Viewing Object Colors in a Gallery

Glenn Davis <gdavis@gluonics.com>
April 1, 2020

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 ) 
```

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

```r
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:

```r
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

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

Here are the 8-bit device values:

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

<table>
<thead>
<tr>
<th>Color</th>
<th>R</th>
<th>G</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>dark skin</td>
<td>115</td>
<td>82</td>
<td>68</td>
</tr>
<tr>
<td>light skin</td>
<td>195</td>
<td>149</td>
<td>128</td>
</tr>
<tr>
<td>blue sky</td>
<td>93</td>
<td>123</td>
<td>157</td>
</tr>
<tr>
<td>foliage</td>
<td>91</td>
<td>108</td>
<td>65</td>
</tr>
<tr>
<td>blue flower</td>
<td>130</td>
<td>129</td>
<td>175</td>
</tr>
<tr>
<td>bluish green</td>
<td>98</td>
<td>191</td>
<td>170</td>
</tr>
<tr>
<td>orange</td>
<td>220</td>
<td>123</td>
<td>46</td>
</tr>
<tr>
<td>purplish blue</td>
<td>72</td>
<td>92</td>
<td>168</td>
</tr>
<tr>
<td>moderate red</td>
<td>194</td>
<td>84</td>
<td>97</td>
</tr>
<tr>
<td>purple</td>
<td>91</td>
<td>59</td>
<td>104</td>
</tr>
<tr>
<td>yellow green</td>
<td>161</td>
<td>189</td>
<td>62</td>
</tr>
<tr>
<td>orange yellow</td>
<td>229</td>
<td>161</td>
<td>40</td>
</tr>
<tr>
<td>Blue</td>
<td>42</td>
<td>63</td>
<td>147</td>
</tr>
<tr>
<td>Green</td>
<td>72</td>
<td>149</td>
<td>72</td>
</tr>
<tr>
<td>Red</td>
<td>175</td>
<td>50</td>
<td>57</td>
</tr>
<tr>
<td>Yellow</td>
<td>238</td>
<td>200</td>
<td>22</td>
</tr>
<tr>
<td>Magenta</td>
<td>188</td>
<td>84</td>
<td>150</td>
</tr>
</tbody>
</table>
Note that all of these patches are inside the sRGB gamut, except for Cyan.

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

\begin{verbatim}
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), mai=c(0.2,0.2,0.2,0.2) )
plotPatchesRGB( obj, space='sRGB', which='scene', back='gray20', labels=FALSE )
\end{verbatim}

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 )

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')

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.
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.

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.

```r
# 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 linear 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


Appendix

This document was prepared April 1, 2020 with the following configuration:

- R version 3.6.3 (2020-02-29), i386-w64-mingw32
- Running under: Windows 7 (build 7601) Service Pack 1
- Matrix products: default
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
• Other packages: colorSpec 1.2-1, knitr 1.28, spacesRGB 1.3-0, spacesXYZ 1.1-1

• Loaded via a namespace (and not attached): MASS 7.3-51.5, Rcpp 1.0.3, compiler 3.6.3, digest 0.6.25, evaluate 0.14, highr 0.8, htmltools 0.4.0, magrittr 1.5, microbenchmark 1.4-7, rlang 0.4.4, rmarkdown 2.1, stringi 1.4.6, stringr 1.4.0, tools 3.6.3, xfun 0.12, yaml 2.2.1