Times",
    &TitleFootnoteSize)
      /* "system" titles & footnotes */

  'HeadingFont' =
   ("&TableHeadingFont , Times New Roman, Times",
    &TableHeadingSize)
 /* column & row headings, and OBS, ID, & SUM values */

  'DataFont'    =
   ("&TableDataFont    , Arial          , Helvetica",
    &TableDataSize)
      /* table data. DataFont being added by LeRB.
         Not in ODS Default style. */

  'DocFont'     = ("Comic Sans MS, Courier",4);
      /* My default for unassigned fonts.
         Conspicuous font chosen to be obvious if used
         by ODS, so that a way can be found to assign a
         preferred font, instead of "my default". */

style color_list /

  'SafeRed'      = CXFF0000  /* Browser-Safe red */
  'SafeBlue'     = CX0000FF  /* Browser-Safe blue */
  'SafeMagenta'  = CXFF00FF  /* Browser-Safe magenta */

  'WebPageBackgroundColor'       =
   &WebPageBackgroundRGBcolor

  'TitleFootnoteBackgroundColor' =
   &TitleFootnoteBackgroundRGBcolor

  'TitleFootnoteColor'           =
   &TitleFootnoteRGBcolor

  'TableBackgroundColor'         =
   &TableBackgroundRGBcolor

  'TableContentColor'            =
   &TableContentRGBcolor

  'TableBoundariesColor'         =
   &TableFrameRulesGridRGBcolor;

style colors /
  'systitlefg'           =
   color_list('TitleFootnoteColor')
     /* "system" titles & footnotes */
  'systitlebg'           =
   color_list('TitleFootnoteBackgroundColor')
     /* background for "system" title/footnote areas
        However, if transparency is enabled,
        then this color is ignored. */
  'headerfg'             =
   color_list('TableContentColor')
     /* override fgA2, which is the
        ODS default for table row & column labels */
  'headerbg'             =
   color_list('TableBackgroundColor')
     /* background for table row & column labels */
  'datafg'               =
   color_list('TableContentColor')
     /* table cell data */
  'databg'               =
   color_list('TableBackgroundColor')
     /* background for table cell data */
  'docfg'                =
   color_list('SafeMagenta')
     /* My default for unassigned foreground colours.
        Conspicuous colour chosen to be obvious if used
        by ODS, so that a way can be found to assign a
        preferred colour, instead of "my default". */
  'docbg'                =
   color_list('WebPageBackgroundColor')
     /* background for web page and ??? */
  'tableborder'          =
   color_list('TableBoundariesColor')
     /* actually, for table frame AND table rules */
  'TableGrid'            =
   color_list('TableBoundariesColor')
     /* (TableGrid is an LeRB replacement for where
         tablebg is used by ODS default style)
         Colour of table grid when cellspacing > 0 AND
         "style Table from Output"
         does not assign background. */
  'link2'                =
   color_list('SafeBlue') /* std for unvisited links */
  'link1'                =
   color_list('SafeRed'); /* std for visited links   */

style SysTitleAndFooterContainer from Container /
  cellpadding = 0 /* compact the title/footnote area */
  cellspacing = 0 /* no grid for title/footnote area */
%if %upcase(&TitleFootnoteBackgrdTransparency) = YES
%then %do;
  background = _undef_;
style systemtitle  / background = _undef_;
style systemfooter / background = _undef_;
     /* Three instances of background = _undef_ above
        make the title and footnote areas transparent.
        They let the web page background show through.
        When this option is selected,
        the systitlebg colour is actually ignored. */
%end;
%else %do;
 ;   /* needed to end this STYLE statement */
%end;

style Output from Container /
     /* these statements control
        table grid and table border */
  rules = NONE /* NONE to override rules=GROUPS,
    preventing double line between table labels & data.
    ALL would create fixed-width thin line
    around all interior cells and
    at inner edges of all perimeter cells */
%if %upcase(&TableGrid) eq NO
```

```
%then %do;
  frame = &TableFrame /* BOX for on, VOID for off */
  cellspacing = 0 /* override 1. Space between cells.
    Also, space between outer cells and any frame. */
%end;
%else %do;
  frame = VOID /* but keeping default cellspacing=1 */
%end;
  cellpadding = &TableSpacing /* override default 7 */
  background = colors('TableGrid')
     /* colour of grid (LeRB replacement for tablebg),
        if cellspacing > 0 AND
        "style Table from Output" does not override it.
   This is NOT the table background on the web page. */
  bordercolor = colors('tableborder')
     /* colour of table frame and table rules */
  borderwidth = 1;
     /* thickness of table frame, same as default */
  /* NOTE: bordercolor affects more table parts
     than does borderwidth. There is no apparent
     way to thicken the rules, when present. */

style Data from Cell / font = fonts('DataFont');
     /* Added to override default use of DocFont */

%if %upcase(&PROCoutputSeparators) = NO %then %do;
style Body / pagebreakhtml = _undef_;
  /* suppress rule between successive PROC outputs */
%end;
%else %do;
style html / 'ThinLineAfterSpace' =
  " <hr size=&PROCoutputSepLineThickness>";
style Body / pagebreakhtml =
  html('ThinLineAfterSpace');
  /* one space and a thin line
     between successive PROC outputs */
%end;

end; run; quit;

%mend CustomBaseStyleBuild;
```

Creating the Custom ODS Style

```
%CustomBaseStyleBuild(
       StyleName=LeRBsugi29ColorDemo,
       TableHeadingSize=1,
       TableFrame=void,
       TableSpacing=1,
       WebPageBackgroundRGBcolor=CXFFFFFF,
                              /* Web-Safe white */
       TableBackgroundRGBcolor=CXFFFFFF,
       TitleFootnoteBackgrdTransparency=YES,
       PROCoutputSeparators=YES);
```

Data for Colour Text Displays

```
data FontCharacters; label FontCharacters='00'X;
infile cards; input @1 FontCharacters $51.;
cards;
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
1 2 3 4 5 6 7 8 9 0 " ' & ? $ @ # * - + = ! ( ) / \
The quick brown fox jumped over a lazy dog.
;
run;
```

Macro to Display Text on a Background or to Create a Solid Colour Sample

```
%macro ColorTable(TextColor=,BackgroundColor=);
title1 "&TextColor"
%if %upcase(&BackgroundColor) ne %upcase(&TextColor)
%then %do;
       " on &BackgroundColor"
%end;
                              ;
proc print data=FontCharacters noobs label
     style(data)=[foreground=&TextColor
                  background=&BackgroundColor];
run;
%mend  ColorTable;
```

Creating Text-Background Colour Combinations and Solid Colour Samples (Figures 6, 7, 8, & 9)

```
ods listing close; ods noresults; goptions reset=all;
ods html
   path    = "C:\FolderForSUGI29Output" (url=none)
   body   = "BadTextOnBackgroundColorCombinations.html"
   (title="A Few Bad Combinations of Text Color And
Background Color")
   style   = Styles.LeRBsugi29ColorDemo
   newfile = NONE; * one continuous scrollable
                     web page body file *;
%ColorTable(TextColor      =CX000000,
            BackgroundColor=CX0000FF);
%ColorTable(TextColor      =CXFFFF00,
            BackgroundColor=CXFFFFFF);
%ColorTable(TextColor      =CX99FF99,
            BackgroundColor=CXFFFFFF);
ods html
   path    = "C:\FolderForSUGI29Output" (url=none)
   body  = "GoodTextOnBackgroundColorCombinations.html"
   (title="A Few Good Combinations of Text Color And
Background Color")
   style   = Styles.LeRBsugi29ColorDemo newfile = NONE;
%ColorTable(TextColor      =CX000000,
            BackgroundColor=CXFFFFFF);
%ColorTable(TextColor      =CX000000,
            BackgroundColor=CXFFFF00);
%ColorTable(TextColor      =CX000000,
            BackgroundColor=CXFF99FF);
ods html
   path    = "C:\FolderForSUGI29Output" (url=none)
   body    = "BlueAndTanAndCream.html"
   (title="Blue and Tan and Cream")
   style   = Styles.LeRBsugi29ColorDemo newfile = NONE;
%ColorTable(TextColor      =BLUE,
            BackgroundColor=BLUE);
%ColorTable(TextColor      =TAN,
            BackgroundColor=TAN);
%ColorTable(TextColor      =CREAM,
            BackgroundColor=CREAM);
ods html close; ods listing;
```

Macro and Program for Scrollable Display of Every Browser-Safe Colour

```
%macro AllBrowserSafeColors;
%do RRdecimal = 0 %to 255 %by 51;
  %do GGdecimal = 0 %to 255 %by 51;
    %do BBdecimal = 0 %to 255 %by 51;
data _null_;
DecimalCode = &RRdecimal;
call symput('RR',put(DecimalCode,hex2.));
DecimalCode = &GGdecimal;
call symput('GG',put(DecimalCode,hex2.));
DecimalCode = &BBdecimal;
call symput('BB',put(DecimalCode,hex2.));
run;
%ColorTable(TextColor      =CX&RR&GG&BB,
            BackgroundColor=CX&RR&GG&BB);
    %end;
  %end;
%end;
%mend  AllBrowserSafeColors;
ods listing close; ods noresults; goptions reset=all;
ods html
   path    = "C:\FolderForSUGI29Output" (url=none)
   body    =
   "AllOfTheBrowserSafeColors.html" (title=" . . . ")
   style   = Styles.LeRBsugi29ColorDemo newfile = NONE;
%AllBrowserSafeColors;
ods html close; ods listing;
```