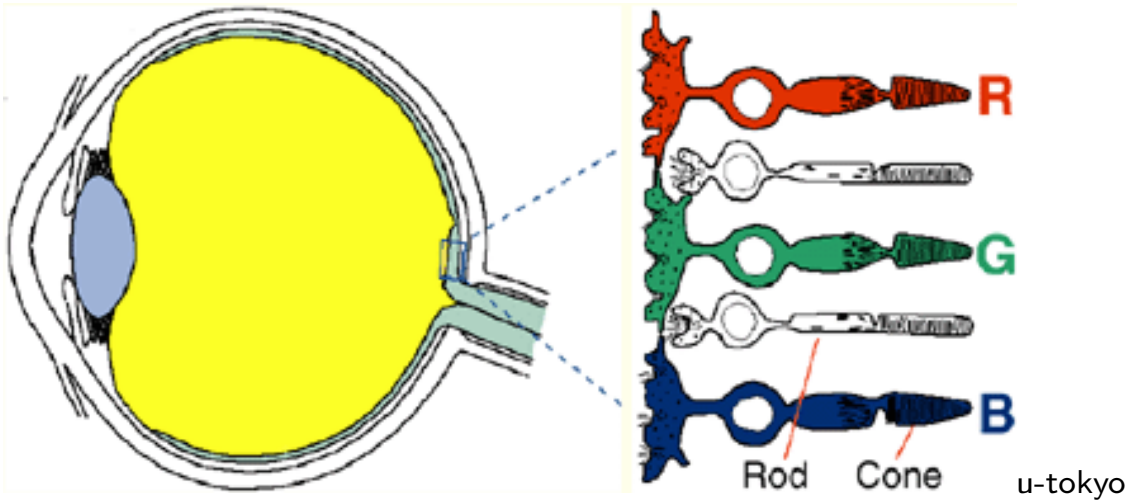
PyCoffee @ Vitacura: True-colour rendering and images manipulation using Python

Daniel Moser

Associate Post-doc

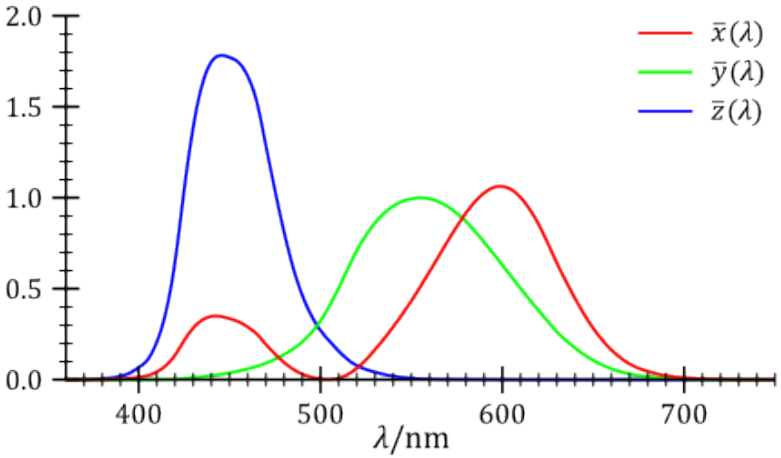
Feb 25, 2016

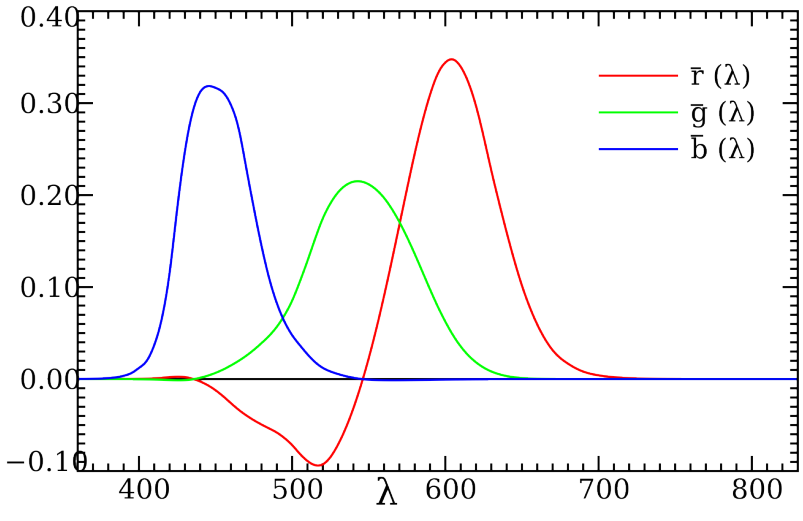
- Human eye is an optical device.
  Its photoreceptors are the rod and cone cells.
- Rod cells work in low brightness conditions ("night vision").
- Cone cells (SML) work in medium and high brightness conditions and are responsible for color vision.



- Three parameters (the stimulus of the three types of cone cells) can *in principle* describe any color sensation (color space).

- *Commission internationale de l'éclairage* (CIE) = international authority on light, illumination, colour, and colour spaces.
- CIE 1931 RGB color space: link between wavelength intensities and physiological perception in human color vision. Color sensation described in terms of Red, Green and Blue values.

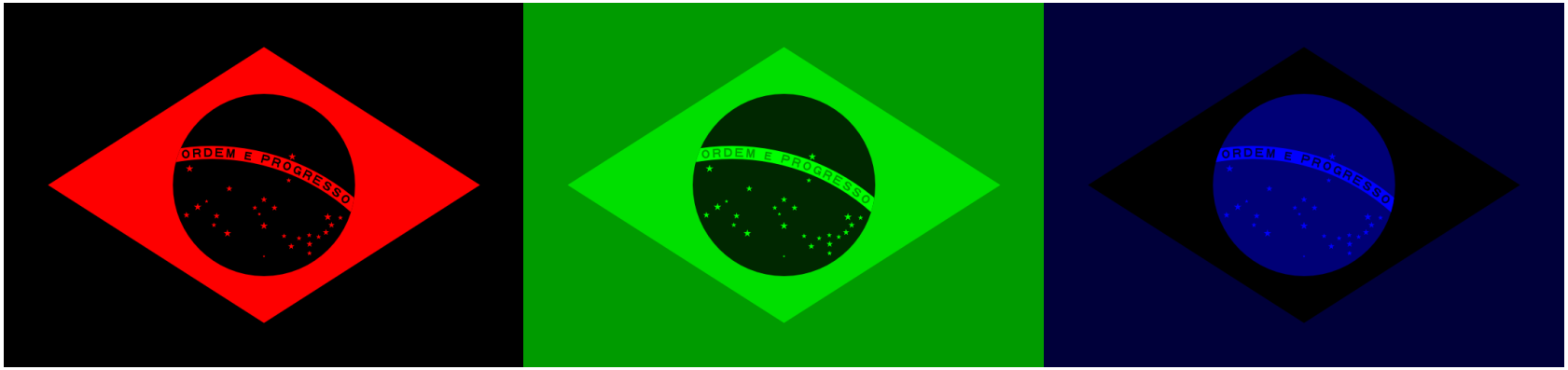




Wikipedia

- Note that CIE 1931 RGB is already normalized.

So, how to create a colored RGB image from a spectrum?
- Individual convolution of the RGB functions and them their combination.

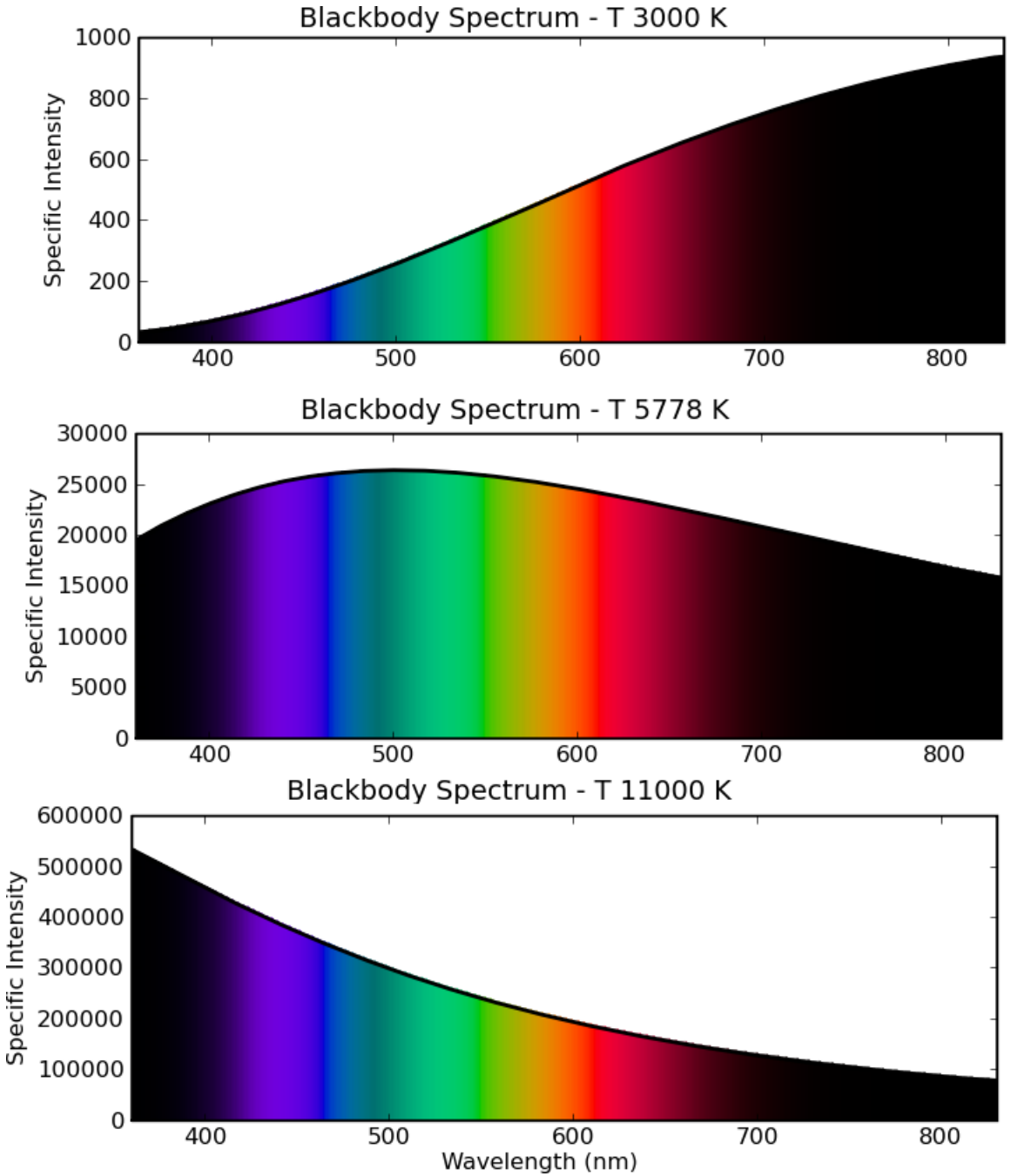


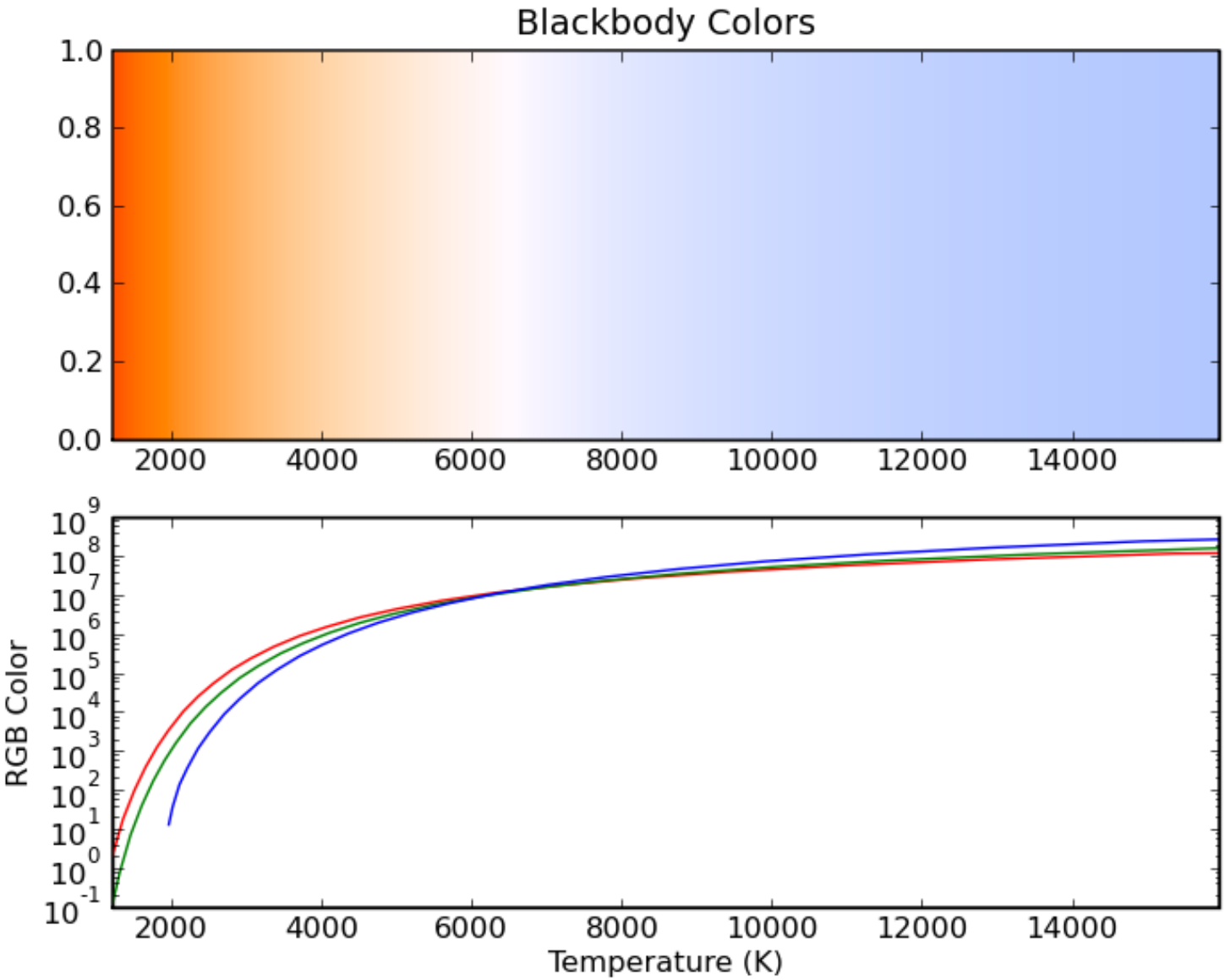
$$R = \int r(\lambda)\,\mathbf{I}(\lambda)\,d\lambda;\quad G = \int g(\lambda)\,\mathbf{I}(\lambda)\,d\lambda;\quad B = \int b(\lambda)\,\mathbf{I}(\lambda)\,d\lambda$$

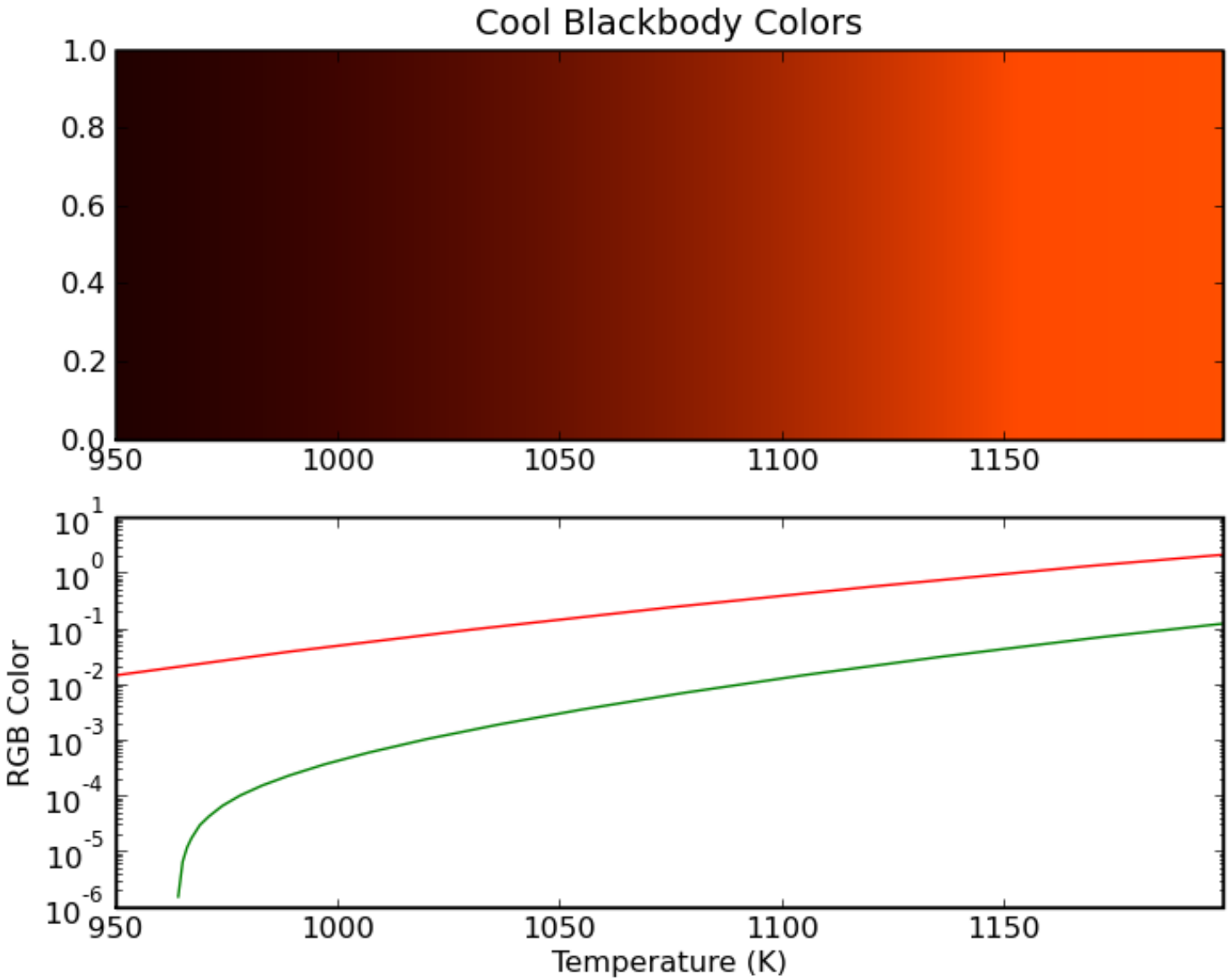
Brazil_flag.py

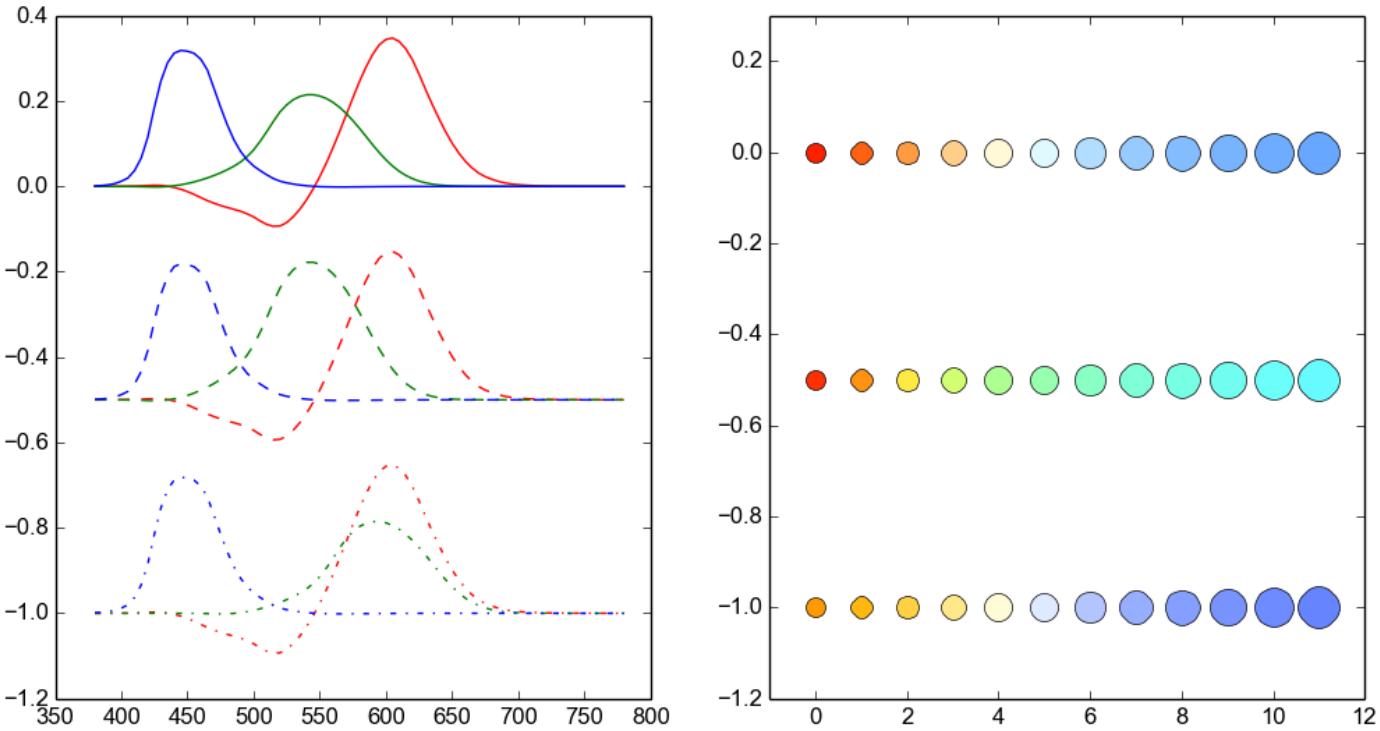
```
"""Export R, G and B layers of the Brazilian flag image"""

from PIL import Image

img = 'Flag_of_Brazil.png'

pilimg = Image.open(img)
rgbArray = np.asarray(pilimg)

colors = ['R','G','B']
for i in range(3):
    # zero all colors
    expimg = rgbArray[:,:,0:3]*0
    # copy just one
    expimg[:,:,i] = rgbArray[:,:,i]
    savimg = Image.fromarray(expimg)
    savimg.save('img_{}.png'.format(colors[i]))
```

Black Body intensities




Blackbody Colors
1.0
0.8
0.6
0.4
0.2
0.0
2000
4000
6000
8000
10000
12000
14000
RGB Color
Temperature (K)

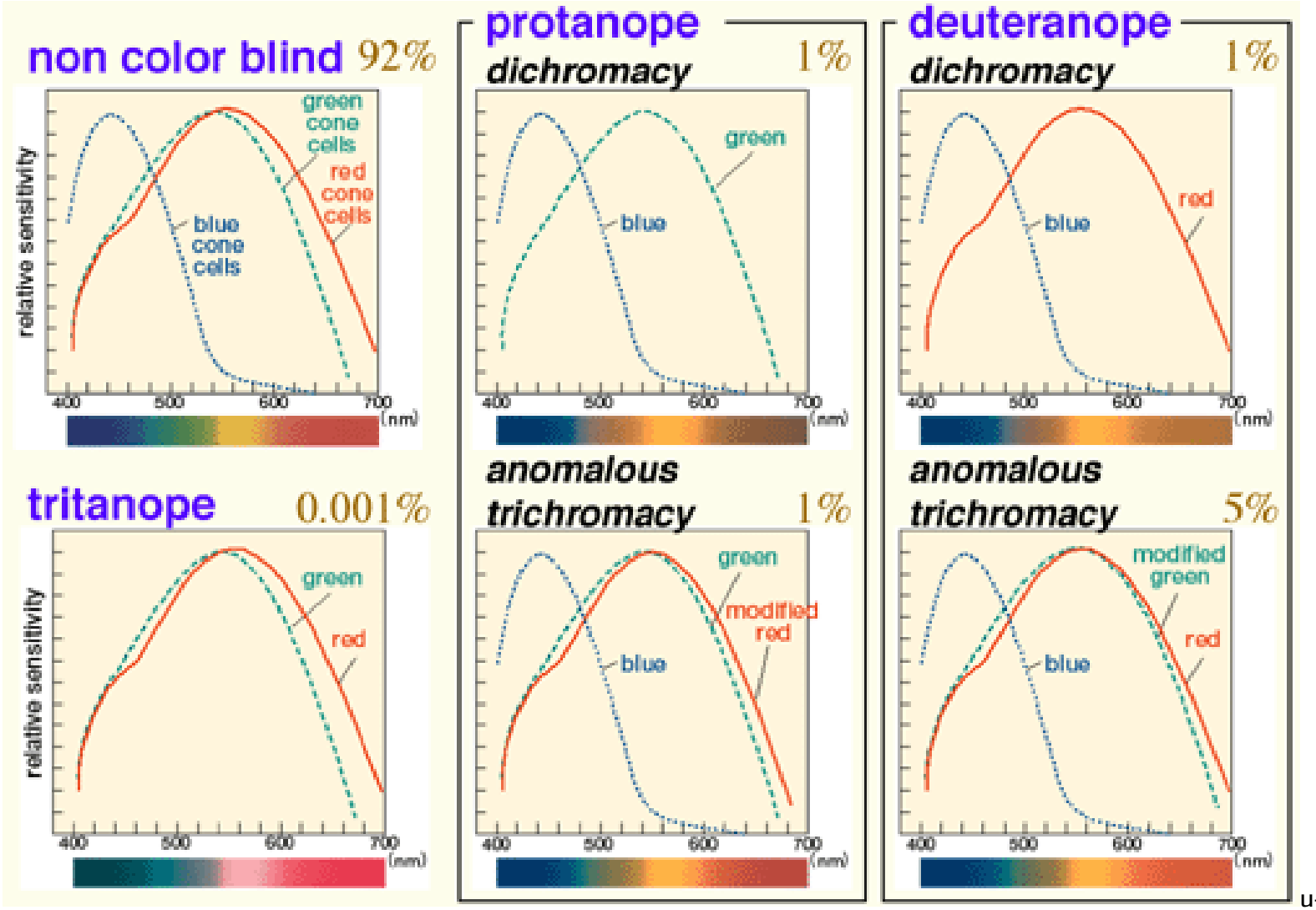

Cool Blackbody Colors
1.0
0.8
0.6
0.4
0.2
0.0
950
1000
1050
1100
1150
RGB Color
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
950
1000
1050
1100
1150
Temperature (K)



Left: three different color sensation responses;
Right: the resulted black body colors, from 1000 to 12,000 K.
Top to bottom: i) normal human vision; ii) artificially decreased red response; iii) and most common human colorblindness.

BB_RGB.PY

```
"""Convolve the CIE 1931 RGB to the Black Body function."""

lbd = np.linspace(370, 730, 101) #nm
Temps = np.arange(1000., 12000+1, 1000) #Kelvin

color = ['red','green','blue']
#CIE 1931 RGB table == Natural
fdat = np.loadtxt(hdt.hdtpath()+'/refs/rgb_eff.txt',
    unpack=True)

fig, (ax0, ax1)  = plt.subplots(1,2, figsize=(12,6))
for i in range(3):
    ax0.plot(fdat[0],    fdat[i+1]-0.0,    color=color[i])

for i in range(len(Temps)):
    BB = phc.BBlbd(Temps[i], lbd*1e-7)
    cor = phim.doColorConv(lbd, BB, fdat)
    ax1.plot([i], [0.0], 'o', ms=10+1*i,
        color=mplcolors.rgb2hex(cor/255.))
```



u-tokyo

More than **6%** of males have anomalous trichromacy.

Tips making figures:

Do not convey information with color only. Show differences BOTH in color and in shape.

"redundant coding"

In addition to color, use the combinations of :

- solid and various dotted lines
- various hatching
- circles, triangles and rectangles
- alphabets and numbers
- etc.

Keep the number of colors to a minimum.

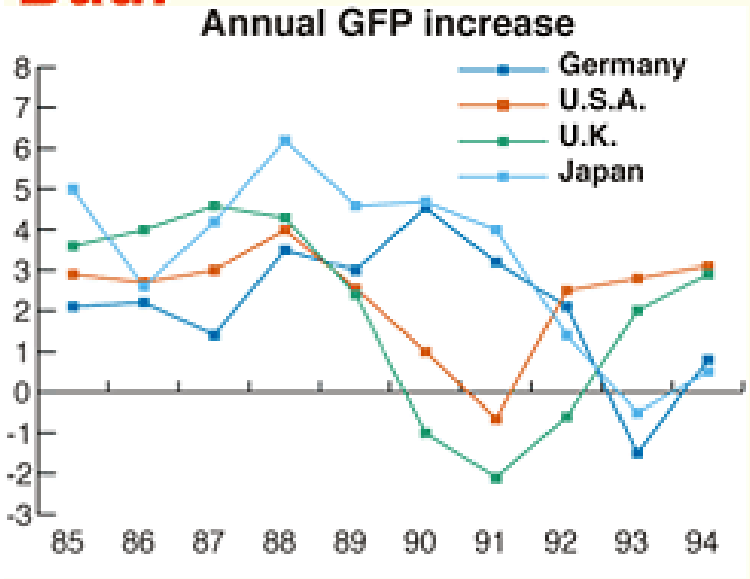
Use combinations of *different symbols with a few, vivid colors* rather than a single symbol with various colors.

Keep contrast not only in hue but also in *brightness.*

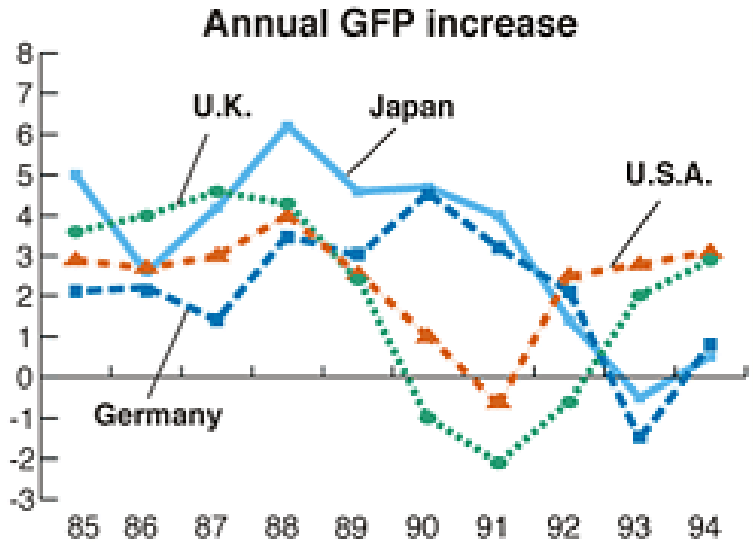
Make it possible to communicate *without using color name.*

u-tokyo

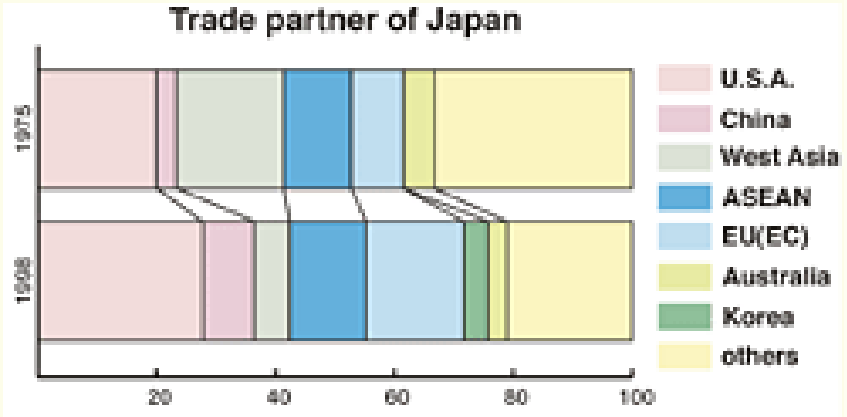
Bad!

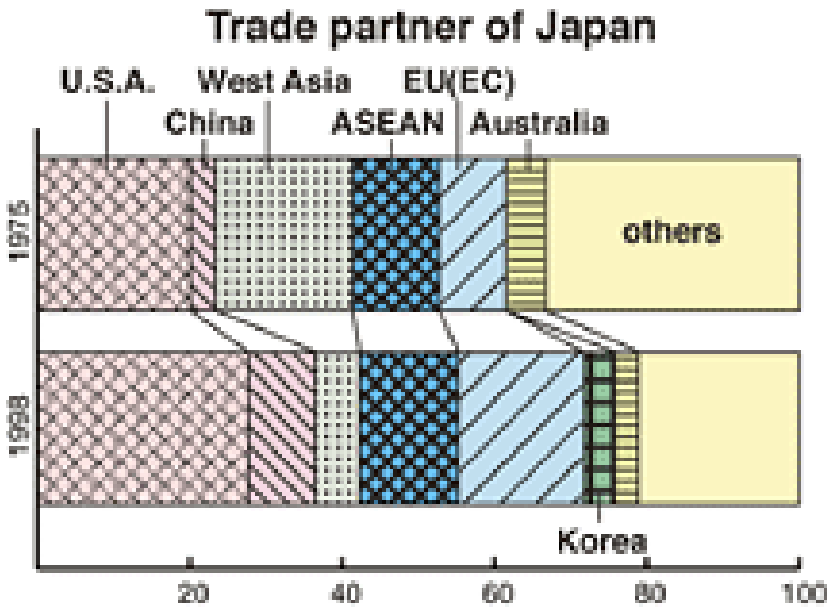


Good!



Bad!

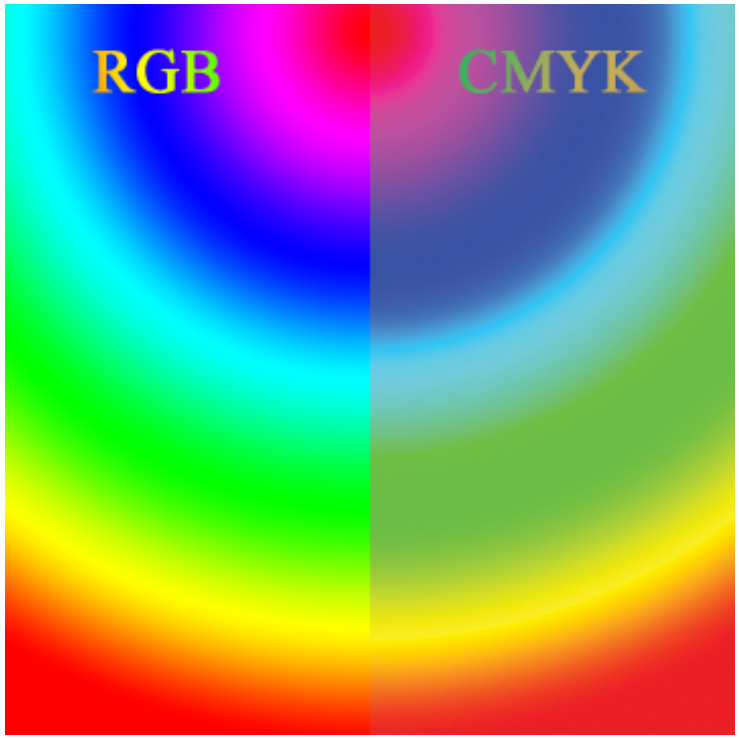




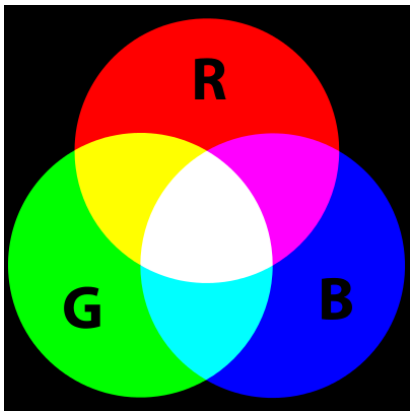
u-tokyo

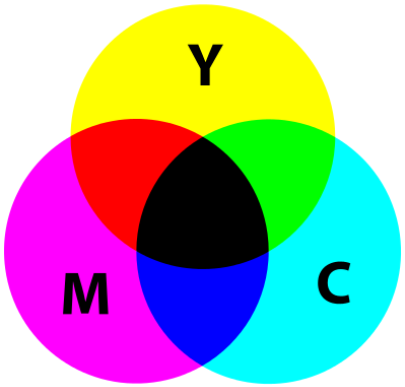
Tips making figures:

- Use colormaps with yellow, brown, blue and red.
- Make it compatible with grayscale!
- See how it will appear when printed comparing with the CMYK color space.



Wikipedia





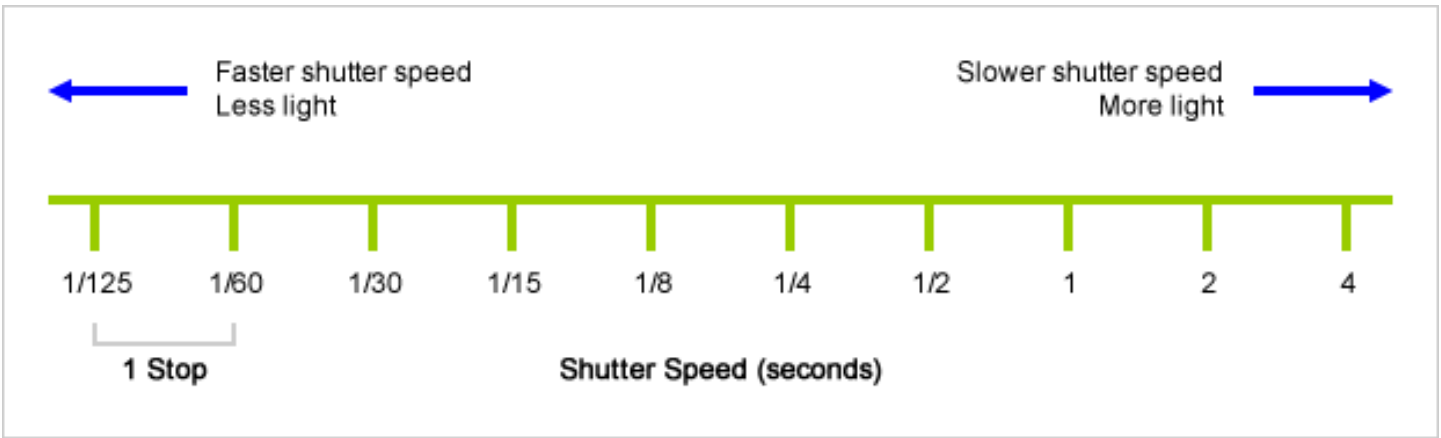
Wikipedia

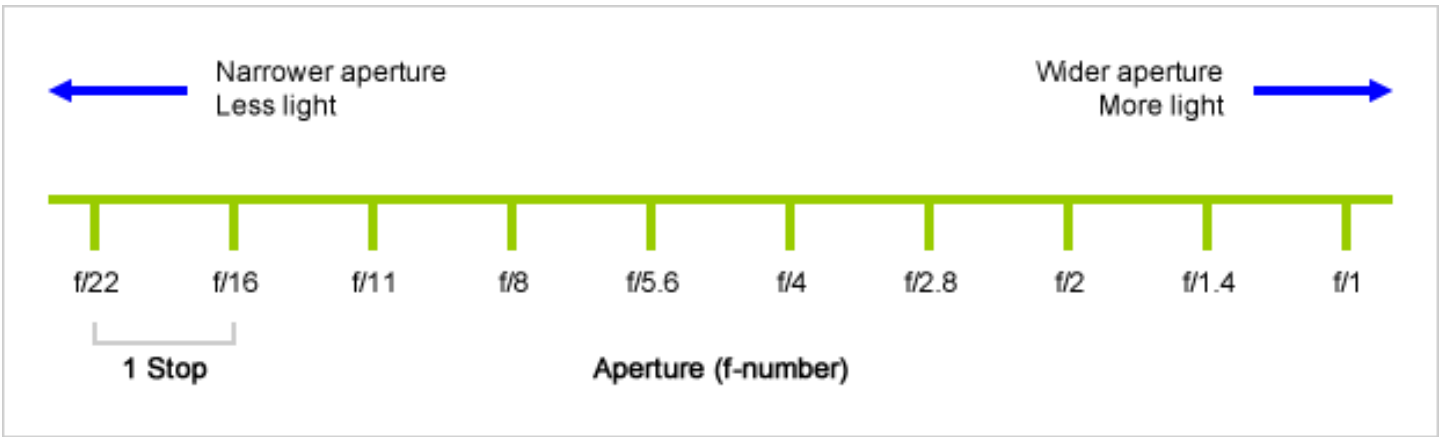
RGB2CMYK.py

```
"""Convert a RGB image to CMYK color space."""

# RGB to CMYK
img = 'Flag_of_Brazil.png'
img = PIL.Image.open(img)
img = img.convert(mode='RGB')
img = phim.rgb2cmyk(np.asarray(img))
img = PIL.Image.fromarray(img, mode='CMYK')
img.save('FlagCYMK.jpg')

# CMYK to RGB: PIL way
img = hdt.hdtpath()+'/refs/repo/CMYK2rgb.jpg'
img = PIL.Image.open(img)
img.save('cmyk.jpg')
img = img.convert(mode='RGB')
img.save('rgb.jpg')
```

- In photography, the brightness (mean) level is set by the exposure.
- Exposure is controlled by shutter speed, aperture, and ISO speed.
- Stops let you directly compare and swap between these.

photographymad





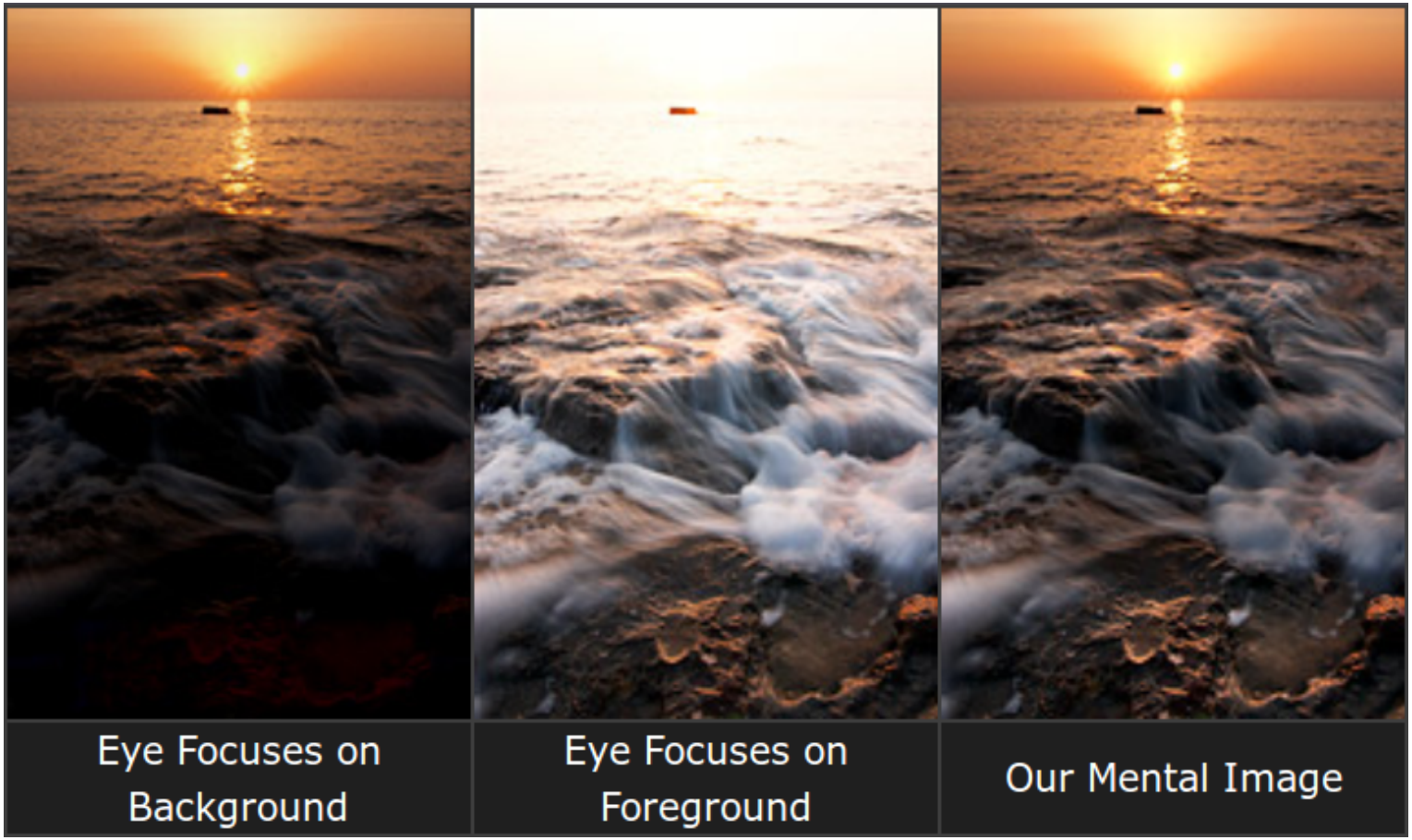
Dynamic range is the brightness difference between dark and bright areas.

Table : Dynamic ranges of common devices

Device	**Stops**	**Contrast**
LCD	8-11	250:1 - 1750:1
CCD Cameras	10-14*	1000:1 - 15000:1
Photographic films	12*	4000:1
Human eye	10-14 (24)	1000:1 - 15000:1 (16000000:1)

- Human eyes numbers: fixed pupil aperture vs. all illumination conditions.
- 16000000:1 only make sense in a log sensitivity.

Human vision constructs images with different intensity levels through dynamic adjustment of the pupil (like a video camera!).



- Now to mimic this process: HDR (high-dynamic-range) imaging!
- Basically, HDR is the combination of images taken under different stop values (brightness levels).
- In this process, underexposed ("zero counts") and overexposed ("saturated") pixels are ignored.

-4 stops -2 stops +2 stops +4 stops

Wikipedia

-4 stops

-2 stops

+2 stops

+4 stops

Result after processing: the HDR image.

Wikipedia

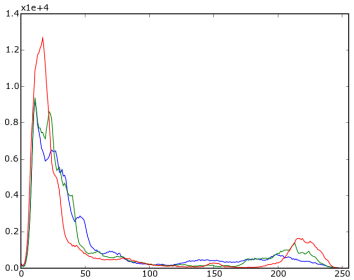
Doing it with Python: “bpowah” script.
https://sites.google.com/site/bpowah/hdrandpythonpil

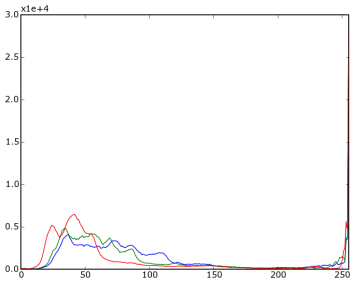
Example HDR.PY:

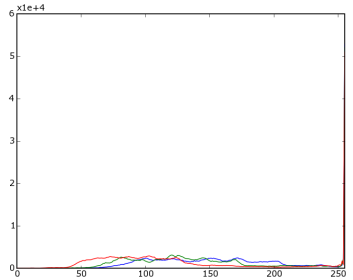
- Put all (stops) images inside a folder (ORIG).

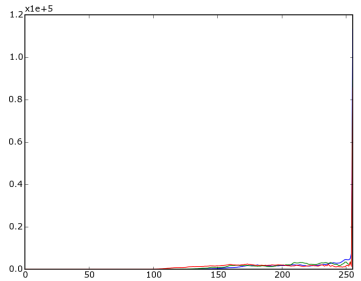
```
from pyhdust.hdrpil import hdr

proj = hdr(case='orig', resize=1.0, img\_type='jpg')
proj.get\_hdr(strength=[0.,1.,2.], naturalness=[0.8,0.9,1.0])
```

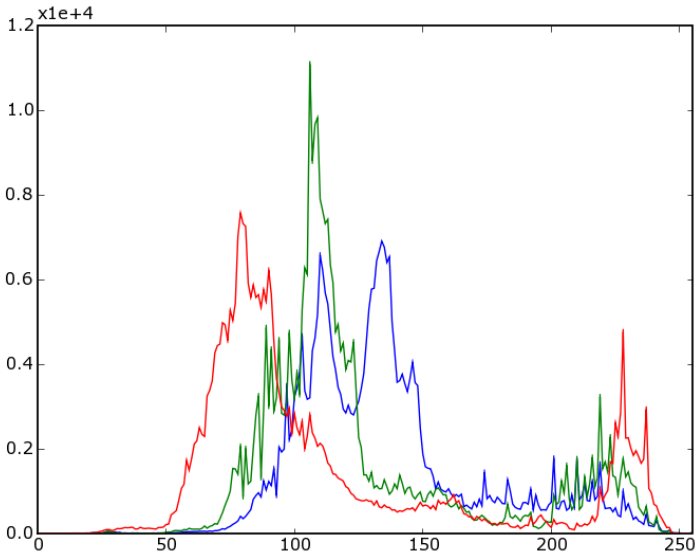
- Output in the OUT subfolder.


x1e+4
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250

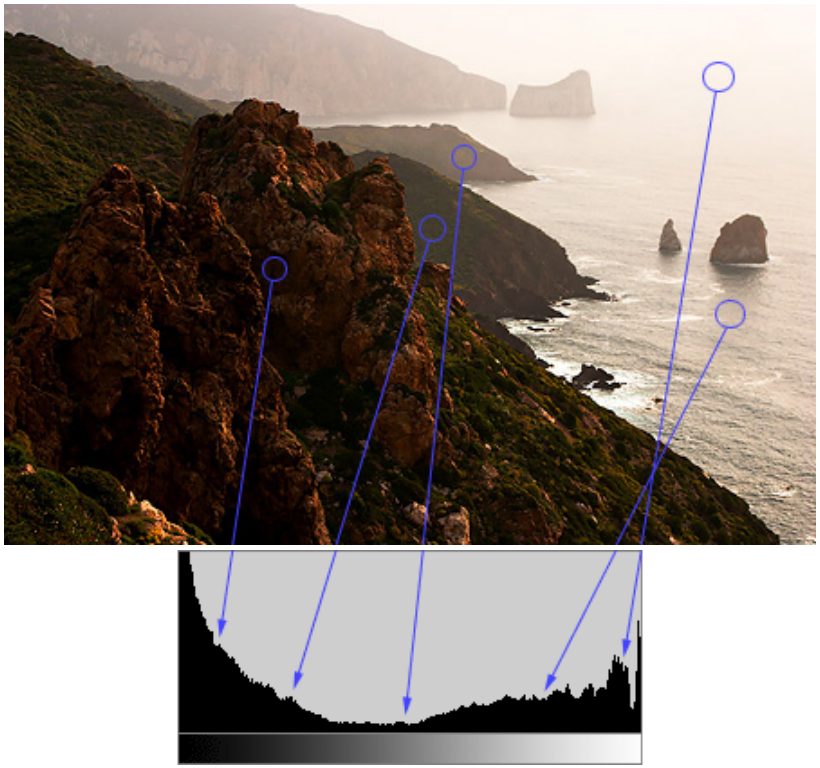

x1e+4
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
50
100
150
200
250

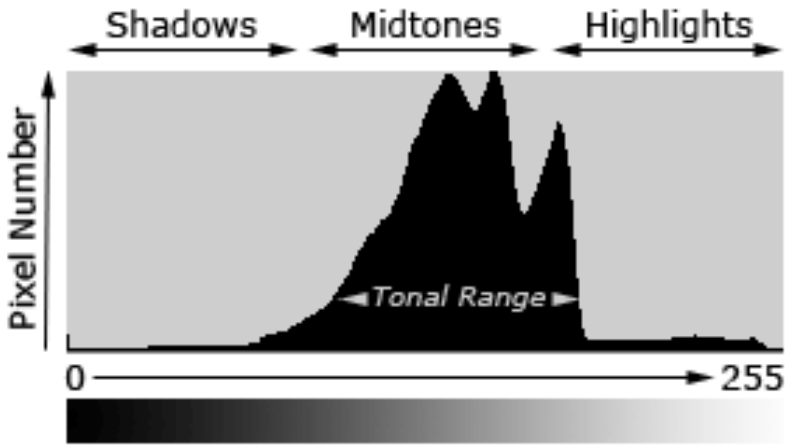

x1e+4
6
5
4
3
2
1
0
0
50
100
150
200
250

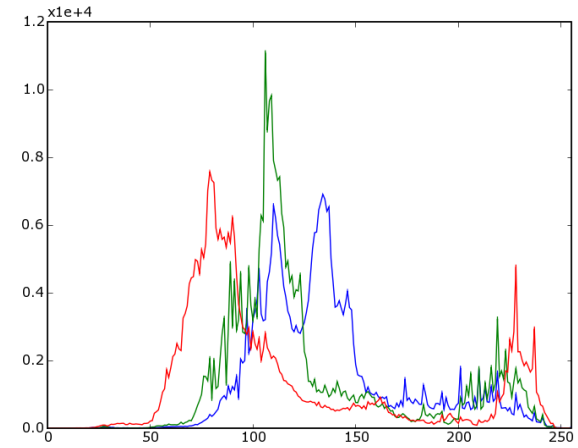

x1e+5
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250

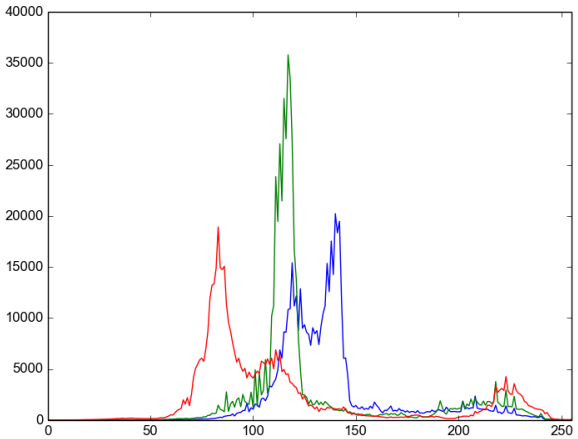

x1e+4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250

- "HDR problem": it changes the image tones ("absolute colors").
- Tonal range is the region where most of the brightness values are present.









- The different re-scaled levels can be combined in different ways.
- In the "bpowah" script, these are the strength and naturalness parameters.

- Tone mapping has be used to enhanced contrast and colors:

Wikipedia

(HDR images with and without tone mapping.)

- Very useful to enhance color response of different brightness levels.

References:

- Wikipedia
  `http://en.wikipedia.org`
- Cambrigde in Colour
  `http://www.cambridgeincolour.com`
- “bpowah” script
  `https://sites.google.com/site/bpowah/hdrandpythonpil`
- “How to make figures and presentations that are friendly to Colorblind people”
  `http://jfly.iam.u-tokyo.ac.jp/color/`
- “ColorPy” package
  `http://markkness.net/colorpy/ColorPy.html`

Topics not covered (but still interesting):

Table : Color vision table

Name of state	Cones	Colors perceived	Porters
Monochromacy	1	100	marine mammals
Dichromacy	2	10,000	most terrestrial non-primate mammals (colorblind)
Trichromacy	3	10 million	most primates
Tetrachromacy	4	100 million	most reptiles, rarely humans
Pentachromacy	5	10 billion	some insects

Topics not covered (but still interesting):

- Tone mapping.
- Sensitivity: the ability to resolve very faint or fast-moving subjects ("Newton's wheel problem").
- Color depth and RGB representations: Arithmetic, Digital 8-bit (numeric and hexadecimal) and 16-bit.
- RGB and RGBA: using transparency.

Topics not covered (but still interesting):

- Application of the colors (EM spec.):
  “Plants under Alien Suns”
  `http://www.solstation.com/life/a-plants.htm`