Viewing Object Colors in a Gallery

Glenn Davis <gdavis@gluonics.com>

December 18, 2021

Introduction

The goal of this **colorSpec** vignette is to display rendered images of a popular color target with different illuminants, both with and without chromatic adaption methods. The figures are best viewed on a display calibrated for sRGB. Featured functions in this vignette are: `extradata()`, and `product()`.

```
library( colorSpec )
library( spacesXYZ )     # for function standardXYZ()
library( spacesRGB )     # for functions RGBfromXYZ() and plotPatchesRGB()
```

Read the target spectra. This data has been kindly provided in CGATS format by [2]. *ColorChecker* is a Registered Trademark of X-Rite, and X-Rite is a Trademark.

```
# read the Macbeth ColorCheck target
path = system.file( 'extdata/targets/CC_Avg30_spectrum_CGATS.txt', package='colorSpec')
MacbethCC = readSpectra( path )            # MacbethCC is a 'colorSpec' object
MacbethCC = MacbethCC[ order(MacbethCC$SAMPLE_ID), ]  # still class 'colorSpec'
print( extradata(MacbethCC), row.names=F )
```

SAMPLE_ID	SAMPLE_NAME	Munsell	ISCC-NBS_Name	LEFT	TOP	WIDTH	HEIGHT
1	dark skin	3YR 3.7/3.2	moderate brown	7	9	29	29
2	light skin	2.2YR 6.47/4.1	light reddish brown	40	9	29	29
3	blue sky	4.3PB 4.95/5.5	moderate blue	73	9	29	29
4	foliage	6.7GY 4.2/4.1	moderate olive green	106	9	29	29
5	blue flower	9.7PB 5.47/6.7	light violet	139	9	29	29
6	bluish green	2.5BG 7/6	light bluish green	172	9	29	29
7	orange	5YR 6/11	strong orange	7	42	29	29
8	purplish blue	7.5PB 4/10.7	strong purplish blue	40	42	29	29
9	moderate red	2.5R 5/10	moderate red	73	42	29	29
10	purple	5P 3/7	deep purple	106	42	29	29
11	yellow green	5GY 7.1/9.1	strong yellow green	139	42	29	29
12	orange yellow	10YR 7/10.5	strong orange yellow	172	42	29	29
13	Blue	7.5PB 2.9/12.7	vivid purplish blue	7	75	29	29
14	Green	0.25G 5.4/8.65	strong yellowish green	40	75	29	29
15	Red	5R 4/12	strong red	73	75	29	29
16	Yellow	5Y 8/11.1	vivid yellow	106	75	29	29
17	Magenta	2.5RP 5/12	strong reddish purple	139	75	29	29
18	Cyan	5B 5/8	strong greenish blue	172	75	29	29
19	white	N9.5/	white	7	108	29	29
20	neutral 8	N8/	light gray	40	108	29	29
21	neutral 6.5	N6.5/	light medium gray	73	108	29	29

```
22     neutral 5          N5/       medium gray  106 108    29    29
23   neutral 3.5         N3.5/        dark gray  139 108    29    29
24        black           N2/            black  172 108    29    29
```

Note that `MacbethCC` is organized as `'df.row'` and contains extra data for each spectrum, notably the coordinates of the patch rectangle.

Viewing with Illuminant D65

Build the "material responder" from Illuminant D65 and standard CMFs:

```
D65.eye = product( D65.1nm, "artwork", xyz1931.1nm, wave='auto' )
#   calibrate so the perfect-reflecting-diffuser is the 'official XYZ'
#   scale XYZ independently
PRD = neutralMaterial( 1, wavelength(D65.eye) )
D65.eye = calibrate( D65.eye, stimulus=PRD, response=standardXYZ('D65'), method='scaling' )
```

Calculate XYZ and then RGB:

```
XYZ = product( MacbethCC, D65.eye, wave='auto' )
RGB = RGBfromXYZ( XYZ, space='sRGB', which='scene' )$RGB   # this is *signal* sRGB
# add the rectangle data to RGB, so they can be plotted in proper places
obj = extradata(MacbethCC)
obj$RGB = RGB
#   display in proper location, and use the sRGB display transfer function
par( omi=c(0,0,0,0), mai=c(0.2,0.2,0.2,0.2) )
plotPatchesRGB( obj, space='sRGB', which='signal', back='gray20', labels=FALSE )
```
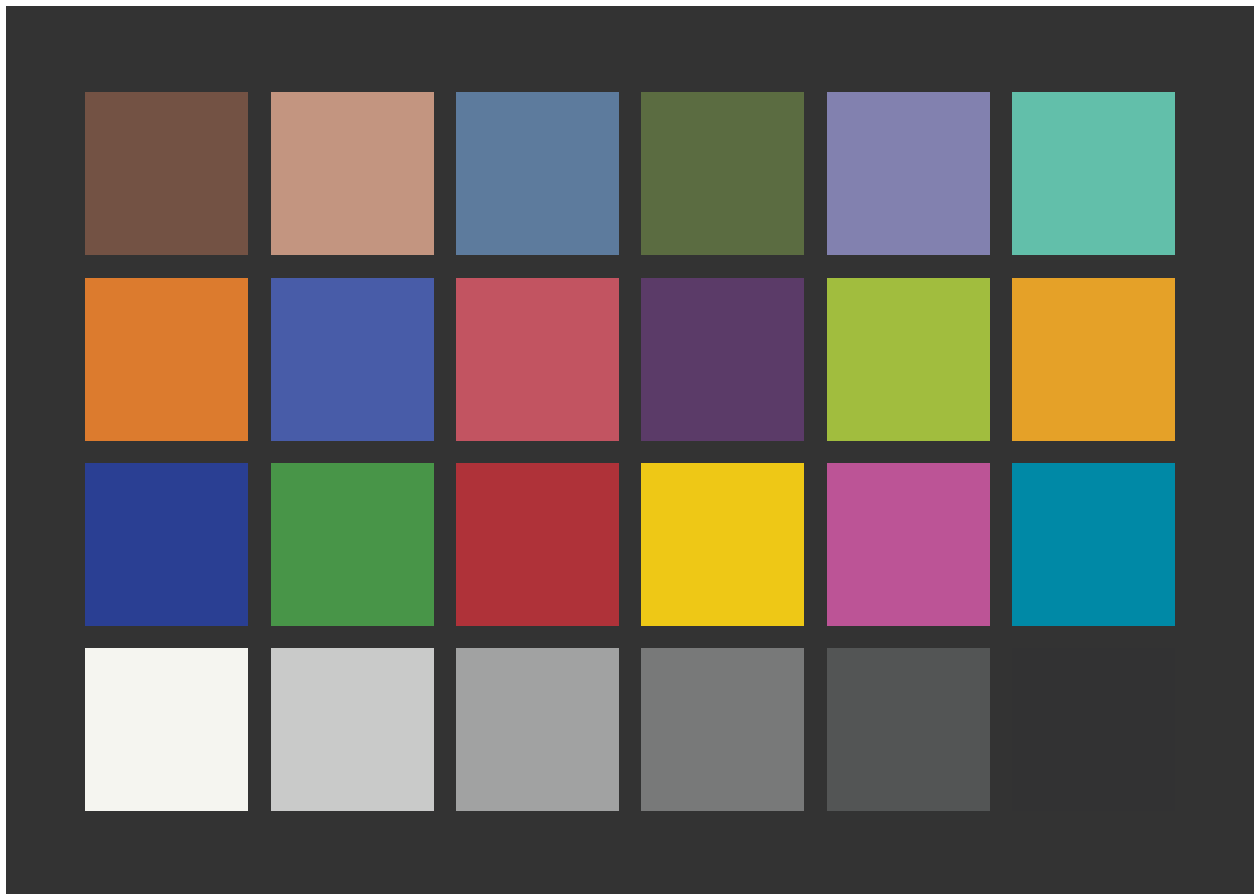
Figure 1: Rendering with Illuminant D65 and xyz1931.1nm

```
obj.first = obj    # save this reference object for later
```

Here are the 8-bit device values:

```
RGB8 = round( 255 * RGB )
print( RGB8 )

                R    G    B
dark skin      115   82   68
light skin     195  149  128
blue sky        93  123  157
foliage         91  108   65
blue flower    130  129  175
bluish green    98  191  170
orange         220  123   46
purplish blue   72   92  168
moderate red   194   84   97
purple          91   59  104
yellow green   161  189   62
orange yellow  229  161   40
Blue            42   63  147
Green           72  149   72
Red            175   50   57
Yellow         238  200   22
Magenta        188   84  150
```

```
Cyan             0 137 166
white          245 245 240
neutral 8      201 202 201
neutral 6.5    161 162 162
neutral 5      120 121 121
neutral 3.5     83  85  85
black           50  50  51
```

Note that all of these patches are inside the sRGB gamut, exept for Cyan.

Another way to do the same thing is use the built-in theoretical camera BT.709.RGB that computes sRGB directly from spectra, and has already been calibrated.

```
RGB = product( D65.1nm, MacbethCC, BT.709.RGB, wave='auto' ) # this is *linear* sRGB
obj = extradata(MacbethCC)
obj$RGB = RGB
par( omi=c(0,0,0,0), mai=c(0.2,0.2,0.2,0.2) )
plotPatchesRGB( obj, space='sRGB', which='scene', back='gray20', labels=FALSE )
```
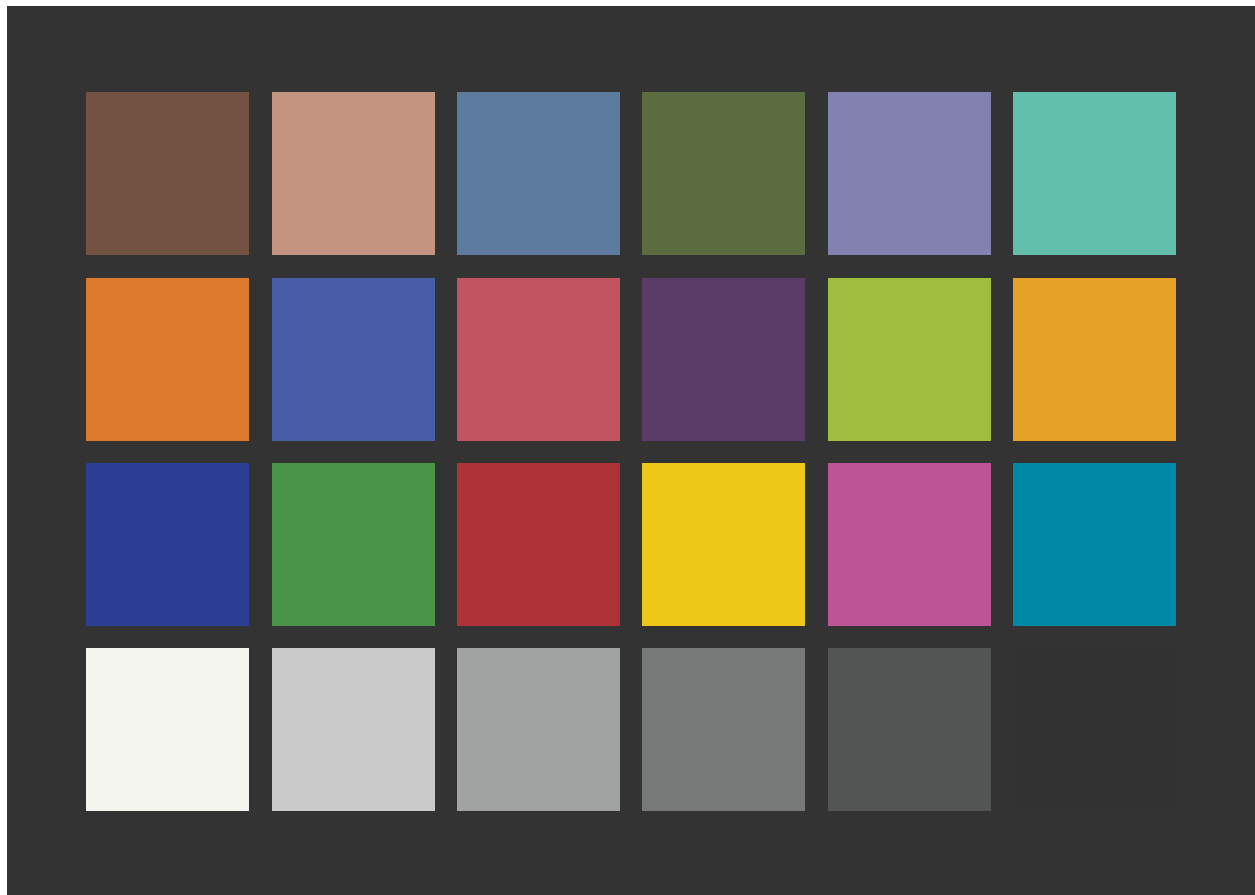

Figure 2: Rendering with Illuminant D65 and Theoretical BT.709.RGB Camera

Viewing with Illuminant D50

Build the "material responder" from Illuminant D50 and standard CMFs:

```
D50.eye = product( D50.5nm, "artwork", xyz1931.5nm, wave='auto' )
#   calibrate so the response to the perfect-reflecting-diffuser is the 'official XYZ' of D50
#   scale XYZ independently
PRD = neutralMaterial( 1, wavelength(D50.eye) )
D50.eye = calibrate( D50.eye, stimulus=PRD, response=standardXYZ('D50'), method='scaling' )
```

Calculate XYZ and then RGB:

```
XYZ = product( MacbethCC, D50.eye, wave='auto' )
obj = extradata(MacbethCC)
obj$RGB = RGBfromXYZ( XYZ, space='sRGB' )$RGB      # this is *signal* sRGB
par( omi=c(0,0,0,0), mai=c(0.2,0.2,0.2,0.2) )
plotPatchesRGB( obj, space='sRGB', which='signal', back='gray20', labels=FALSE )
```


Figure 3: Rendering with Illuminant D50 and xyz1931.5nm

Since D50 is yellower than D65, the result has a yellow cast. Start over, but this time calibrate and adapt to D65 using the Bradford method.

```
D50.eye = product( D50.5nm, "artwork", xyz1931.5nm, wave='auto' )
# calibrate so the response to the perfect-reflecting-diffuser is the 'official XYZ' of D65
# with this chromatic adaption the destination XYZ is a 3x3 matrix times the source XYZ
PRD = neutralMaterial( 1, wavelength(D50.eye) )
XYZ.D65 = standardXYZ('D65')
```

```r
D50toD65.eye = calibrate( D50.eye, stimulus=PRD, response=XYZ.D65, method='Bradford' )
XYZ = product( MacbethCC, D50toD65.eye, wave='auto' )
obj = extradata(MacbethCC)
obj$RGB = RGBfromXYZ( XYZ, space='sRGB' )$RGB        # this is *signal* sRGB
par( omi=c(0,0,0,0), mai=c(0.2,0.2,0.2,0.2) )
plotPatchesRGB( obj, space='sRGB', which='signal', back='gray20', labels=FALSE )
```


Figure 4: Rendering with Illuminant D50 and xyz1931.5nm, but then adapted to D65

The white-balance here is much improved. But it hard to compare colors in this figure with the ones way back in Figure 1. So combine the original D65 rendering in Figure 1 with this D50 rendering in Figure 4 by splitting each square into 2 triangles. We can do this by setting add=T in the second plot.

```r
par( omi=c(0,0,0,0), mai=c(0.2,0.2,0.2,0.2) )
plotPatchesRGB( obj.first, space='sRGB', back='gray20', labels=F )
plotPatchesRGB( obj, space='sRGB', labels=F, shape='bottomright', add=T )
```

Figure 5: Rendering with both D65 (Figure 1), and D50 then adapted to D65 (Figure 4)

The top-left triangle has the color from Figure 1 and the bottom-right triangle has the color from Figure 4. There is a noticeable difference in the **Red** and **Magenta** patches.

A Rendering with a Scanner

Here we calculate a rendering on an RGB scanner. This is not really a gallery situation, but illustrates the similarity of the 2 RGB calculations.

```
#   Build a scanner from Illuminant F11 and the Flea2 camera
scanner = product( subset(Fs.5nm,'F11'), 'artwork', Flea2.RGB, wave='auto' )
#   calibrate scanner so the response to the perfect-reflecting-diffuser is RGB=(1,1,1)
#   set the RGB gains independently
PRD = neutralMaterial( 1, wavelength(scanner) )
scanner = calibrate( scanner, stimulus=PRD, response=1, method='scaling' )
obj = extradata(MacbethCC)
obj$RGB = product( MacbethCC, scanner, wave='auto' ) # this linear RGB is not lin-
ear sRGB
par( omi=c(0,0,0,0), mai=c(0.2,0.2,0.2,0.2) )
plotPatchesRGB( obj, space='sRGB', which='scene', back='gray20', labels=FALSE )
```
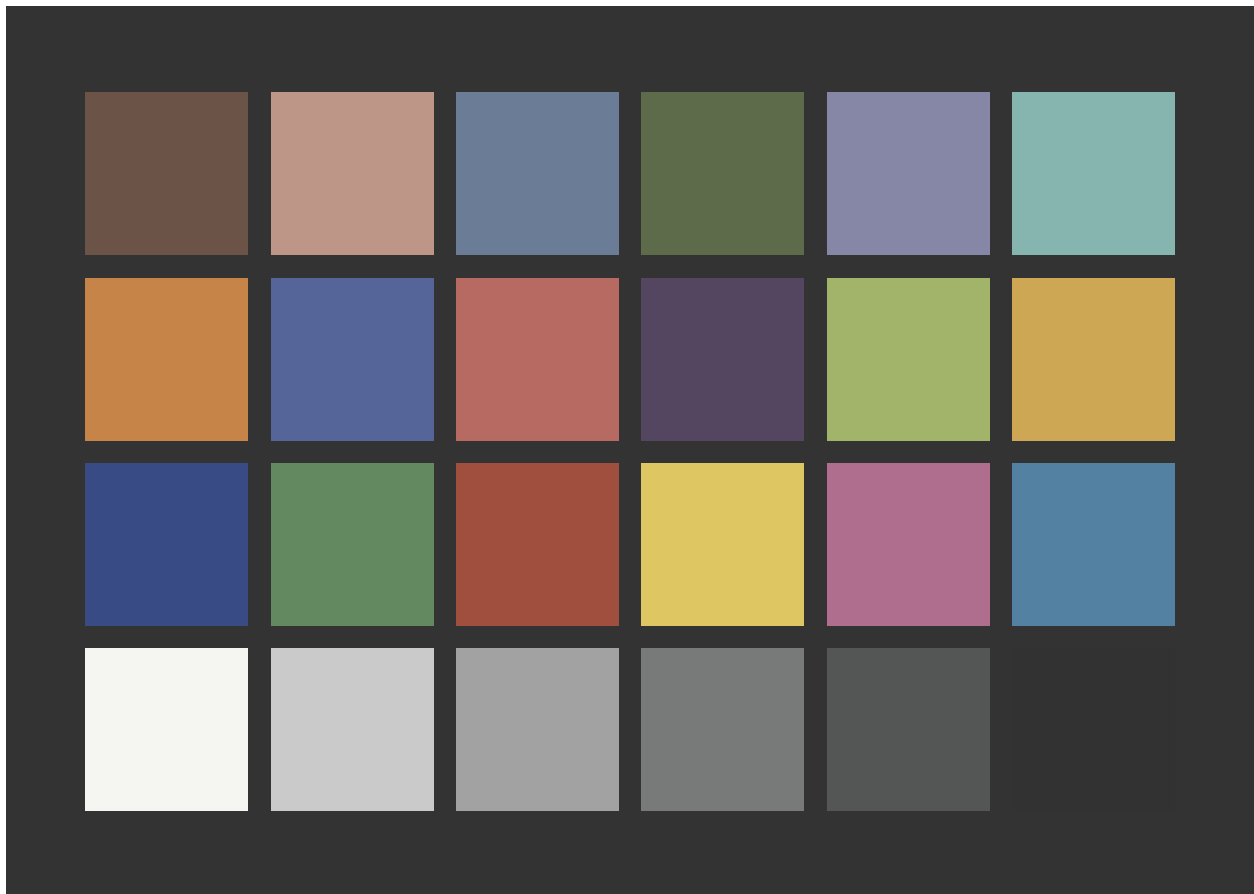
Figure 6: Rendering with a generic RGB scanner

The colors are too pale; this time `Cyan` has a substantial Red signal. Some sort of color management is necessary in this scanner to improve accuracy.

For an interactive viewer along these lines, see [1].

References

[1] Lindbloom, Bruce. GretagMacbeth ColorChecker Calculator. `http://brucelindbloom.com/index.html?ColorCheckerCalculator.html`.

[2] Pascale, Danny. The ColorChecker, page 2. `http://www.babelcolor.com/colorchecker-2.htm`.

Appendix

This document was prepared December 18, 2021   with the following configuration:

- R version 4.1.2 (2021-11-01), `x86_64-w64-mingw32`

- Running under: `Windows 10 x64 (build 19042)`

- Matrix products: default

- Base packages: base, datasets, grDevices, graphics, methods, stats, utils

- Other packages: colorSpec 1.3-0, knitr 1.37, spacesRGB 1.4-0, spacesXYZ 1.1-1

- Loaded via a namespace (and not attached): MASS 7.3-54, compiler 4.1.2, digest 0.6.28, evaluate 0.14, fastmap 1.1.0, highr 0.9, htmltools 0.5.2, magrittr 2.0.1, microbenchmark 1.4.9, rlang 0.4.12, rmarkdown 2.11, stringi 1.7.5, stringr 1.4.0, tools 4.1.2, xfun 0.28, yaml 2.2.1