Plotting Chromaticity Loci of Optimal Colors

Reproducing plots of MacAdam Limits

Glenn Davis <gdavis@gluonics.com>

January 27, 2024

Introduction

The goal of this vignette is to reproduce 2 figures in [Gun82], and to make similar ones. The featured function from **colorSpec** used in this vignette is `sectionOptimalColors()`. But it requires some help from the function `plotOptimals()` and others in the file `optimal-help.R`.

```r
library( colorSpec )
source( "optimal-help.R" )

# make vctor of levels to be used for all the plots
Ylevel=c( seq( 0.10, 0.90, by=0.1 ), 0.95 )
```

Illuminant A

First, build the "material responder" from Illuminant A and standard CMFs:

```r
wave  = seq(380,800,by=5)
A.eye = product( A.1nm, "material", xyz1931.1nm, wavelength=wave )
white = product( neutralMaterial(1,wave=wave), A.eye )
seclist = sectionOptimalColors( A.eye, normal=c(0,1,0), beta=white[2]*Ylevel )
```

Now, make the plot:

```r
par( omi=rep(0,4), mai=c(0.5,0.6,0,0) )
plotOptimals( seclist, Ylevel, xyz1931.1nm, white )
```
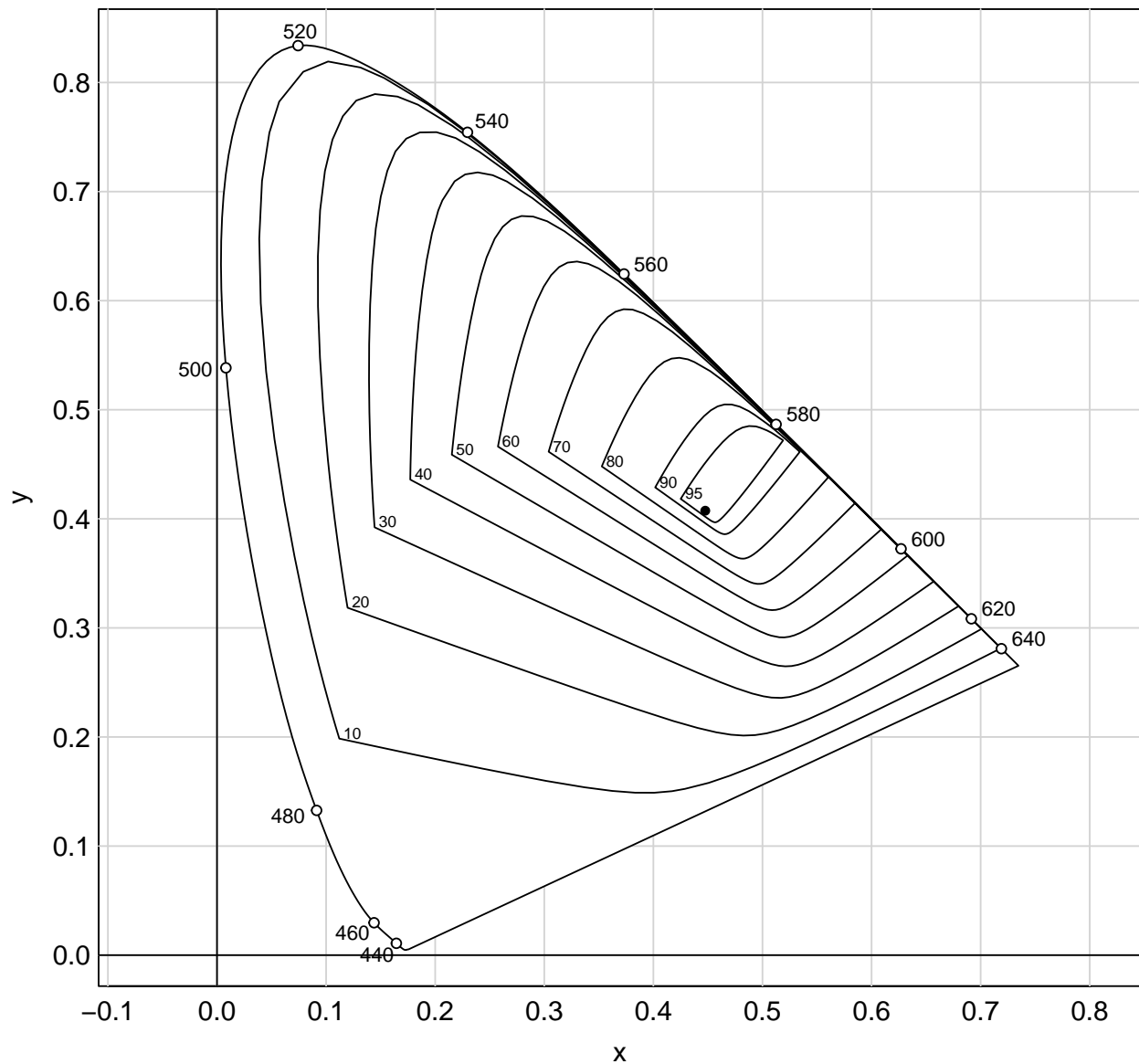
Figure 1: MacAdam Limits for Illuminant A

Compare this with Figure 3(3.7) from [Gun82].

Illuminant D65

First, build the "material responder" from Illuminant D65 and standard CMFs:

```
wave   = seq(380,800,by=5)
D65.eye = product( D65.1nm, "material", xyz1931.1nm, wavelength=wave )
white = product( neutralMaterial(1,wave=wave), D65.eye )
seclist = sectionOptimalColors( D65.eye, normal=c(0,1,0), beta=white[2]*Ylevel )
```

Make the plot:

```
par( omi=rep(0,4), mai=c(0.5,0.6,0,0) )
plotOptimals( seclist, Ylevel, xyz1931.1nm, white )
```
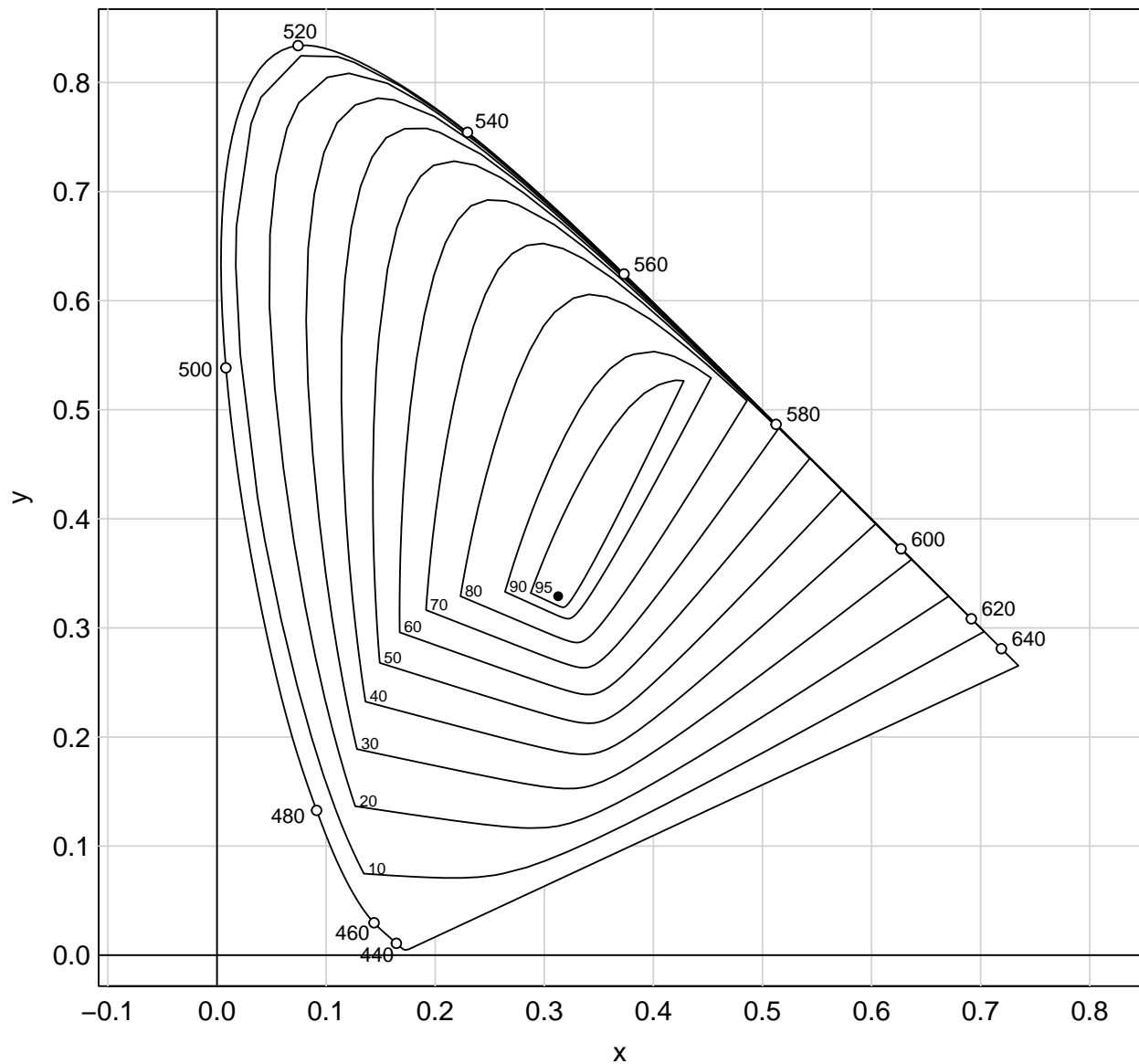
Figure 2: MacAdam Limits for Illuminant D65

Compare this with Figure 4(3.7) from [Gun82].

Illuminant C

First, build the "material responder" from Illuminant C and standard CMFs:

```
wave  = seq(380,780,by=5)
C.eye = product( C.5nm, "material", xyz1931.1nm, wavelength=wave )
white = product( neutralMaterial(1,wave=wave), C.eye )
seclist = sectionOptimalColors( C.eye, normal=c(0,1,0), beta=white[2]*Ylevel )
```

Now, make the plot:

```
par( omi=rep(0,4), mai=c(0.5,0.6,0,0) )
plotOptimals( seclist, Ylevel, xyz1931.1nm, white )
```
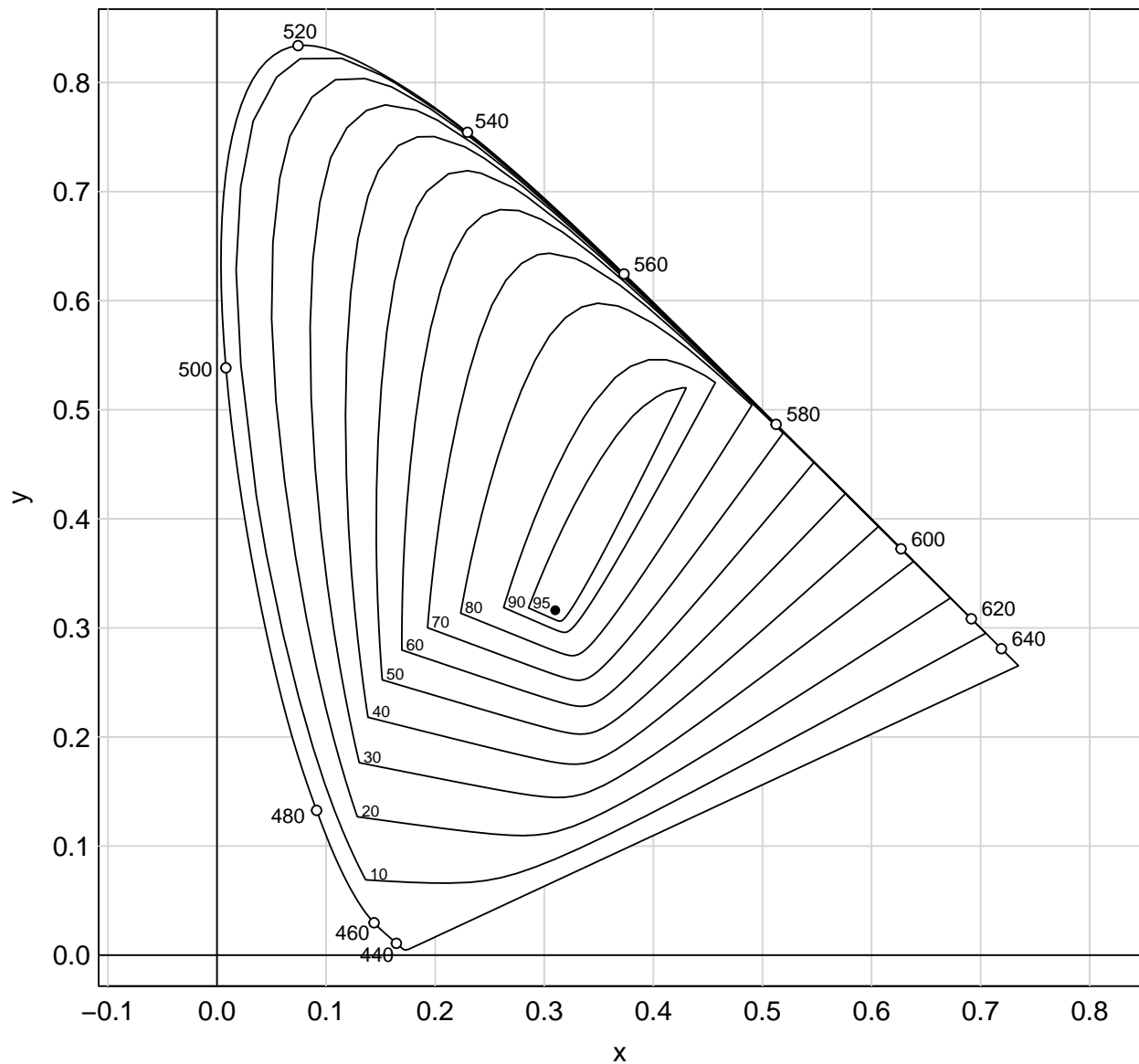
Figure 3: MacAdam Limits for Illuminant C

An RGB Scanner

This also works with object color from an electrical RGB scanner. The chromaticities in this case are:

$$r = R/(R + G + B) \qquad g = G/(R + G + B)$$

Make a scanner from a tungsten source and a Flea2 camera:

```
wave = seq(420,680,by=5)
Flea2.scanner = product( A.1nm, "material", Flea2.RGB, wavelength=wave )
white = product( neutralMaterial(1,wave=wave), Flea2.scanner )
seclist = sectionOptimalColors( Flea2.scanner, normal=c(0,1,0), beta=white[2]*Ylevel )
```

Now, make the plot:

```
par( omi=rep(0,4), mai=c(0.5,0.6,0,0) )
plotOptimals( seclist, Ylevel, Flea2.scanner, white )
```
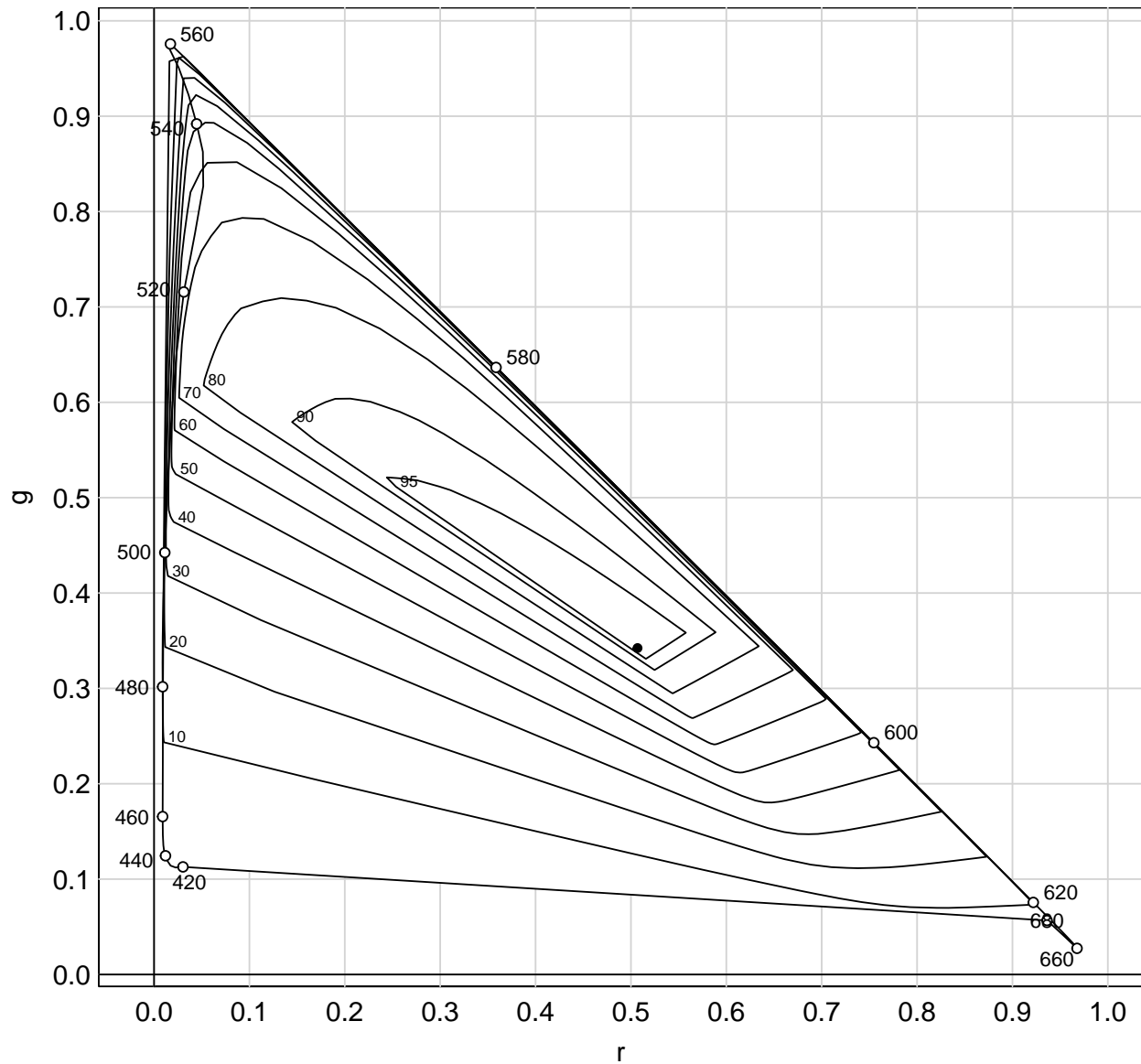

Figure 4: Approximate Output Limits for an RGB scanner

The wavelengths have been trimmed at each end to avoid weak responsivities that wander around too much. Even after trimming, the spectrum locus is not convex in the inverval from about 500 to 560 nm. This shows that the color solid does *not* satisfy the 2-transition property. The optimal color chromaticity locii are convex, as they must be; and some optimal colors are outside the spectrum locus. See Figure 3 and more discussion in [Wes83].

References

[Gun82] Gunther Wyszecki and W.S. Stiles. *Color Science : Concepts and Methods, Quantitative Data and Formulae.* Wiley-Interscience, second edition, 1982.

[Wes83] West, G. and Brill, M. H. Conditions under which Schrödinger object colors are optimal. *Journal of the Optical Society of America*, 73:1223–1225, 1983.

Appendix

This document was prepared January 27, 2024   with the following configuration:

- R version 4.3.2 (2023-10-31 ucrt), `x86_64-w64-mingw32`

- Running under: `Windows 10 x64 (build 19045)`

- Matrix products: default

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

- Other packages: colorSpec 1.5-0, knitr 1.42, spacesRGB 1.5-0, spacesXYZ 1.2-1

- Loaded via a namespace (and not attached): MASS 7.3-60, R6 2.5.1, arrangements 1.1.9, bslib 0.4.2, cachem 1.0.8, cli 3.6.1, compiler 4.3.2, digest 0.6.31, evaluate 0.21, fastmap 1.1.1, gmp 0.7-1, highr 0.10, htmltools 0.5.5, jquerylib 0.1.4, jsonlite 1.8.4, microbenchmark 1.4.10, rlang 1.1.1, rmarkdown 2.21, rootSolve 1.8.2.3, sass 0.4.6, tools 4.3.2, xfun 0.39, yaml 2.3.7