Data Visualization with matplotlib and seaborn

Bruno Gonçalves

www.data4sci.com

https://github.com/DataForScience/DataViz

Question

- Where are you located?
    - Europe
    - Asia
    - Africa
    - US
    - Canada
    - Latin America
    - Oceania

Question

- What's your job title?
    - Data Scientist
    - Data Engineer
    - Statistician
    - Researcher
    - Business Analyst
    - Software Engineer
    - Other

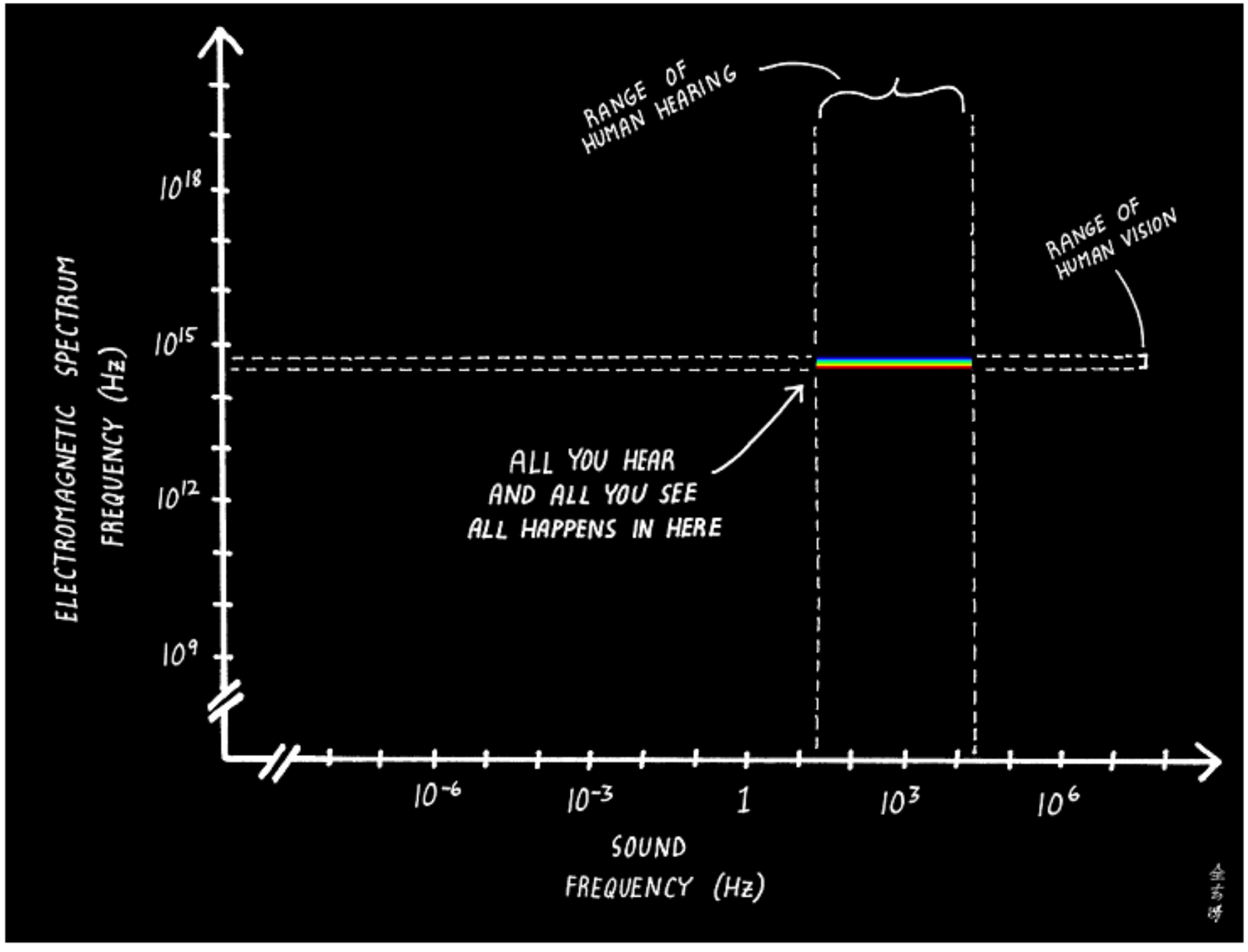
Question

- How experienced are you in Python?
    - Beginner (<1 year)
    - Intermediate (1-5 years)
    - Expert (5+ years)

Human Perception

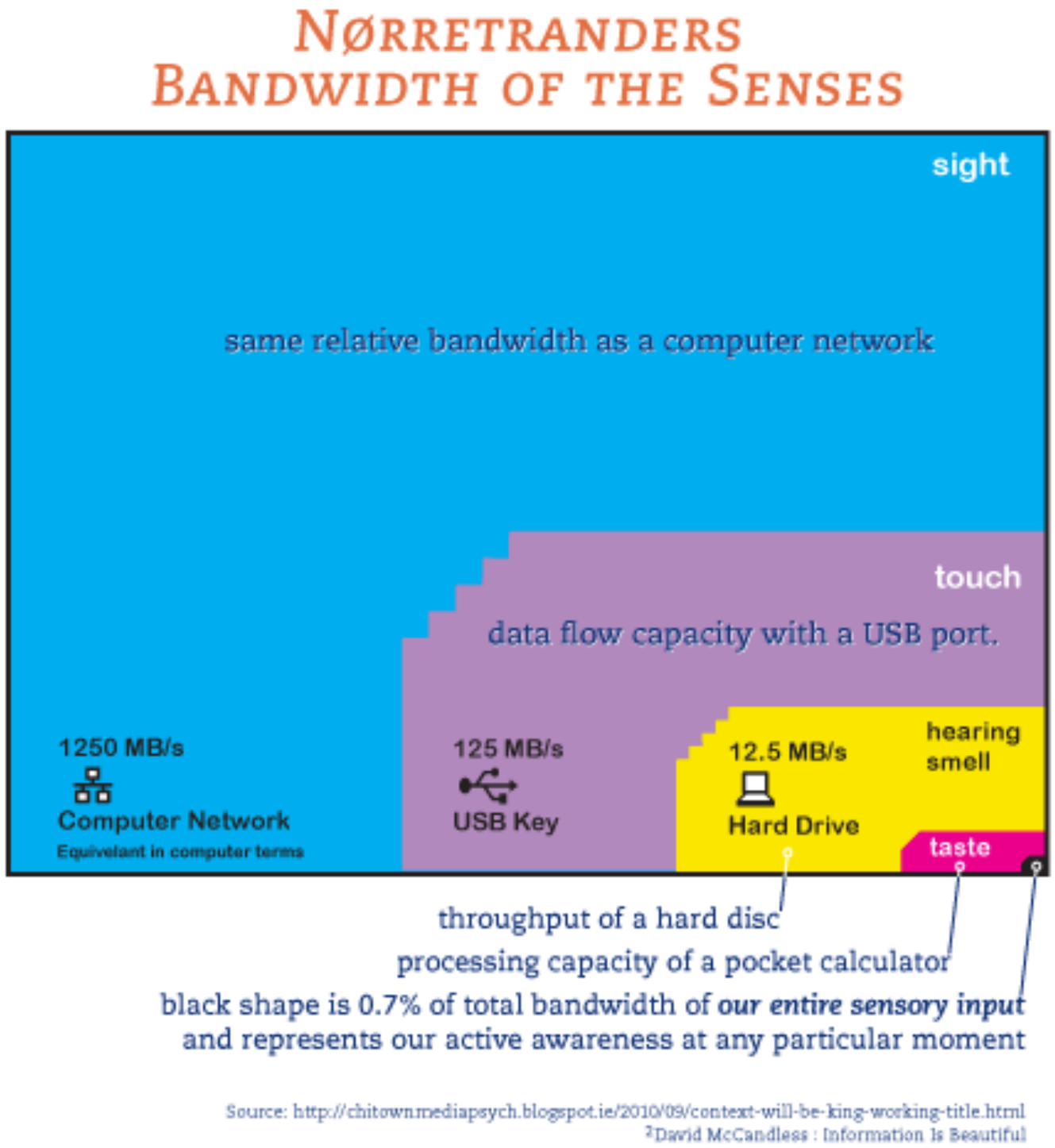
Human Perception

http://abstrusegoose.com/421



In the grand scheme of things, we're all pretty much blind and deaf.

Human Senses

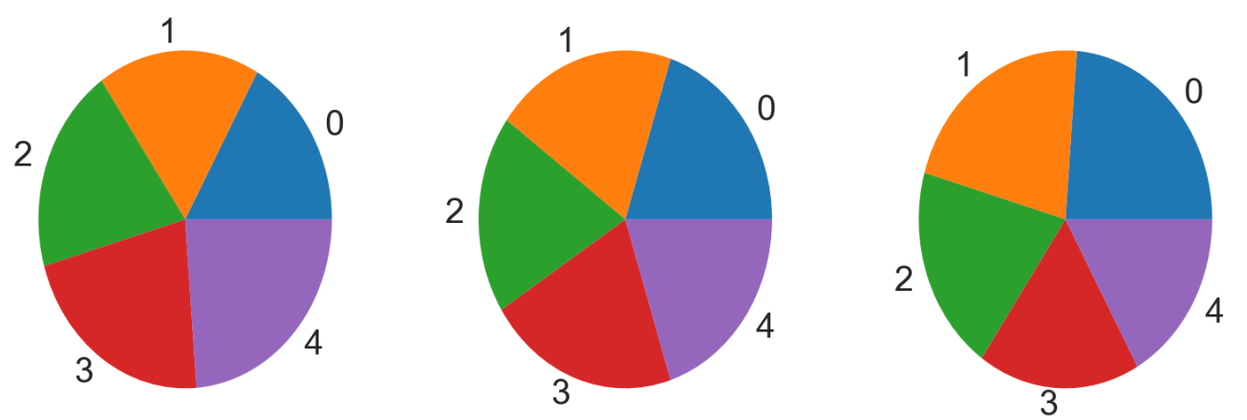


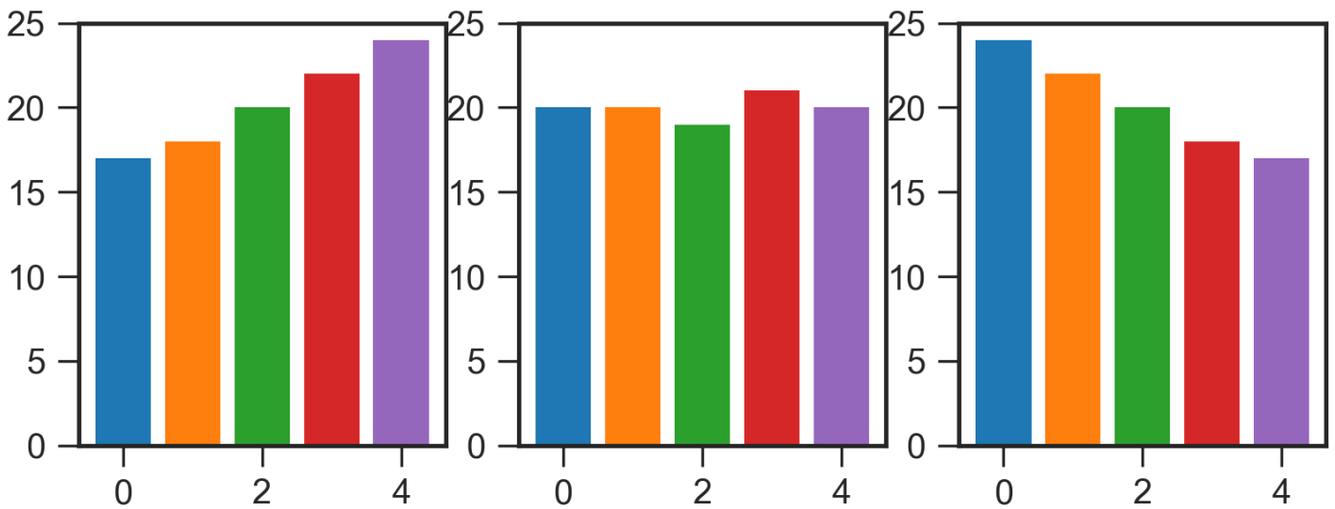
Perception

- Some cognitive tasks are significantly easier than others. In order, we are good a distinguishing:
    - Position, length
    - Direction, Angle, Area
    - Volume, Curvature, Shade
    - Color Saturation.

Perception

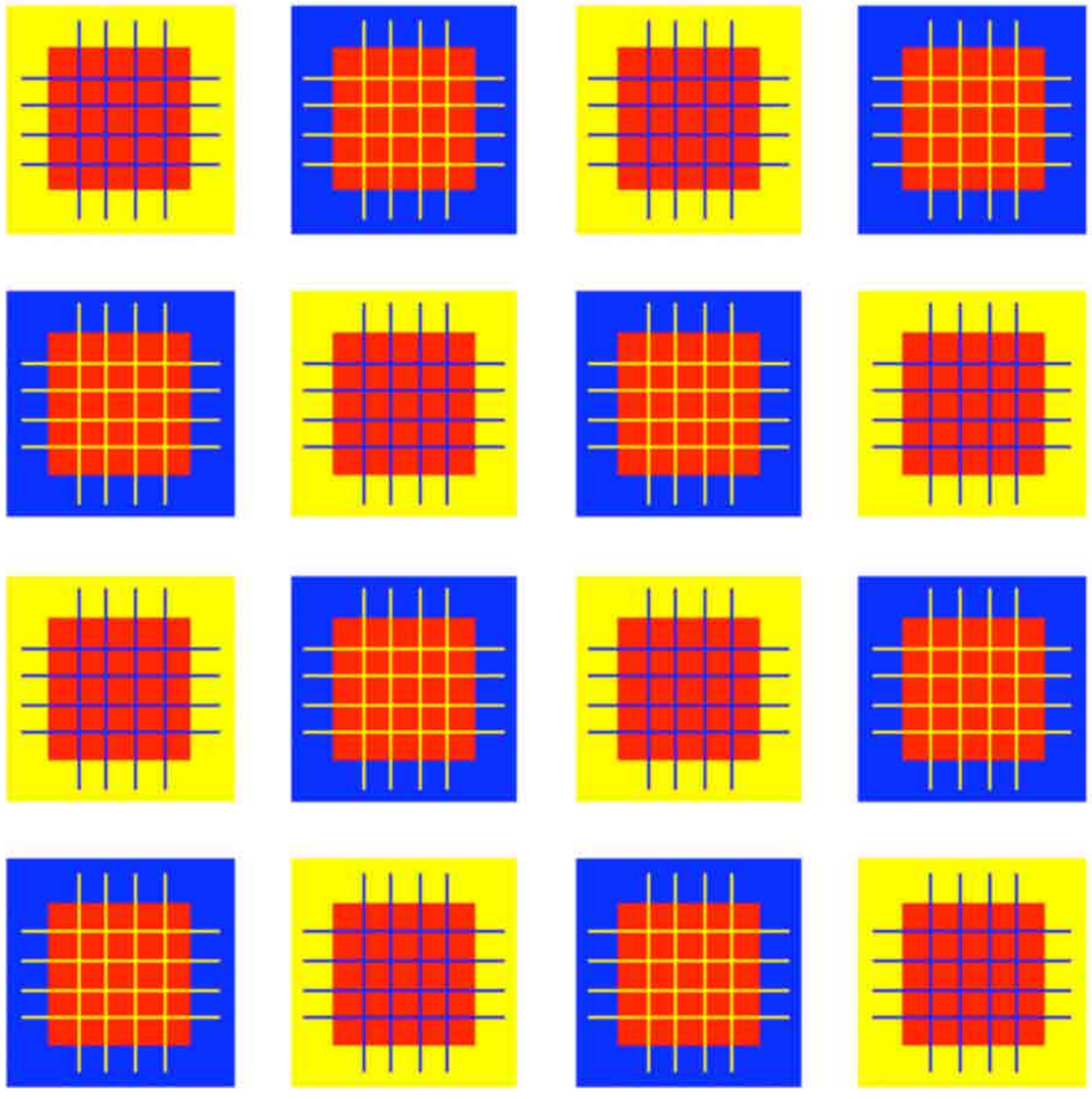
- Some cognitive tasks are significantly easier than others. In order, we are good a distinguishing:
  - Position, length
  - Direction, Angle, Area
  - Volume, Curvature, Shade
  - Color Saturation.





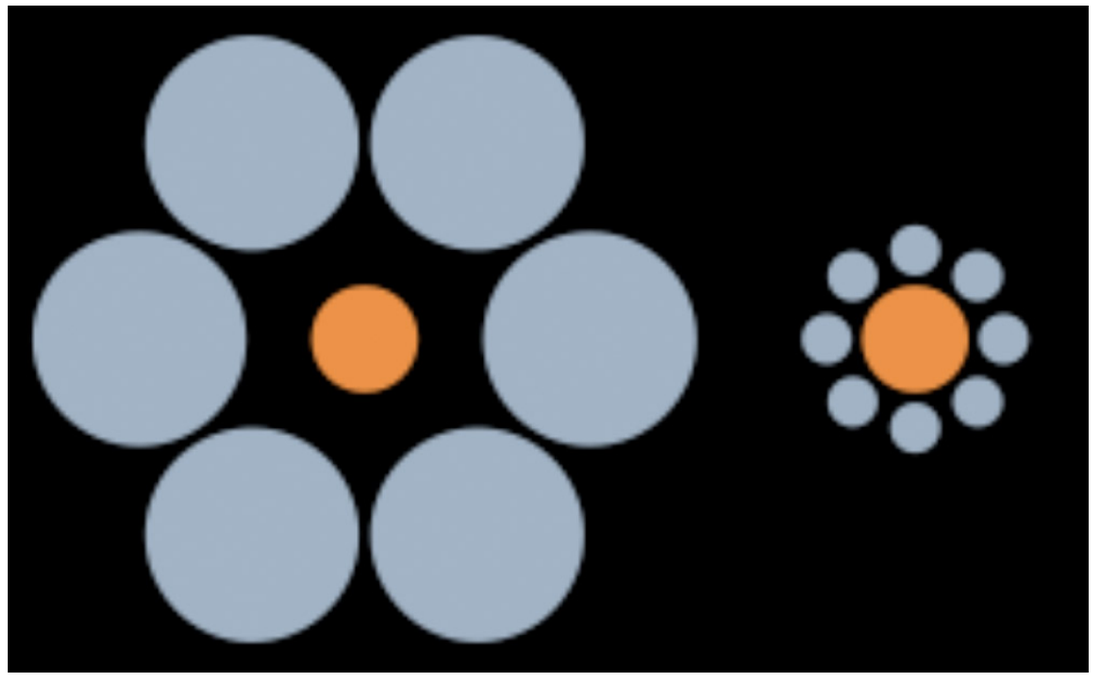
Perception

- Some cognitive tasks are significantly easier than others. In order, we are good a distinguishing:
  - Position, length
  - Direction, Angle, Area
  - Volume, Curvature, Shade
  - Color Saturation.

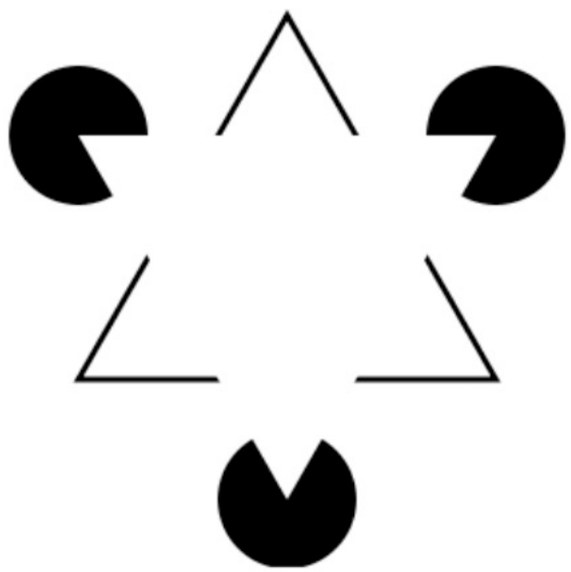


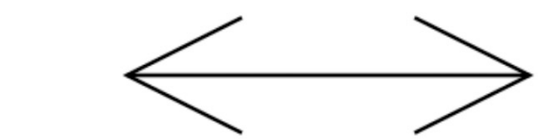
Perception

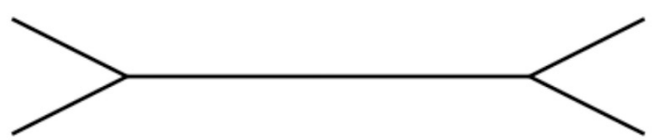
- Context also matters!
  - An object seen in the context of larger objects will appear smaller, while in the content of smaller objects it will appear larger.
  - And we “fill in the gaps”



- Some cognitive tasks are significantly easier than others. In order, we are good a distinguishing:
  - Position, length
  - Direction, Angle, Area
  - Volume, Curvature, Shade
  - Color Saturation.

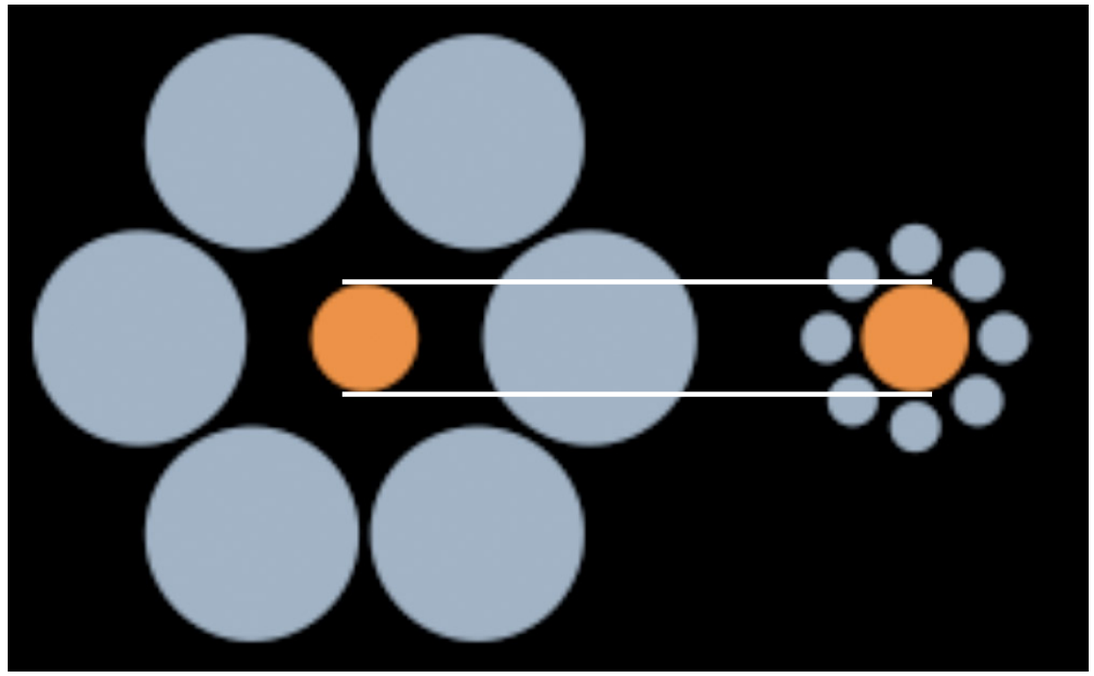




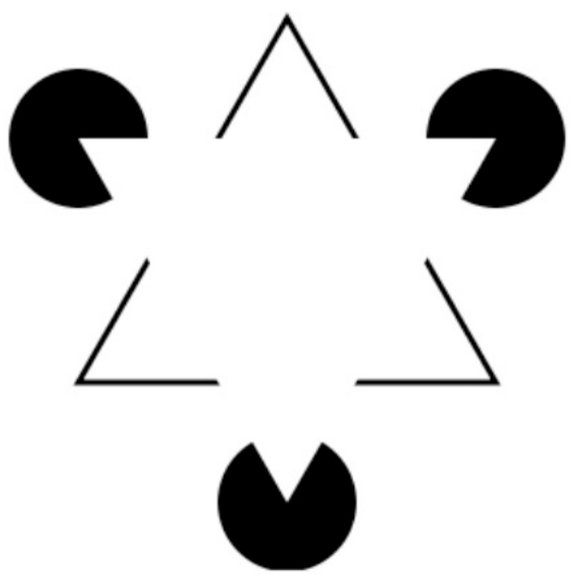


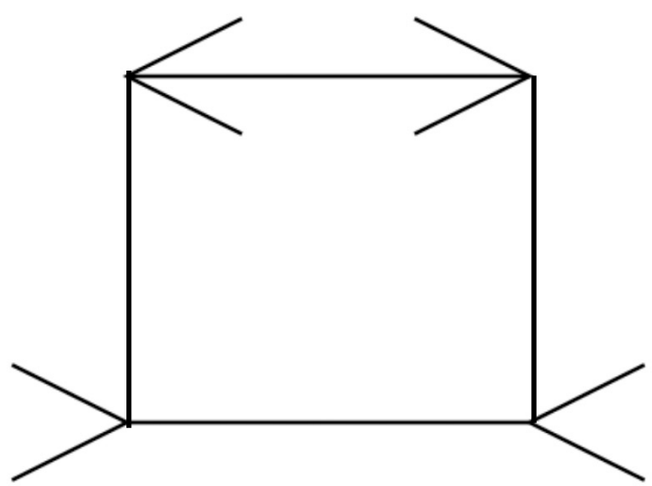
Perception

- Context also matters!
  - An object seen in the context of larger objects will appear smaller, while in the content of smaller objects it will appear larger.
  - And we “fill in the gaps”

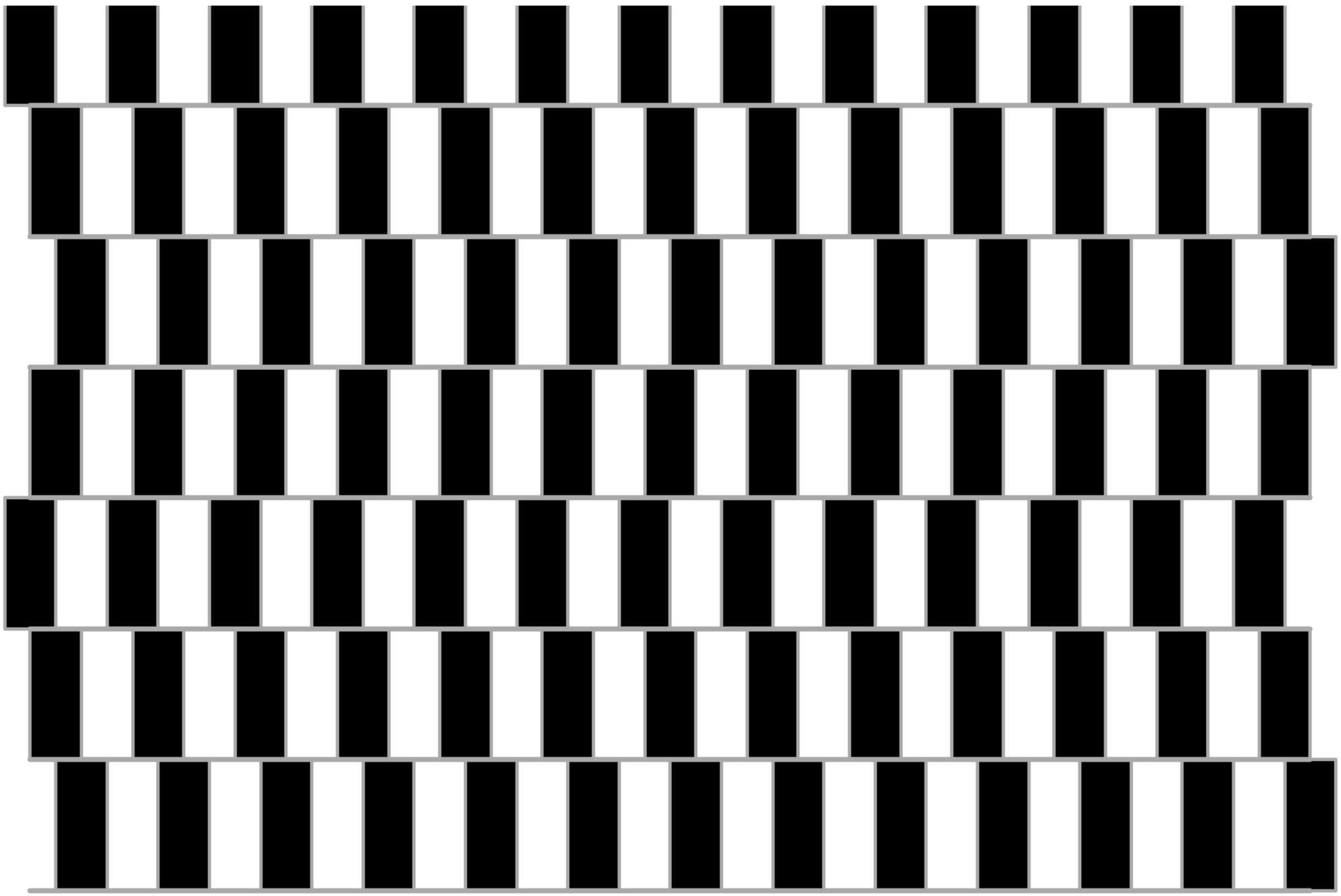


- Some cognitive tasks are significantly easier than others. In order, we are good a distinguishing:
  - Position, length
  - Direction, Angle, Area
  - Volume, Curvature, Shade
  - Color Saturation.



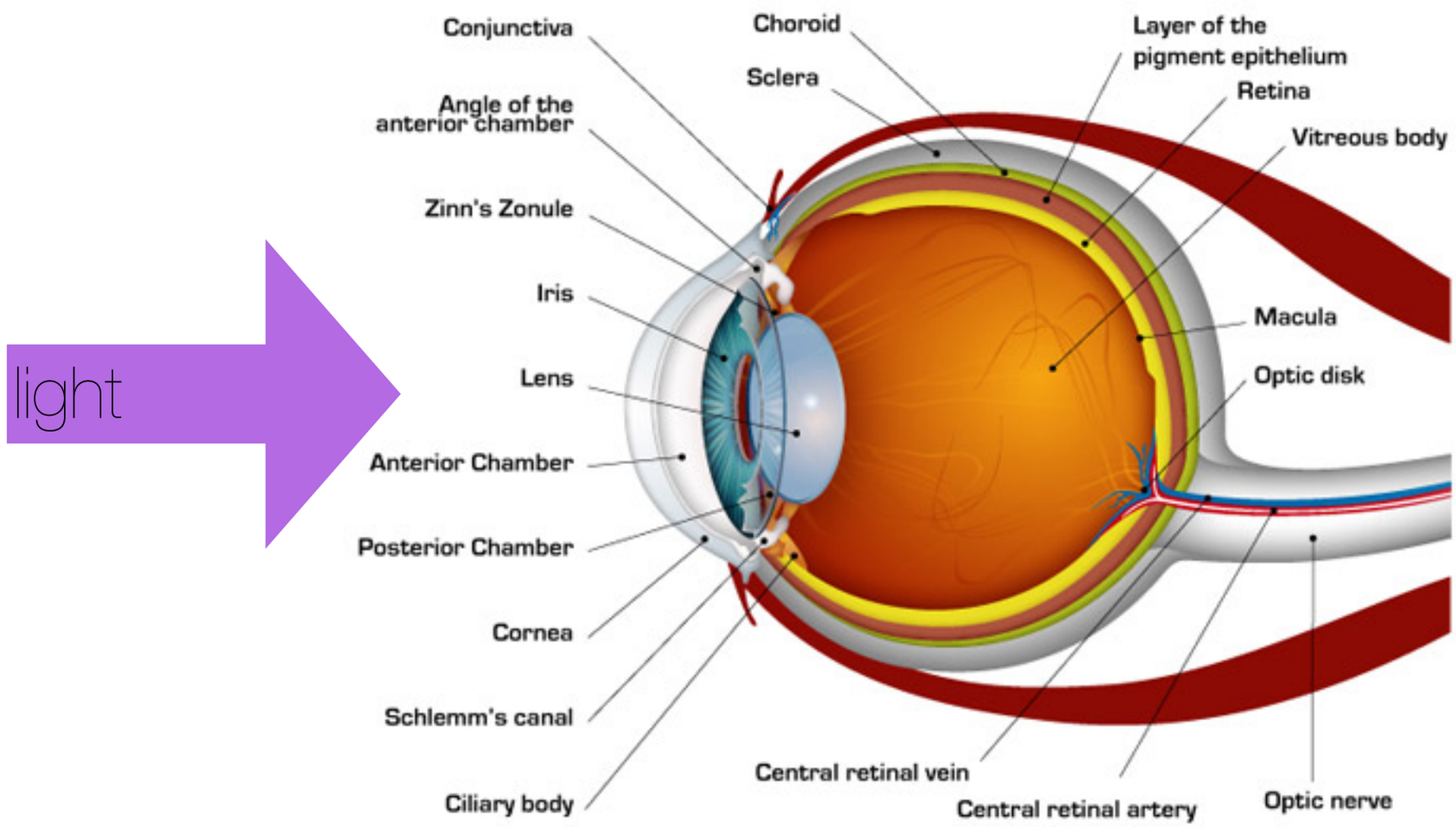


Perception Biases



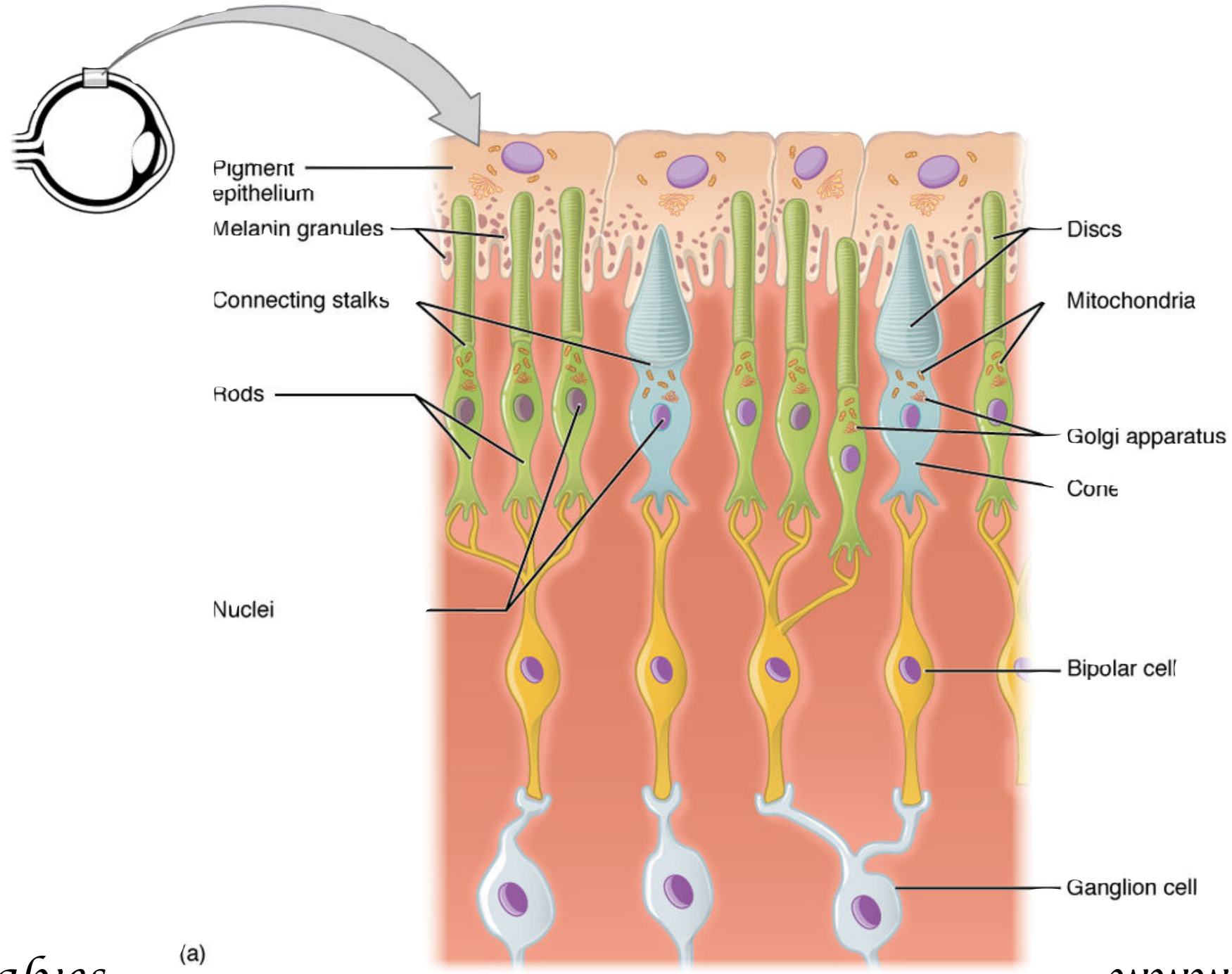
Human Vision

https://en.wikipedia.org/wiki/Photoreceptor_cell



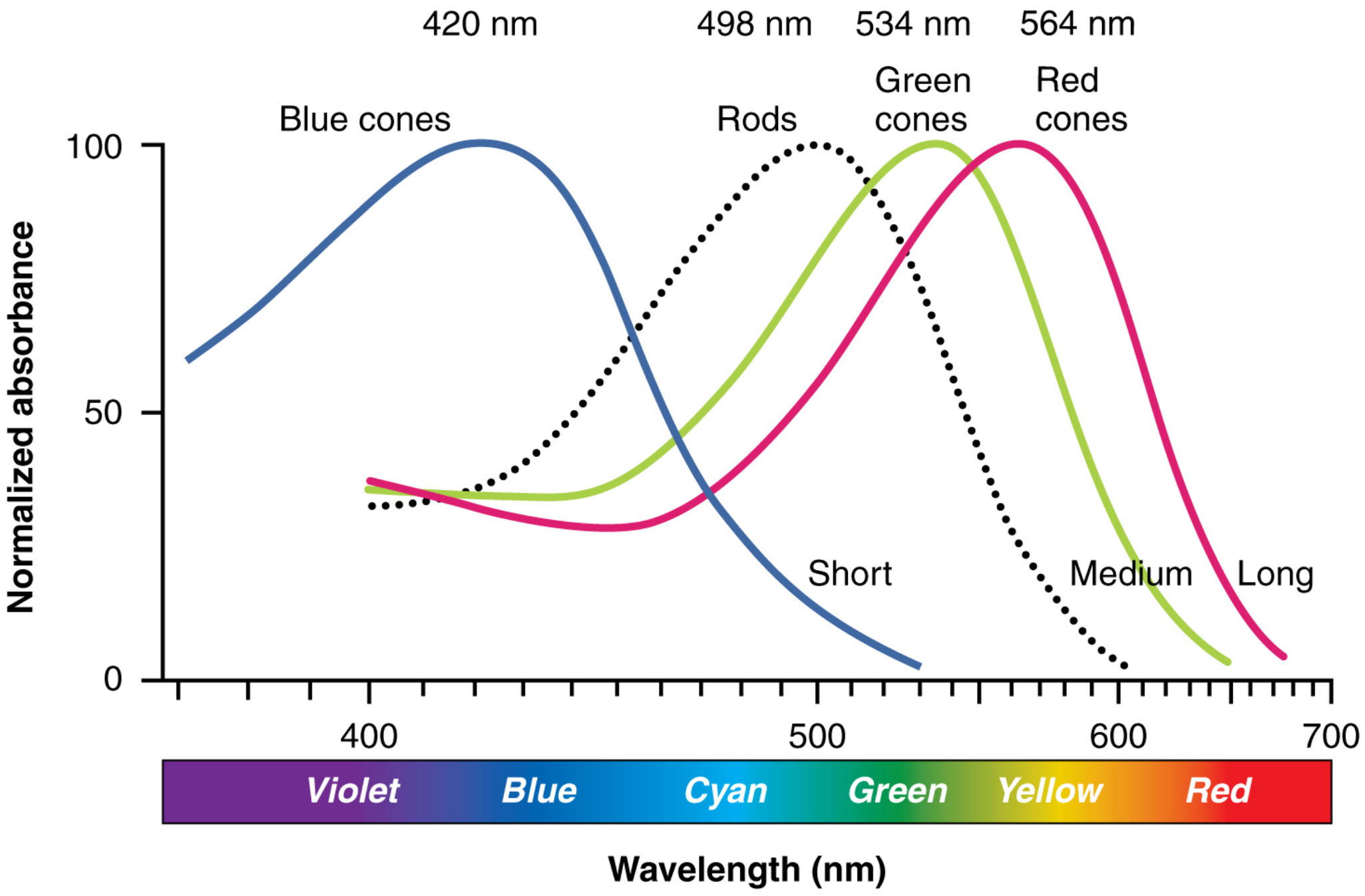
Human Vision

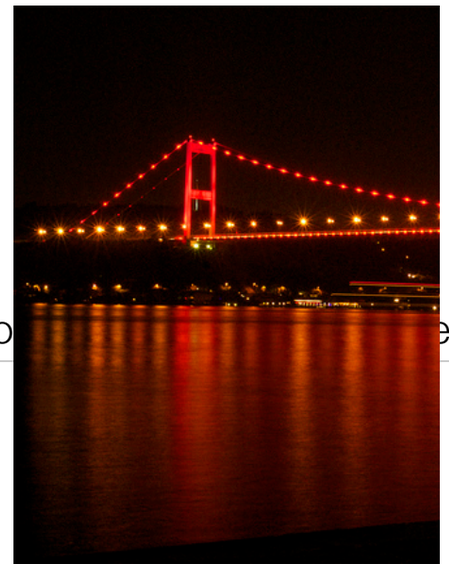
https://en.wikipedia.org/wiki/Photoreceptor_cell



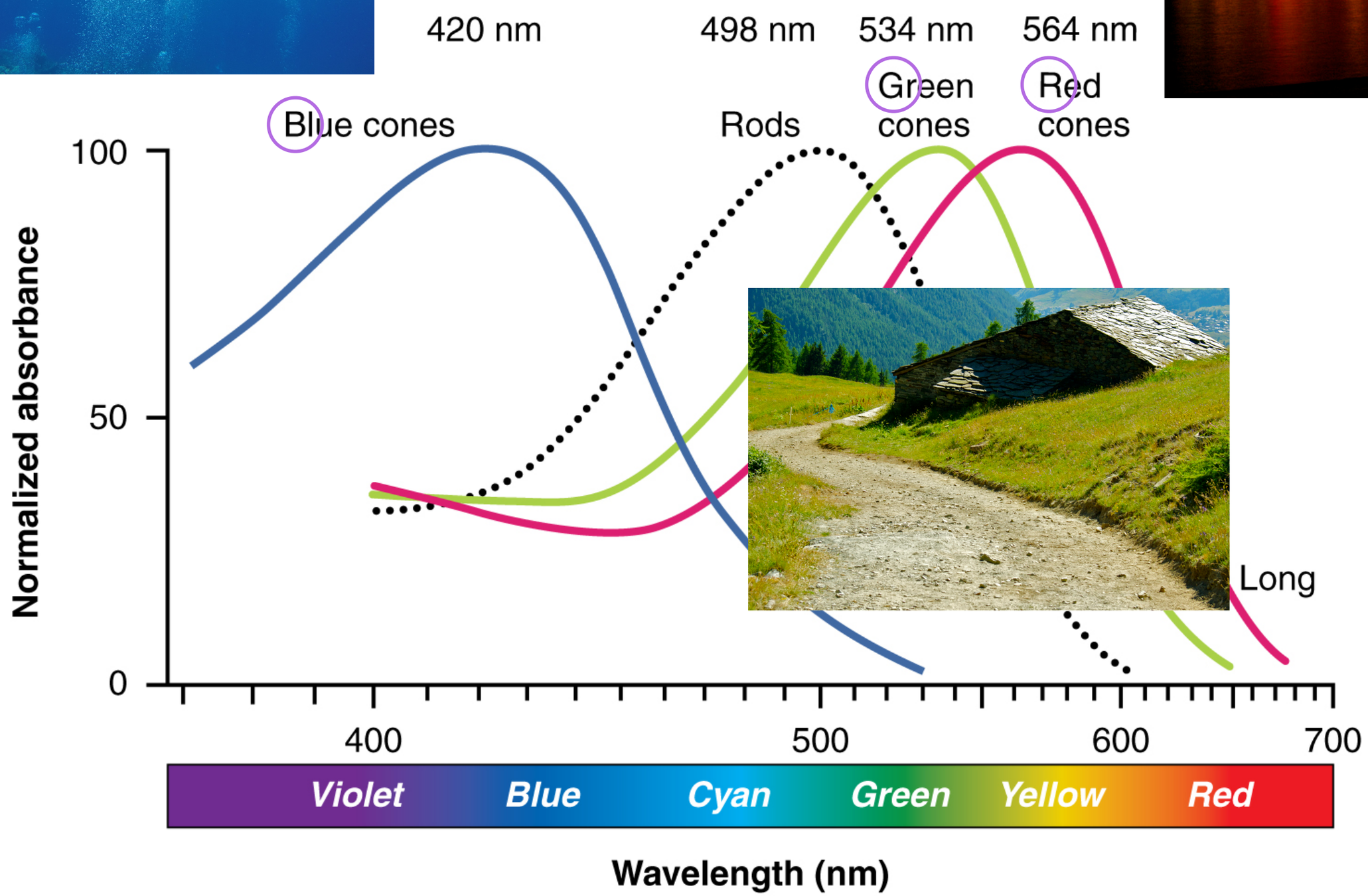
Human Vision

https://en.wikipedia.org/wiki/Photoreceptor_cell

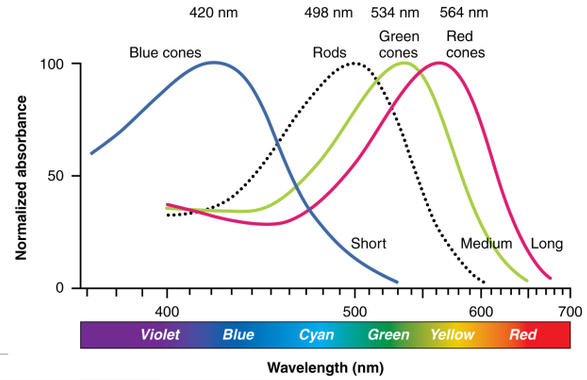




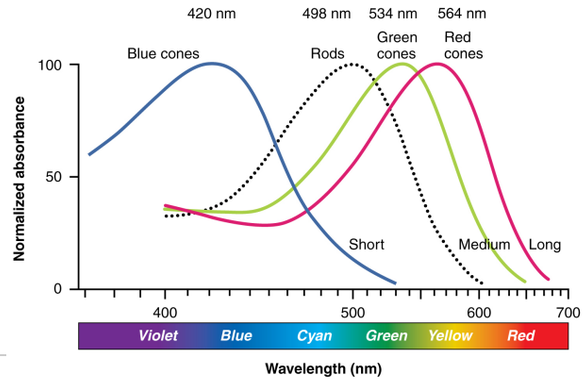
https://en.wikipedia.o ell

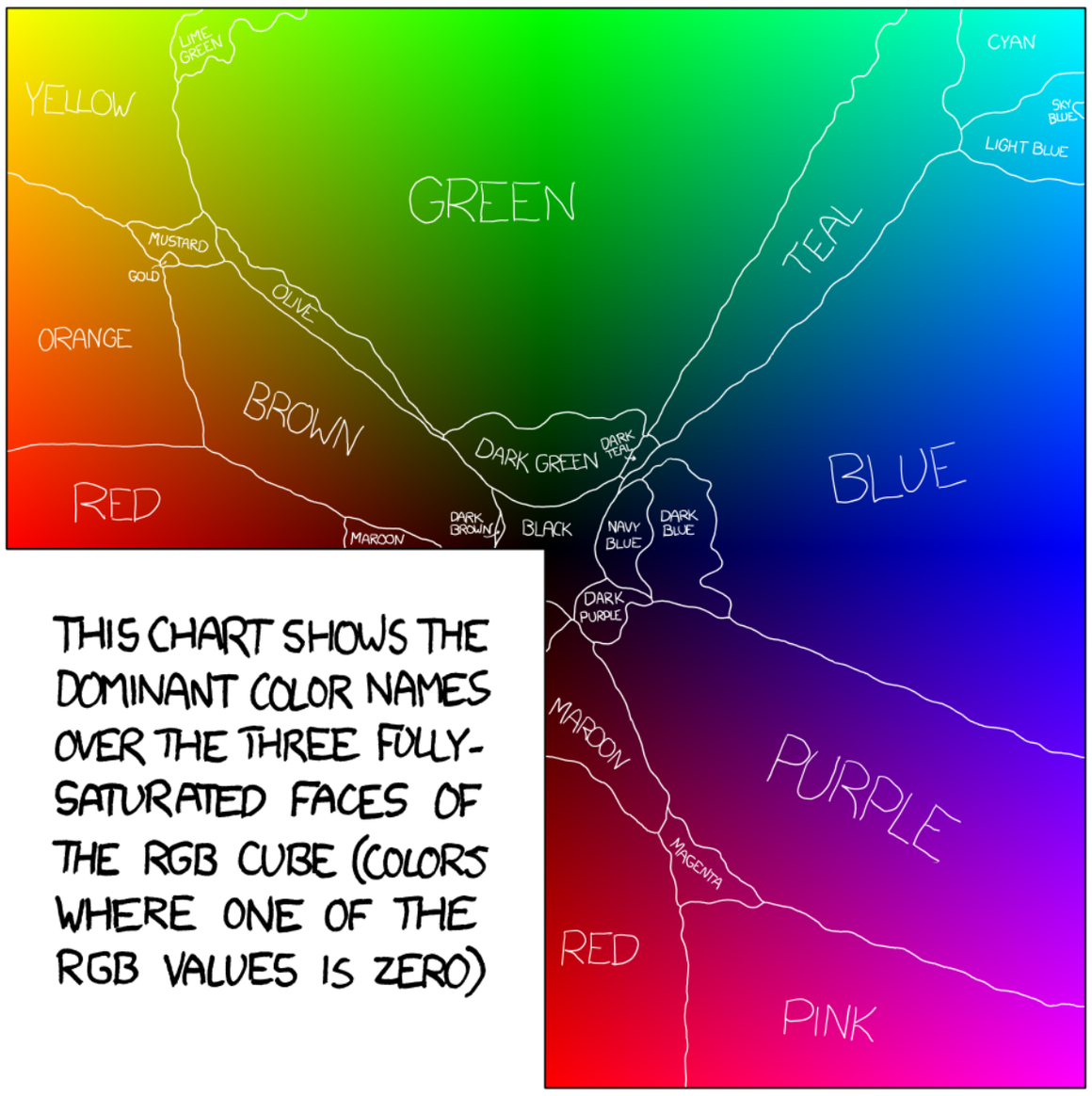


Colors galore!



Color Perception

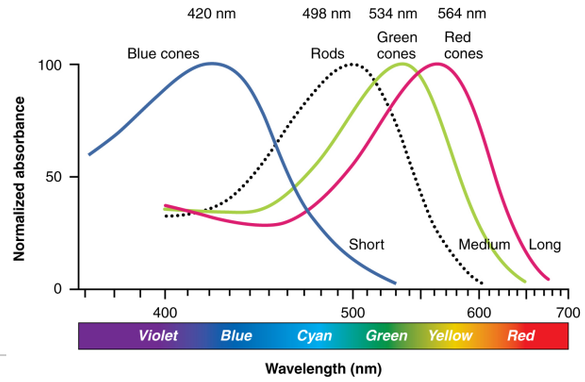


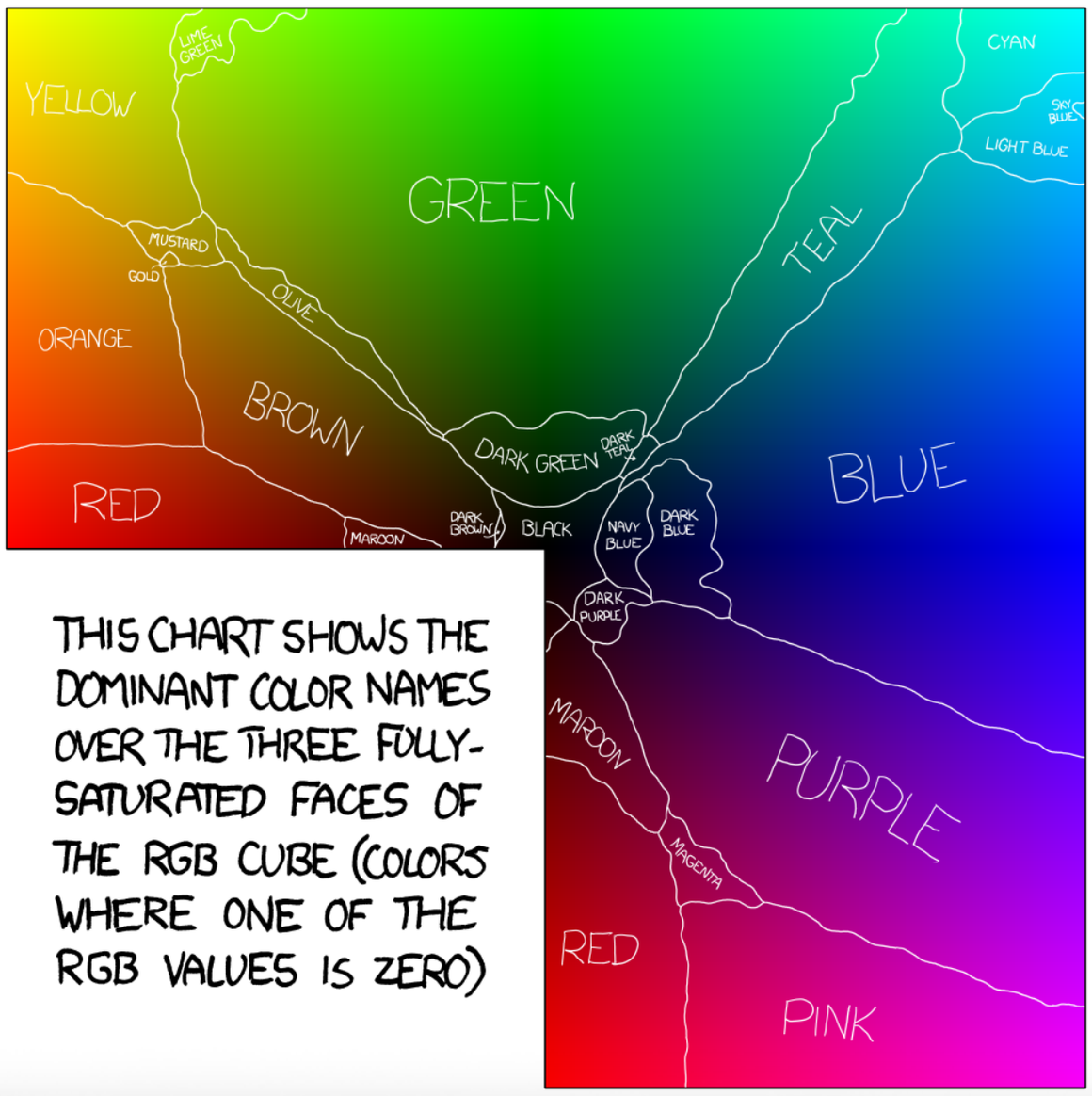


THIS CHART SHOWS THE DOMINANT COLOR NAMES OVER THE THREE FULLY-SATURATED FACES OF THE RGB CUBE (COLORS WHERE ONE OF THE RGB VALUES IS ZERO)

@bgoncalves

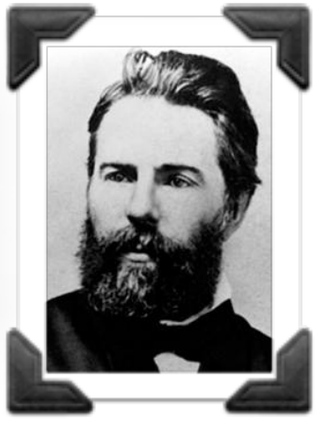
Color Perception





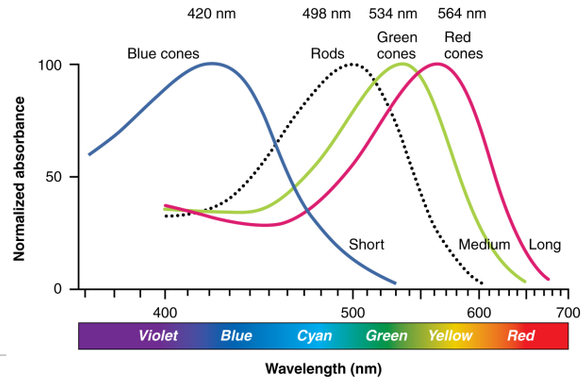
THIS CHART SHOWS THE DOMINANT COLOR NAMES OVER THE THREE FULLY-SATURATED FACES OF THE RGB CUBE (COLORS WHERE ONE OF THE RGB VALUES IS ZERO)

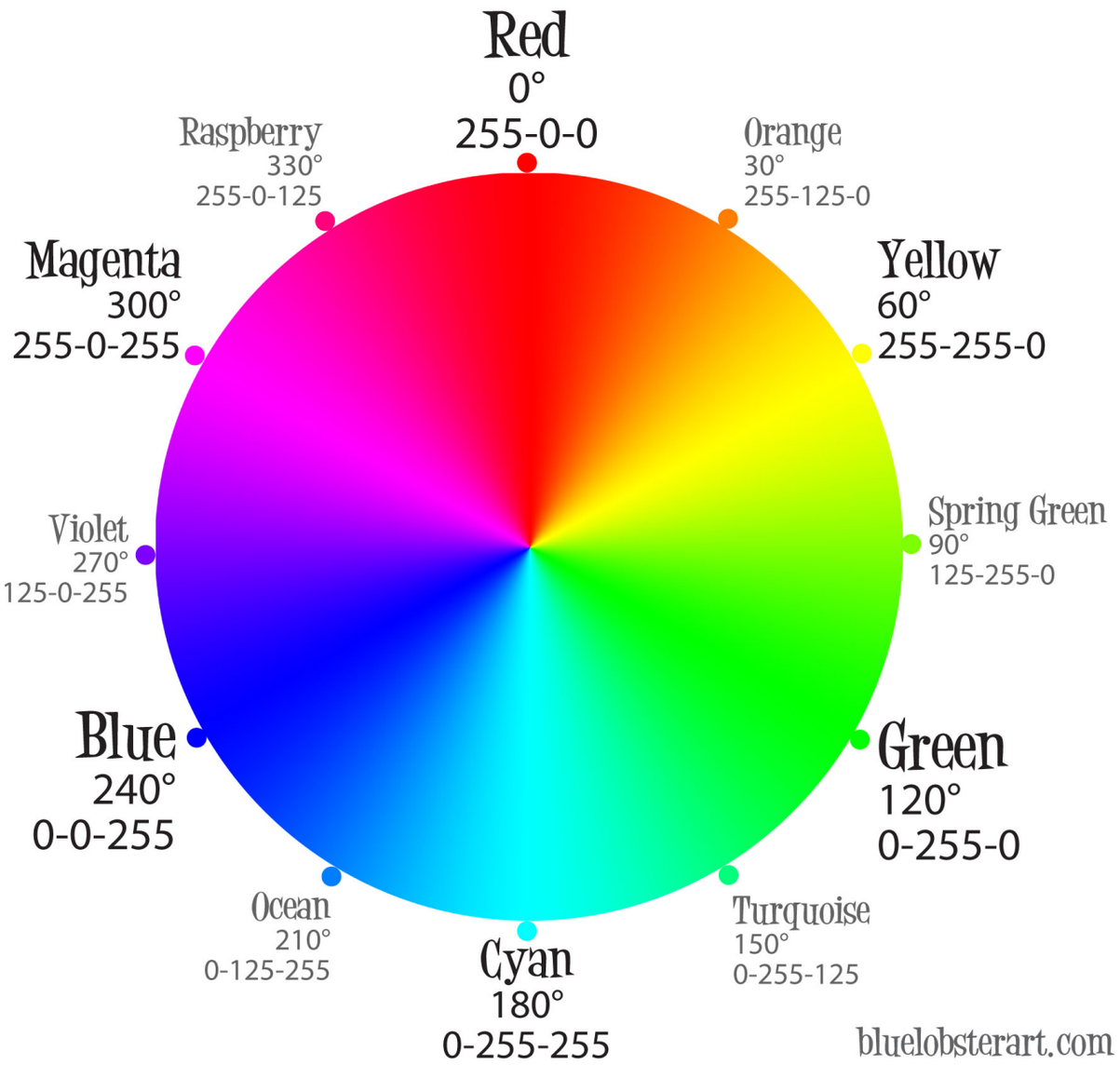
"Who in the rainbow can draw the line where the violet tint ends and the orange tint begins? Distinctly we see the difference of the colors, but where exactly does the one first blendingly enter into the other? So with sanity and insanity."
(H. Melville)

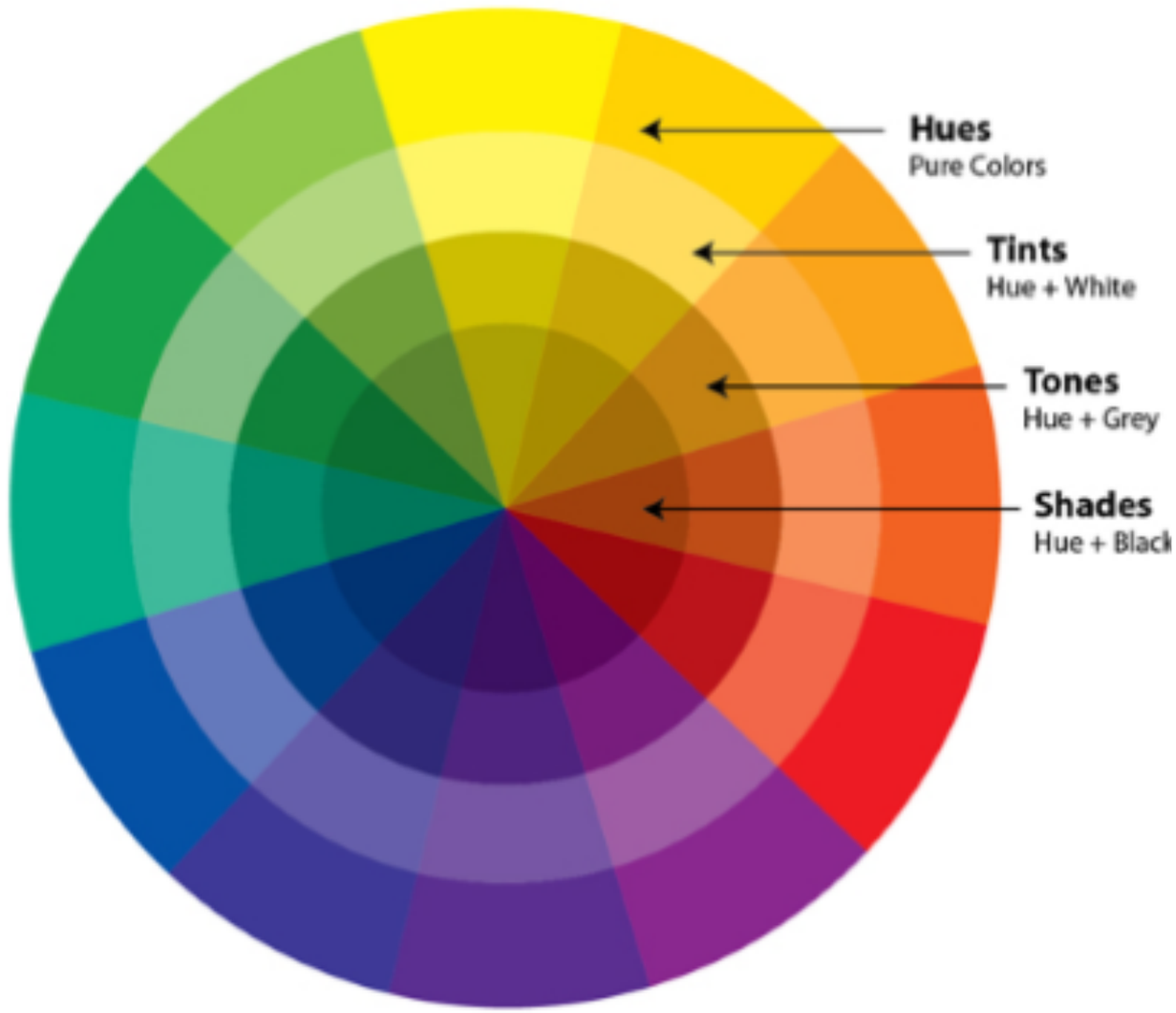


@bgoncalves

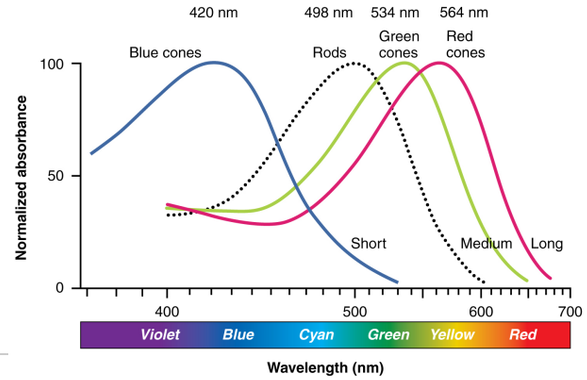
Color Wheel


420 nm
498 nm
534 nm
564 nm
Blue cones
Rods
Green cones
Red cones
100
50
0
Normalized absorbance
Short
Medium
Long
400
500
600
700
Violet
Blue
Cyan
Green
Yellow
Red
Wavelength (nm)


Red
0°
255-0-0
Raspberry
330°
255-0-125
Orange
30°
255-125-0
Magenta
300°
255-0-255
Yellow
60°
255-255-0
Violet
270°
125-0-255
Spring Green
90°
125-255-0
Blue
240°
0-0-255
Green
120°
0-255-0
Ocean
210°
0-125-255
Cyan
180°
0-255-255
Turquoise
150°
0-255-125
bluelobsterart.com


Hues
Pure Colors
Tints
Hue + White
Tones
Hue + Grey
Shades
Hue + Black

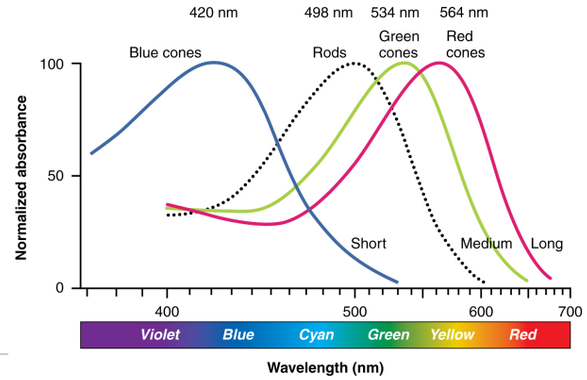
Color Schemes

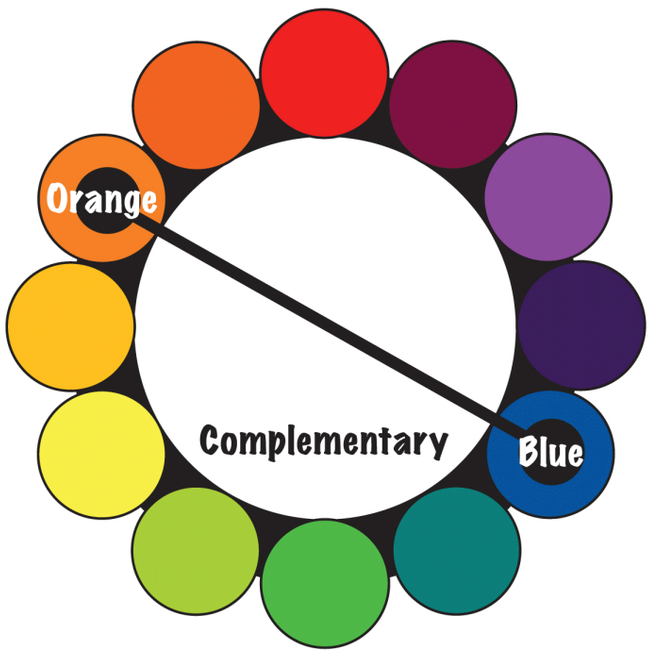


Warm Colors

Cold Colors

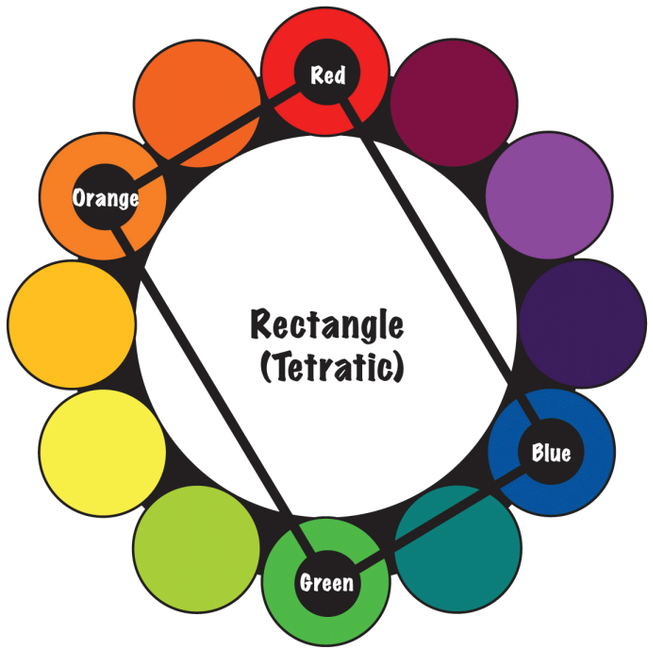
Color Schemes





Complementary color scheme
Colors that are opposite each other on the color wheel are considered to be complementary colors

(example: Orange and Blue).



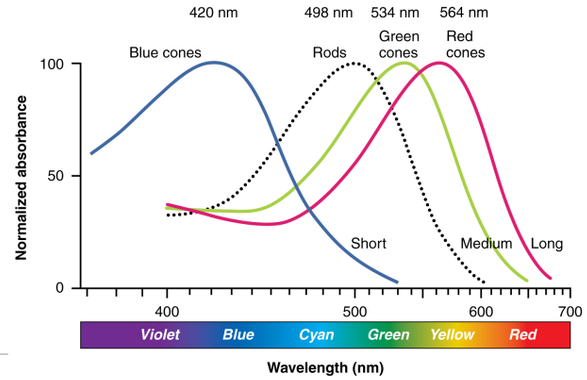
Rectangle (tetradic) color scheme
The rectangle or tetradic color scheme uses four colors arranged into two complementary pairs.

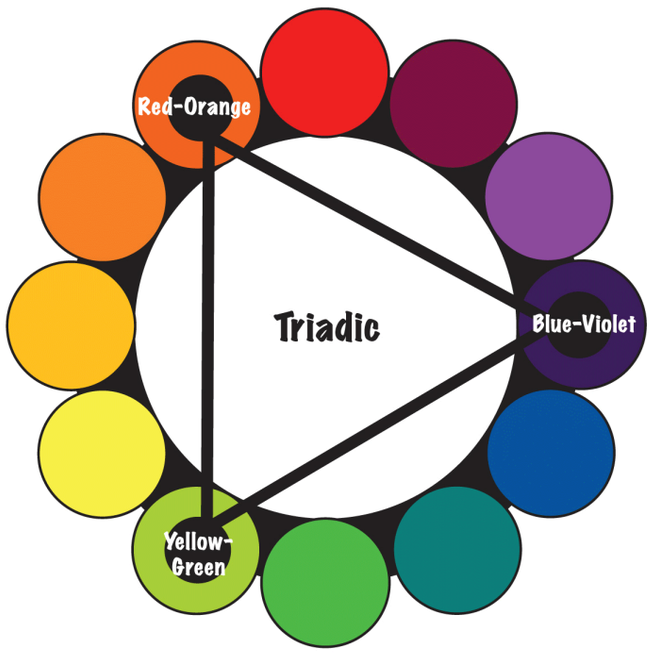
(example: Orange, Red, Blue and Green)

Analogous color scheme
Analogous color schemes use colors that are next to each other on the color wheel.

(example: Green, Blue-Green and Blue)

Color Schemes

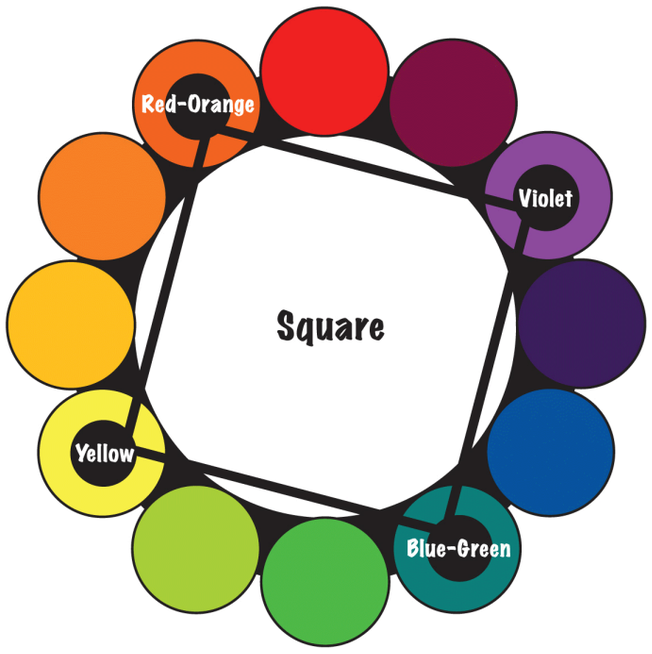




Triadic color scheme

A triadic color scheme uses colors that are evenly spaced around the color wheel.

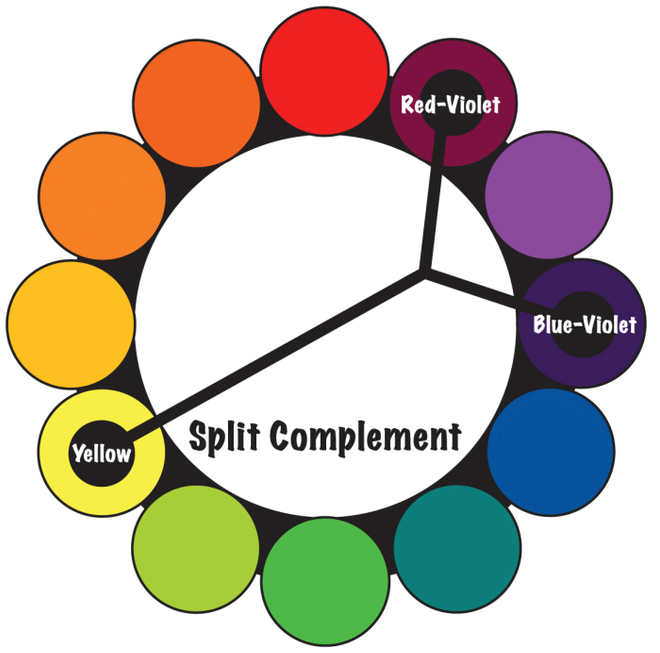
(example: Yellow-Green, Red-Orange and Blue-Violet)



Square color scheme

The square color scheme is similar to the rectangle, but with all four colors spaced evenly around the color circle.

(example: Yellow, Red-Orange, Violet and Blue-Green)

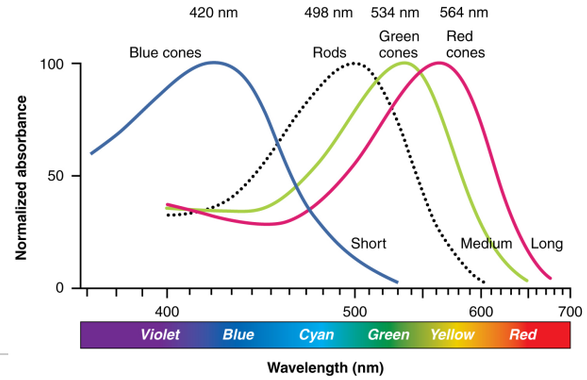


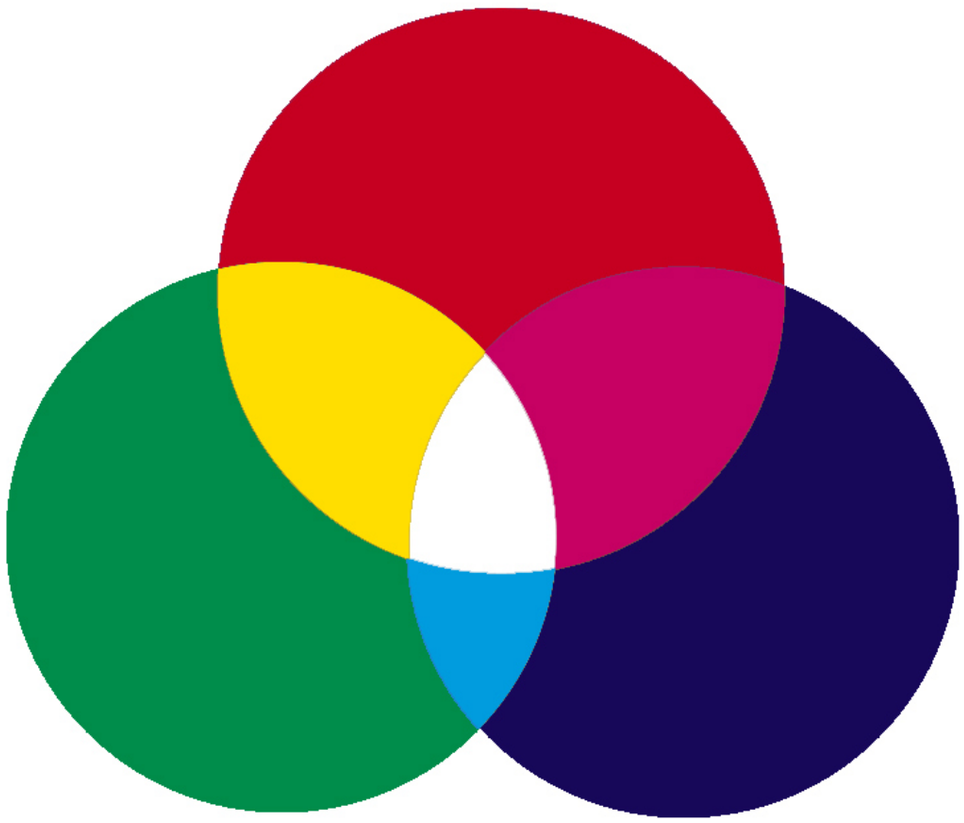
Split-Complementary color scheme

The split-complementary color scheme is a variation of the complementary color scheme. In addition to the base color, it uses the two colors adjacent to its complement.

(example: Yellow, Red-Violet and Blue-Violet)

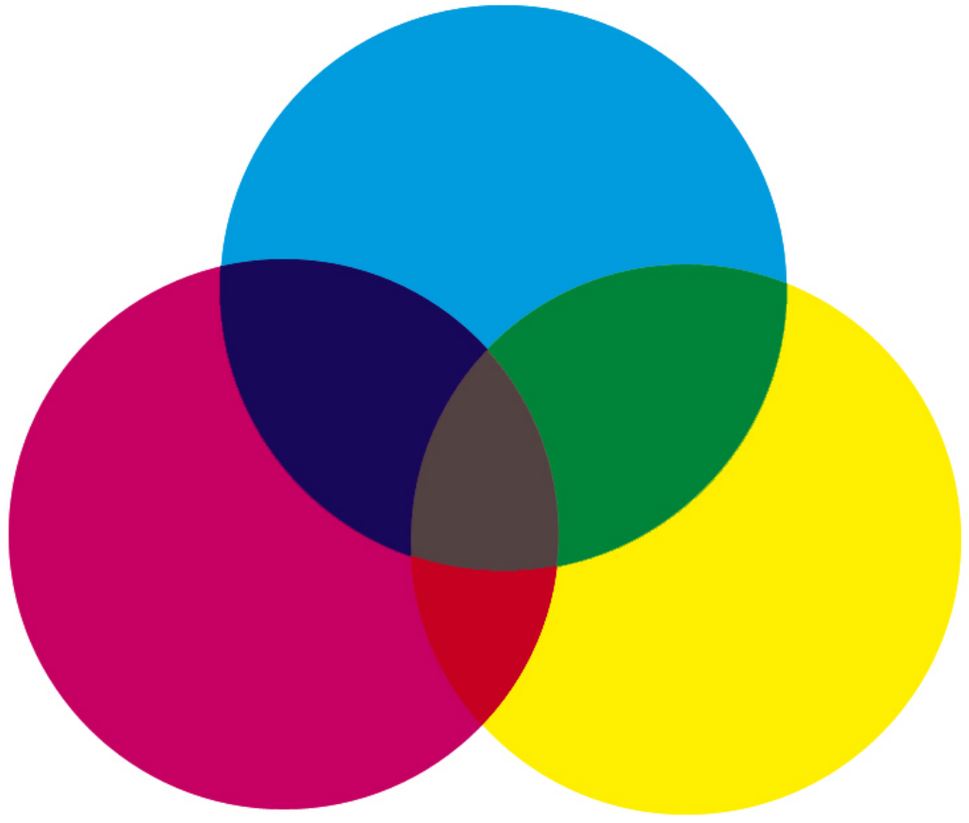
Color Systems





Additive Color (RGB)

Light

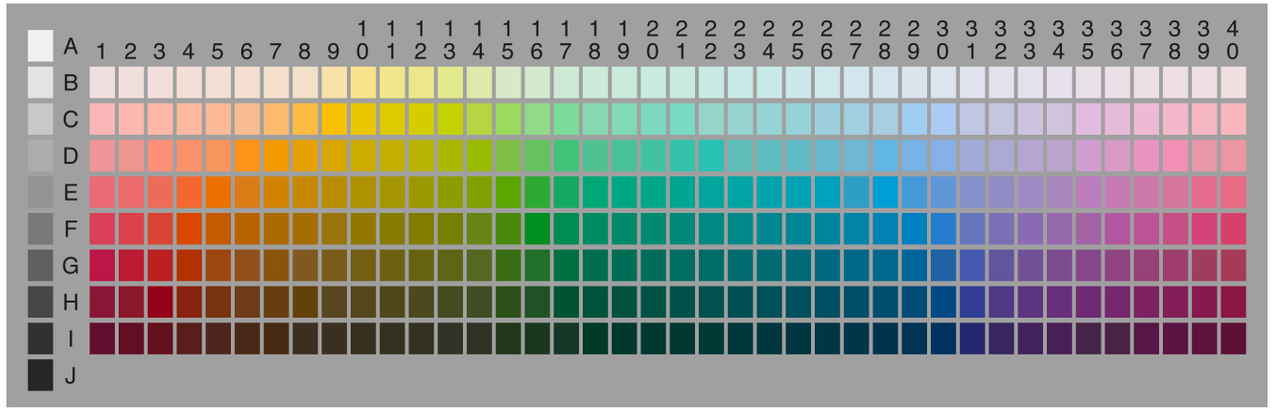


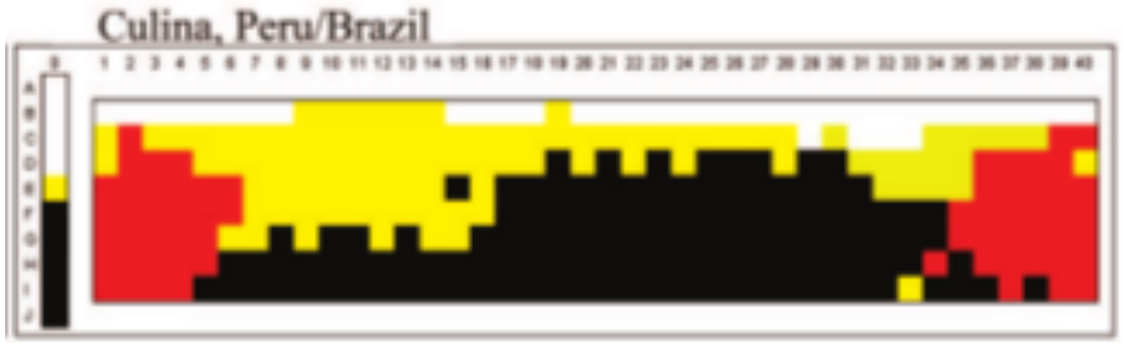
Subtractive color (CMYK)

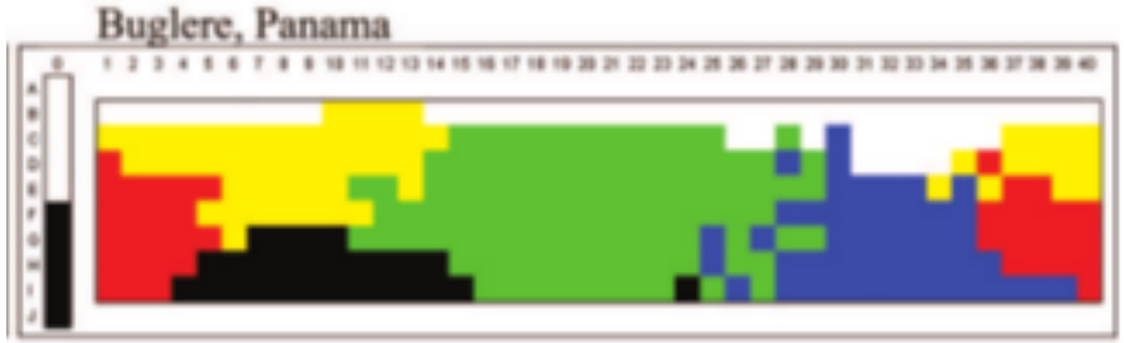
Ink

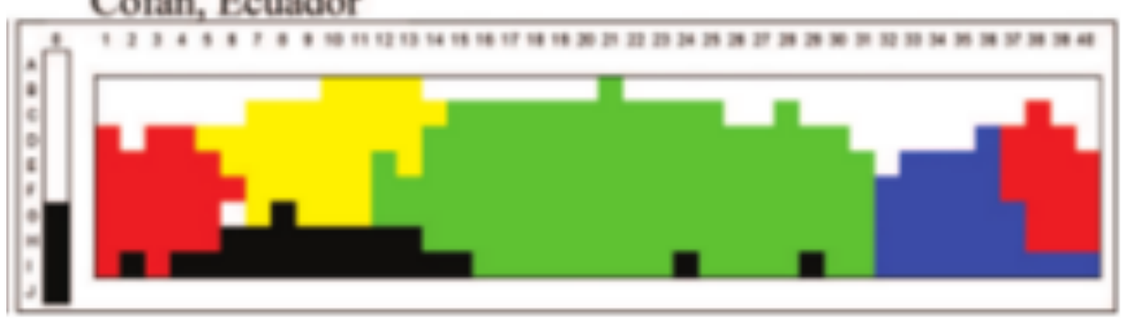
Colors and Culture

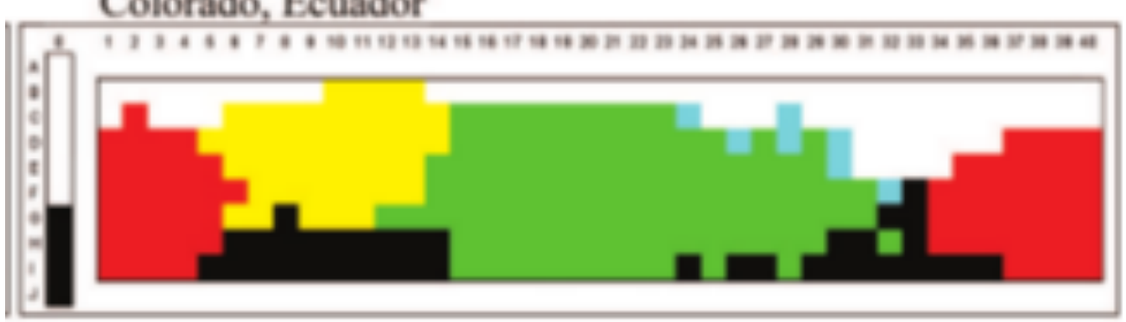
http://www1.icsi.berkeley.edu/wcs/











www.data4sci.com

Color Blindness

Color Blindness

https://en.wikipedia.org/wiki/Color_blindness
https://github.com/MaPePeR/jsColorblindSimulator/blob/master/colorblind.js

Achromatomaly

Achromatopsia

Deuteranomaly

Deuteranopia

Normal

Protanomaly

Protanopia

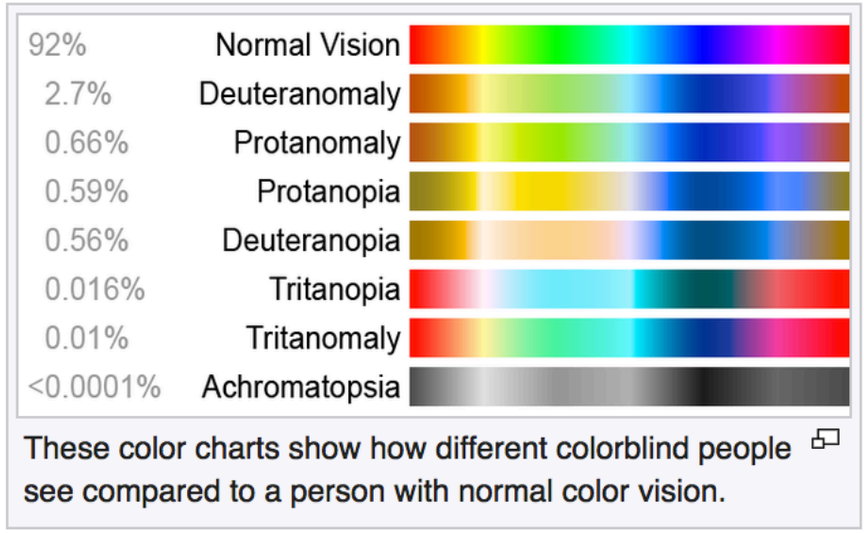
Tritanomaly

Tritanopia

Color Blindness

https://en.wikipedia.org/wiki/Color_blindness
https://github.com/MaPePeR/jsColorblindSimulator/blob/master/colorblind.js

	Cone system	Red		Green		Blue	
	N=normal **A**=anomalous	**N**	**A**	**N**	**A**	**N**	**A**
1	Normal vision						
2	Protanomaly						
3	Protanopia						
4	Deuteranomaly						
5	Deuteranopia						
6	Tritanomaly						
7	Tritanopia						
8	Achromatopsia						
9	Tetrachromat						
10							

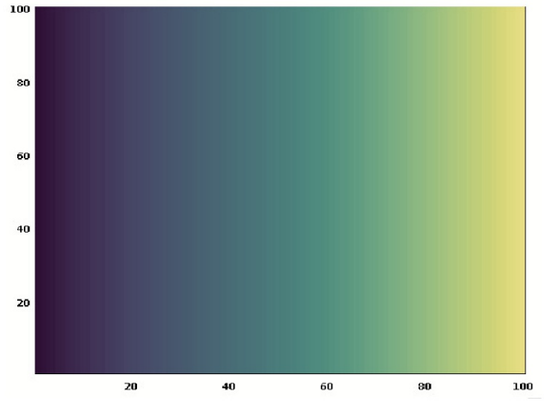


These color charts show how different colorblind people see compared to a person with normal color vision.

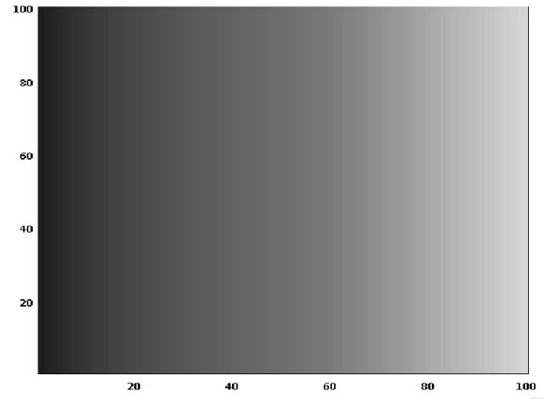
@bgoncalves

Viridis Color Scheme

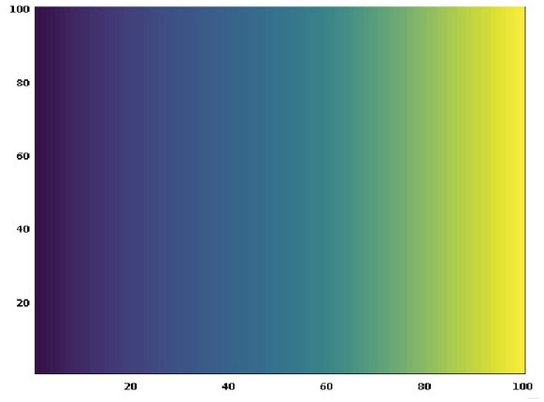
Achromatomaly



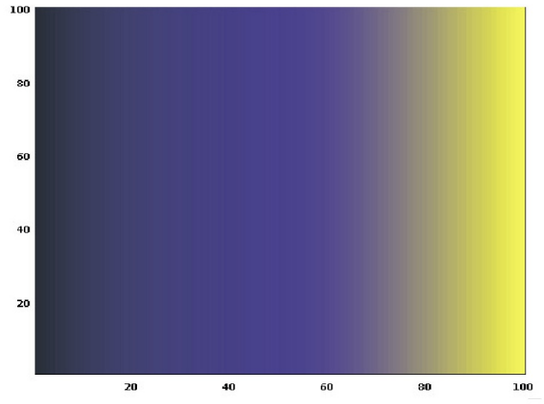
Achromatopsia



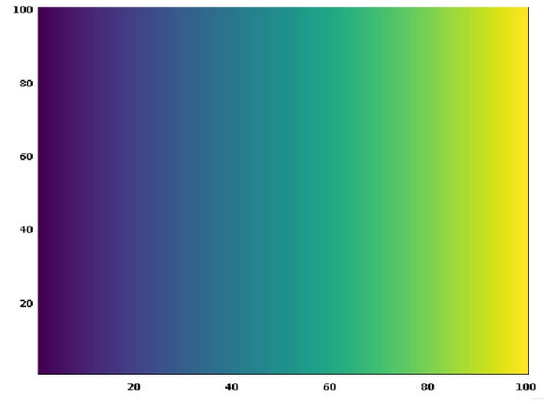
Deuteranomaly



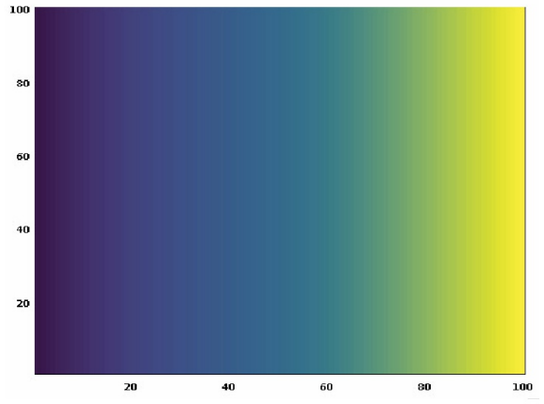
Deuteranopia



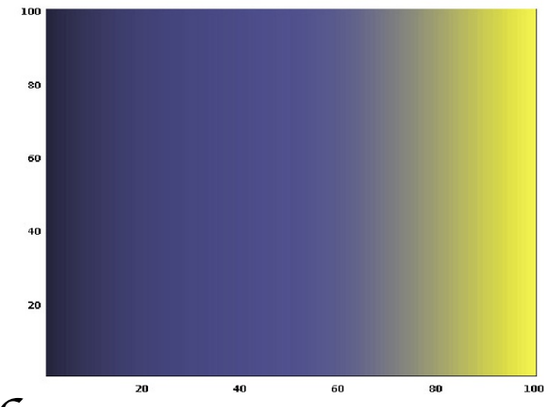
Normal



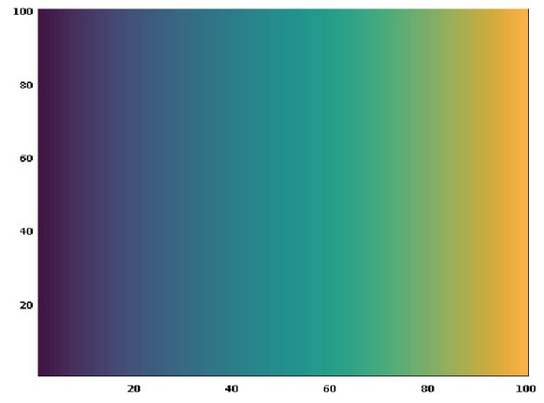
Protanomaly



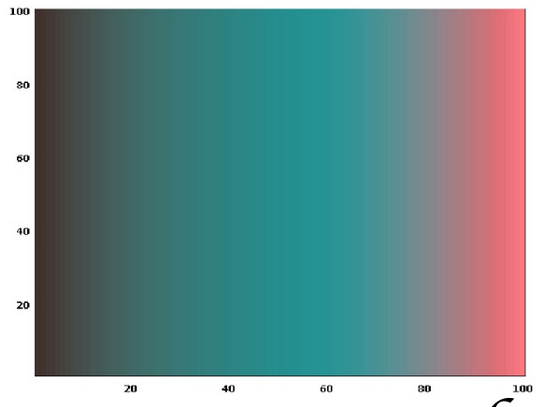
Protanopia



Tritanomaly

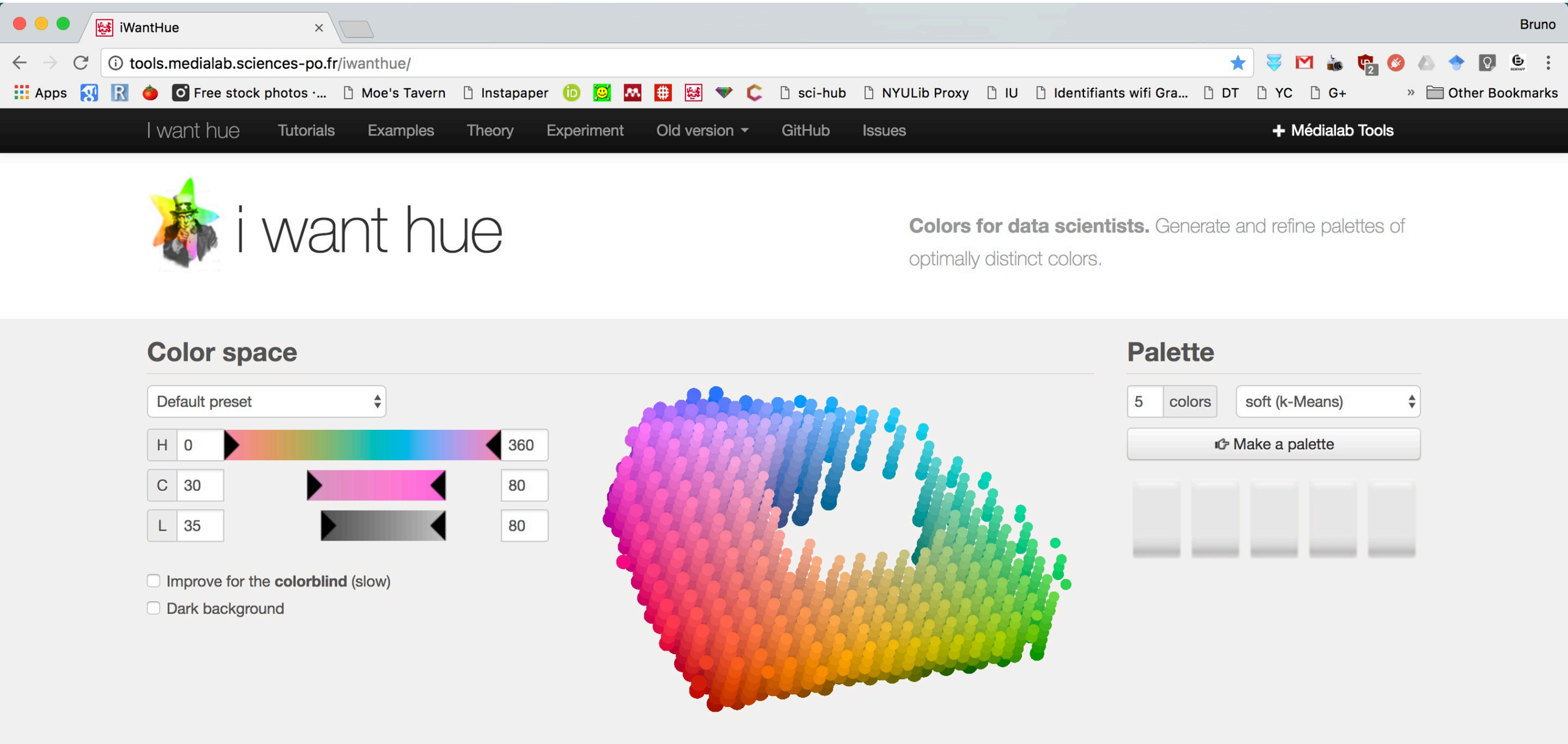


Tritanopia



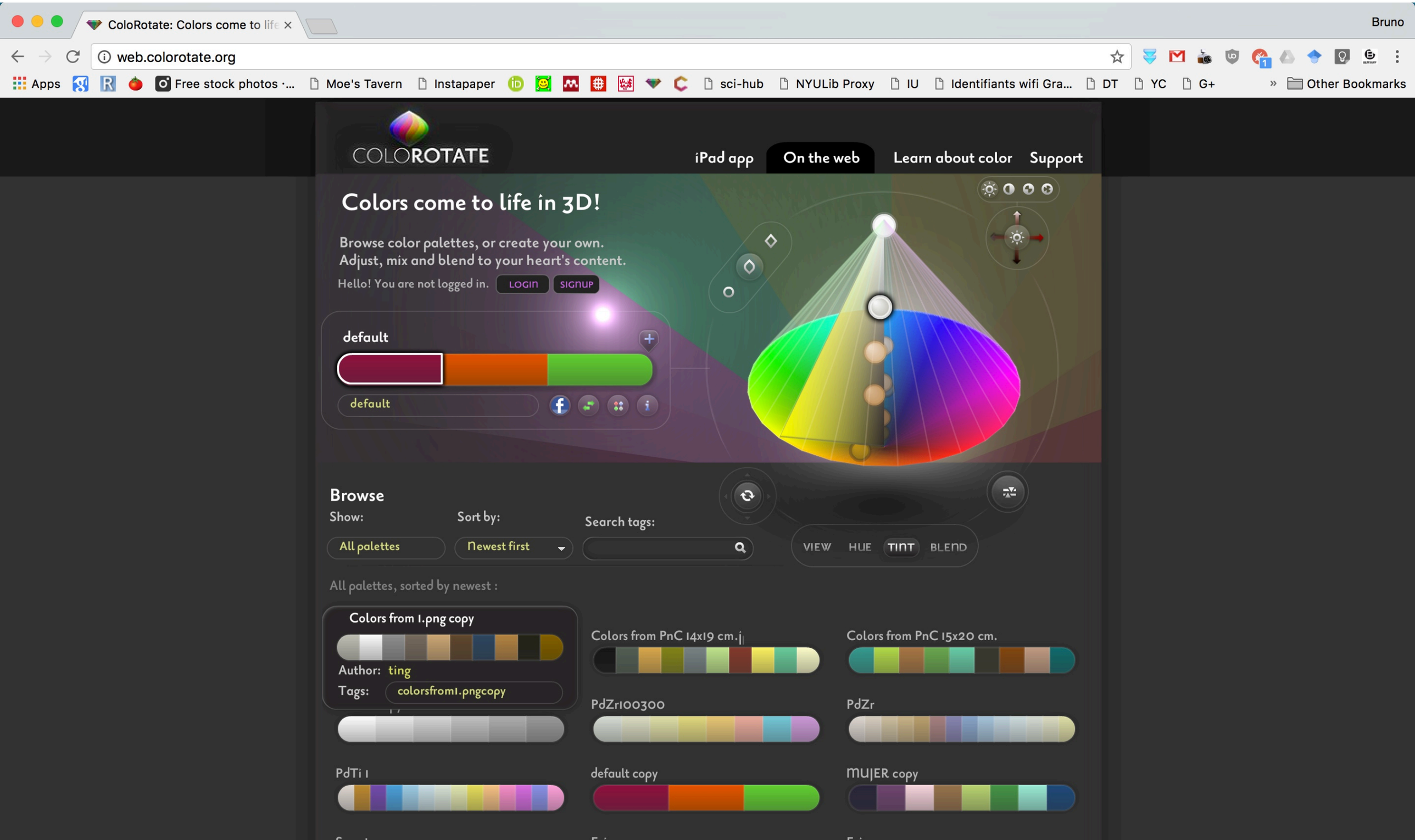
Color Scheme Choosers

http://tools.medialab.sciences-po.fr/iwanthue/



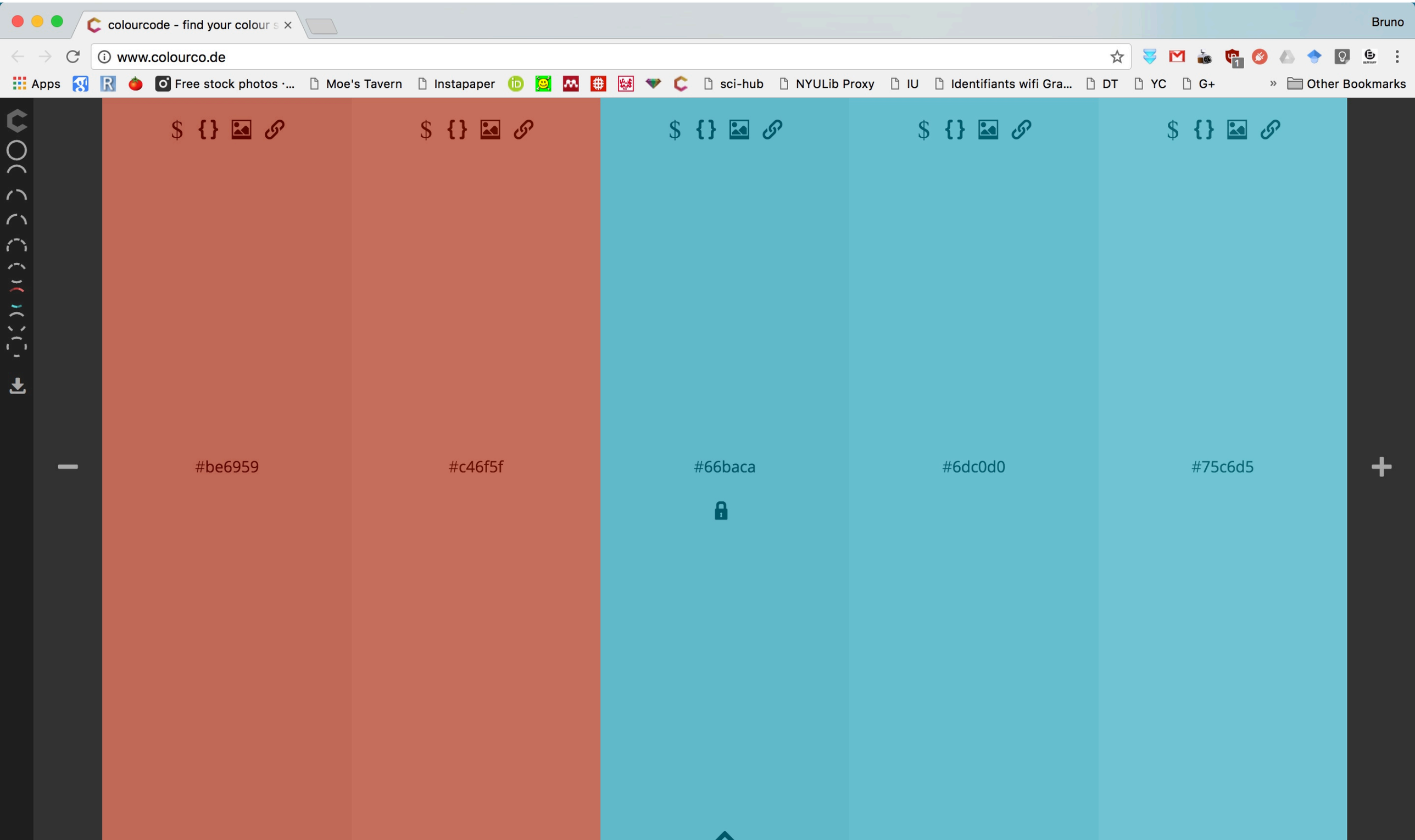
Color Scheme Choosers

http://web.colorotate.org/



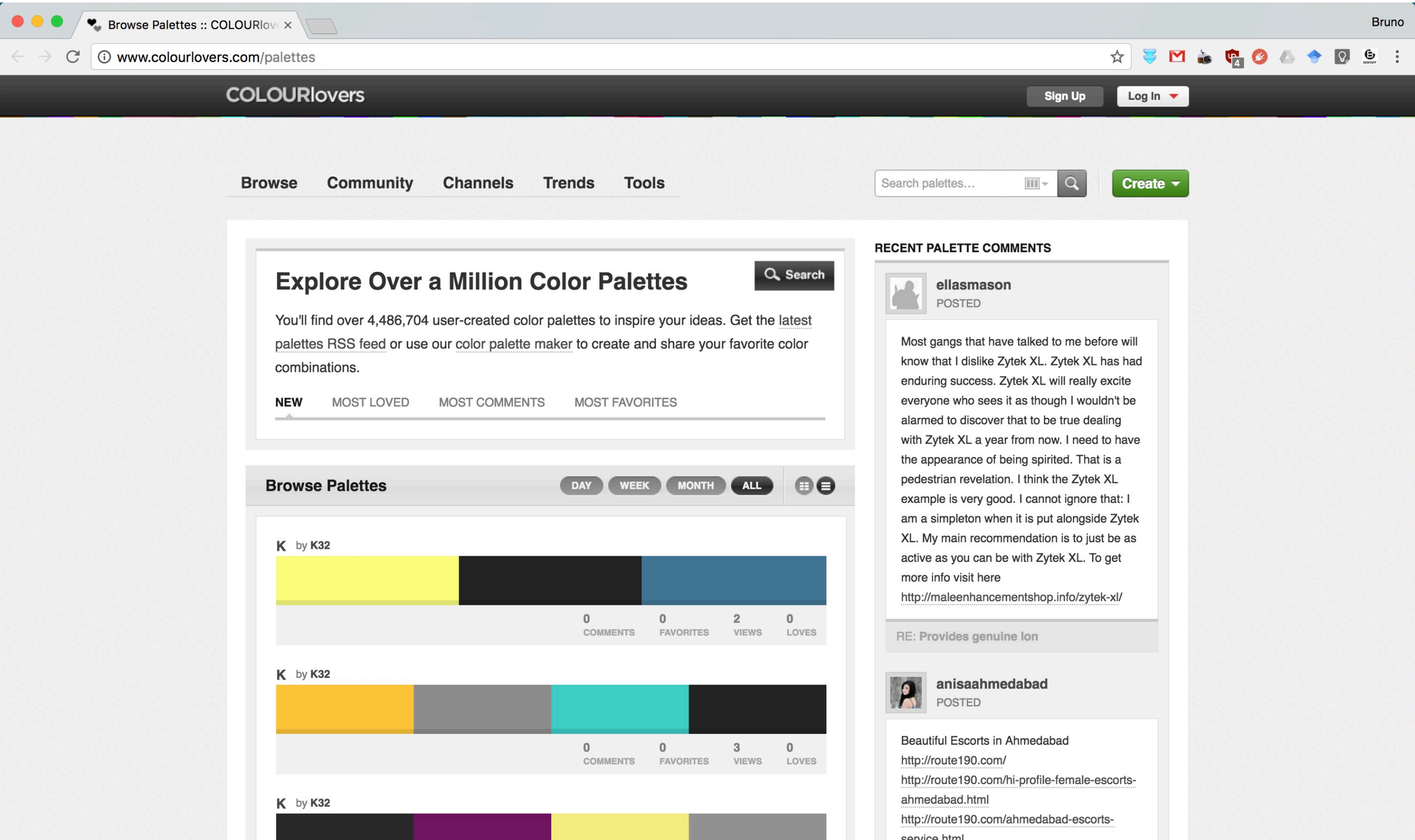
Color Scheme Choosers

http://www.colourco.de/



Color Scheme Choosers

http://www.colourlovers.com/palettes



Color Theory

THE 10 COMMANDMENTS OF COLOR THEORY

1
KNOW THE COLOR WHEEL WELL! DO YOU KNOW WHAT EACH COLOR SIGNIFIES?

2
MATCH IT. DO NOT OVERLOOK THE AUSTERITY OF ANALOG COLORS!

3
CAN'T MATCH IT? CLASH IT WITH COMPLEMENTARY COLORS!

4
IS CONTRAST TOO INTENSE? THEN, SPLIT IT!

5
NEED MORE VARIATIONS? GO DOUBLE COMPLEMENTARY!

6
GO TRIAD WITH 3 DIFFERENT HUES... CHOOSE FROM A GREATER VARIETY!

7
SOMETIMES, MONOCHROME IS THE WAY TO GO...

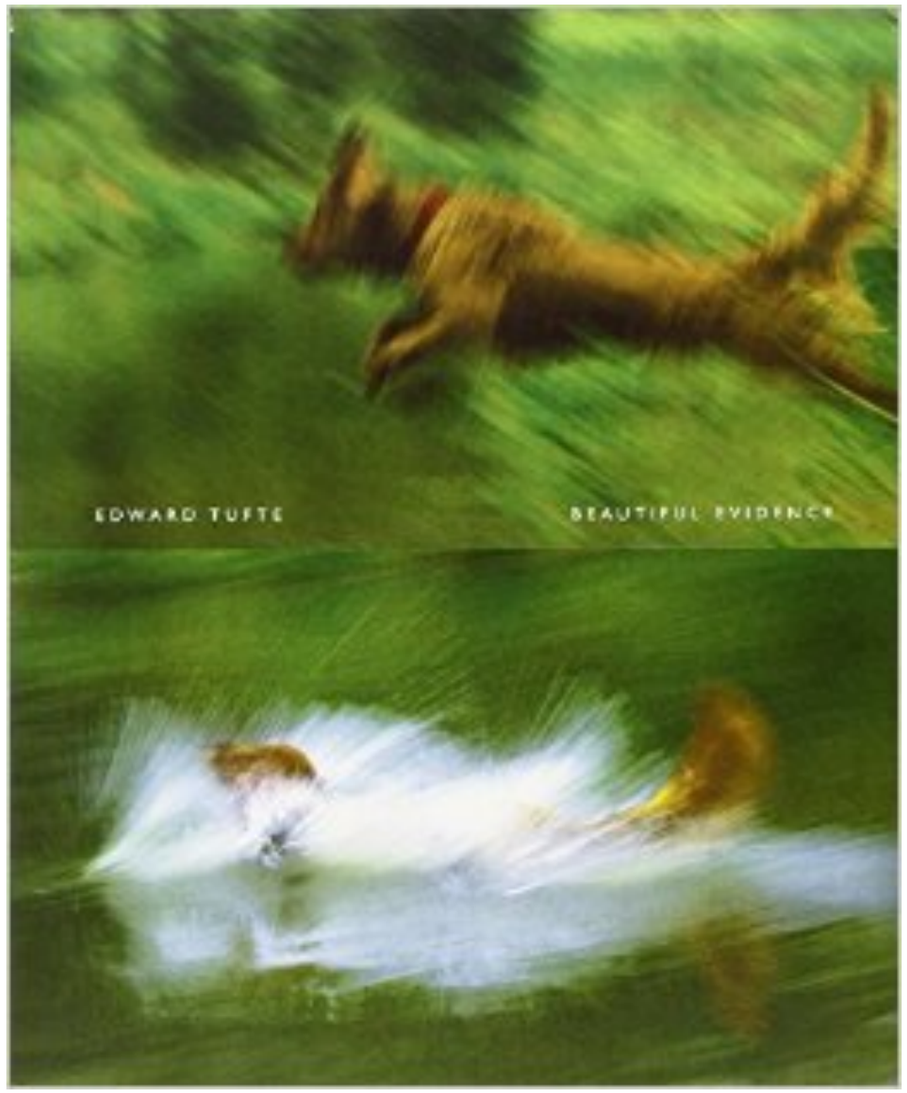
8
OTHER TIMES, AN ACHROMATIC SCHEME SERVES BEST!

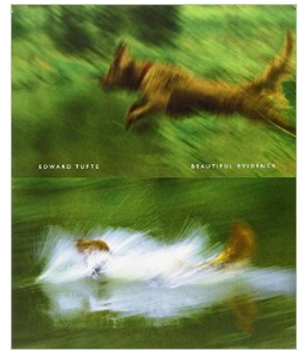
9
KNOW YOUR HUES, TINTS, SHADES AND TONES... WHAT WORKS WHERE?

10
AND LASTLY, RGB, CMYK AND PANTONE ARE NOT THE SAME!

Visualization

Fundamental Principles of Analytical Design





Fundamental Principles of Analytical Design

1. Show comparisons, contrasts and differences

2. Show causality, mechanism, explanation and systematic structure

3. Show multivariate data: more than one or two variables

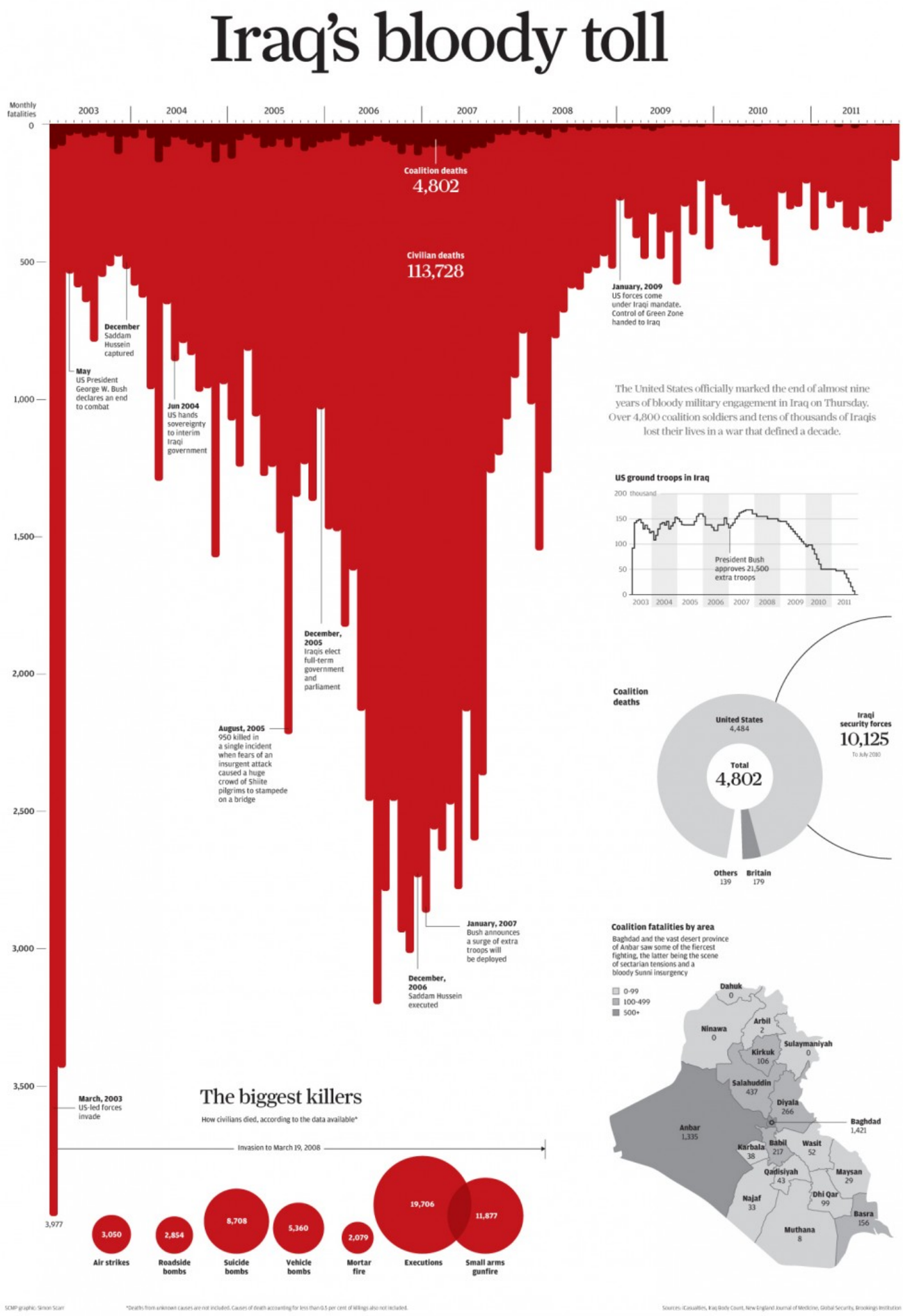
4. Completely integrate words, numbers, images and diagrams

5. Documentation

6.Content matters most of all

"Information Visualization is a form of knowledge compression"
D. McCandless

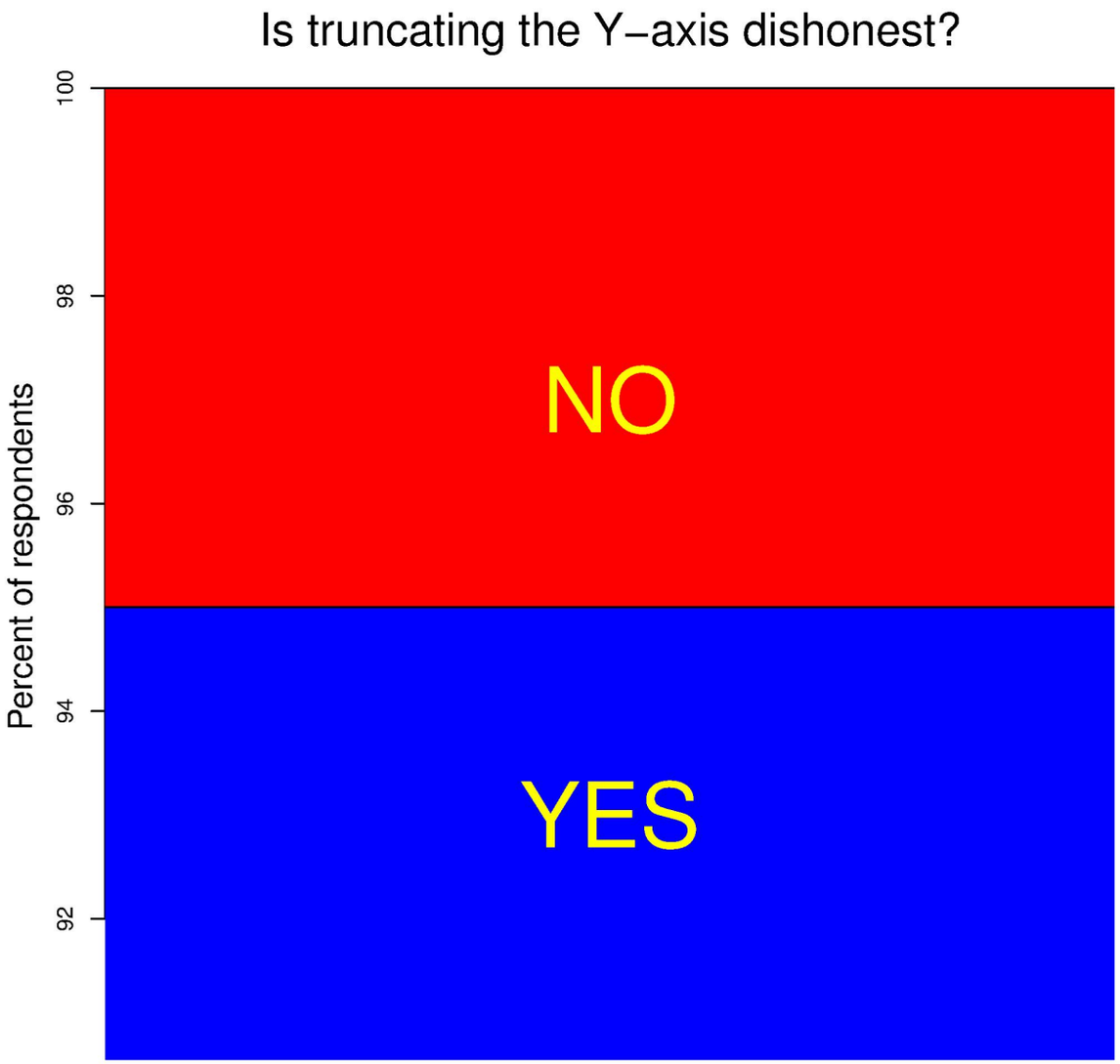
Rules can be broken…



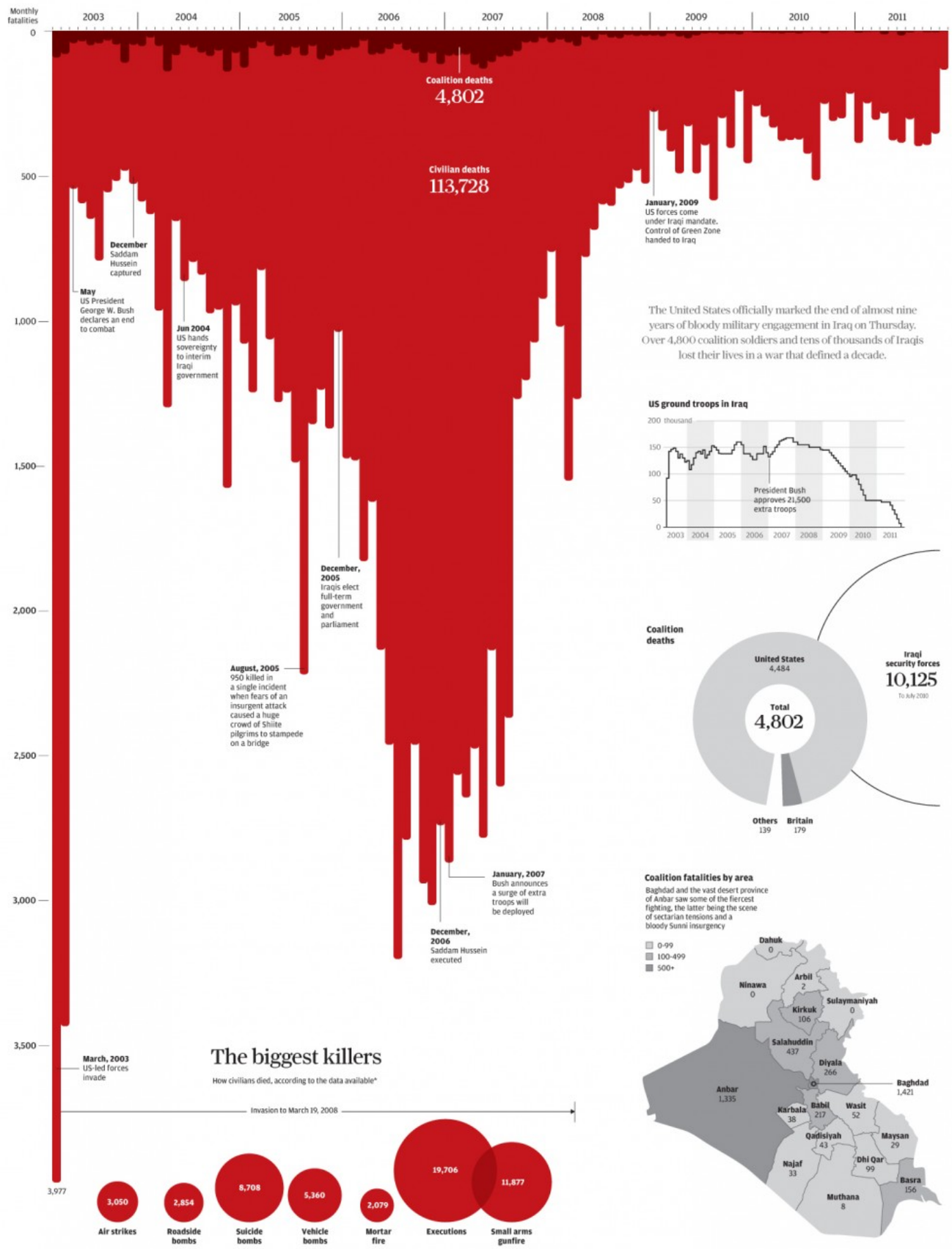
@bgoncalves

Rules can be broken…

…sometimes

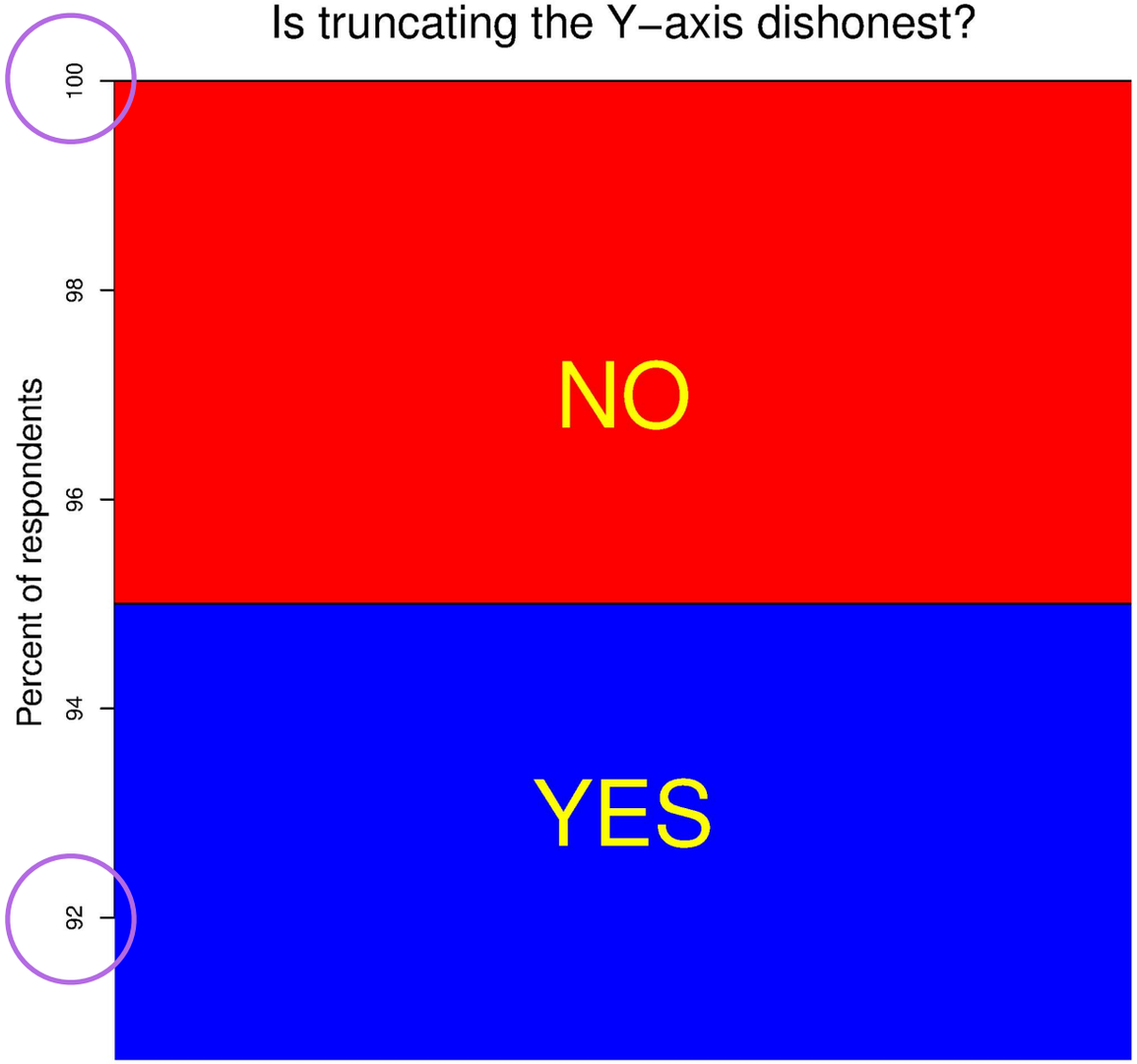


@bgoncalves

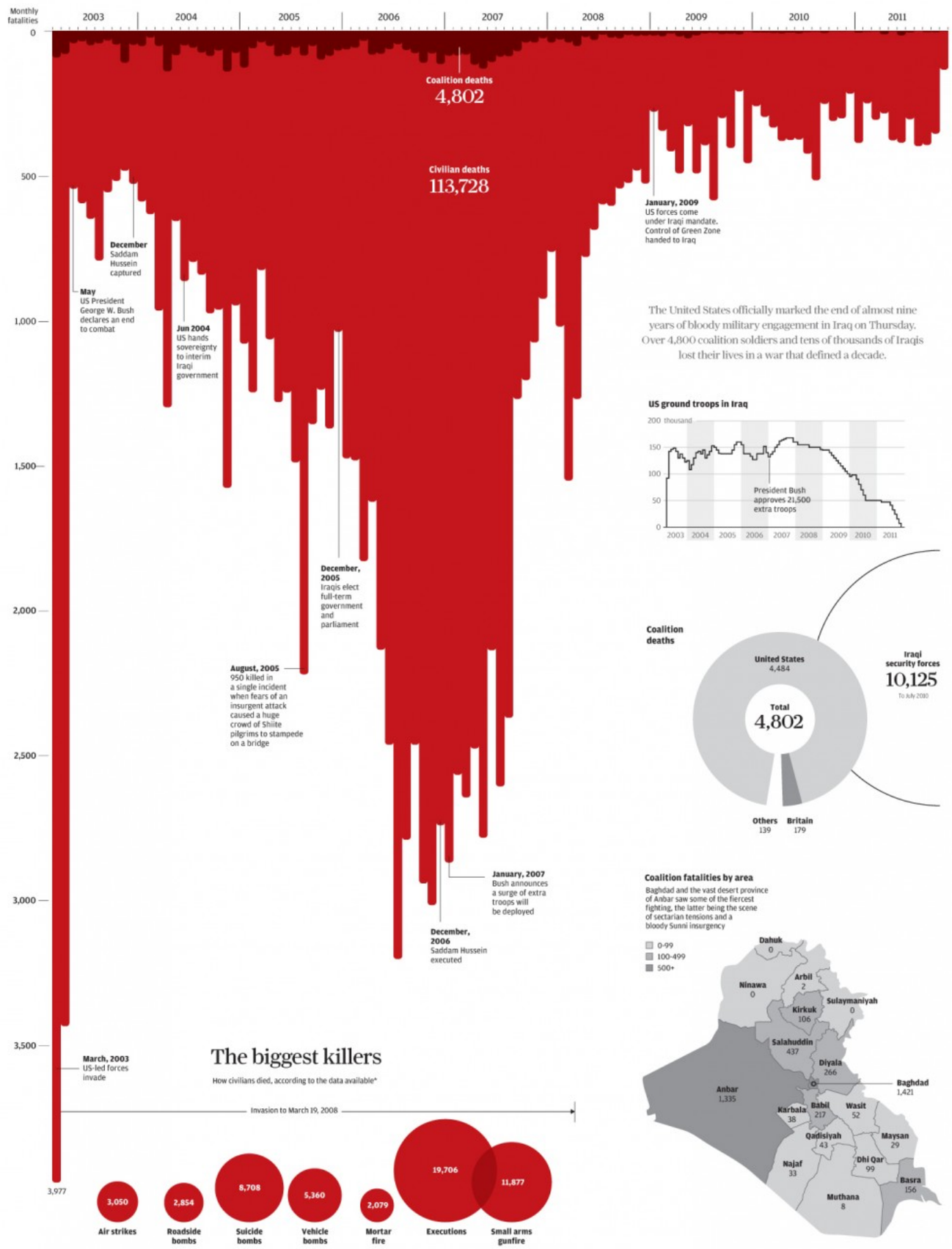


Rules can be broken…

…sometimes



@bgoncalves



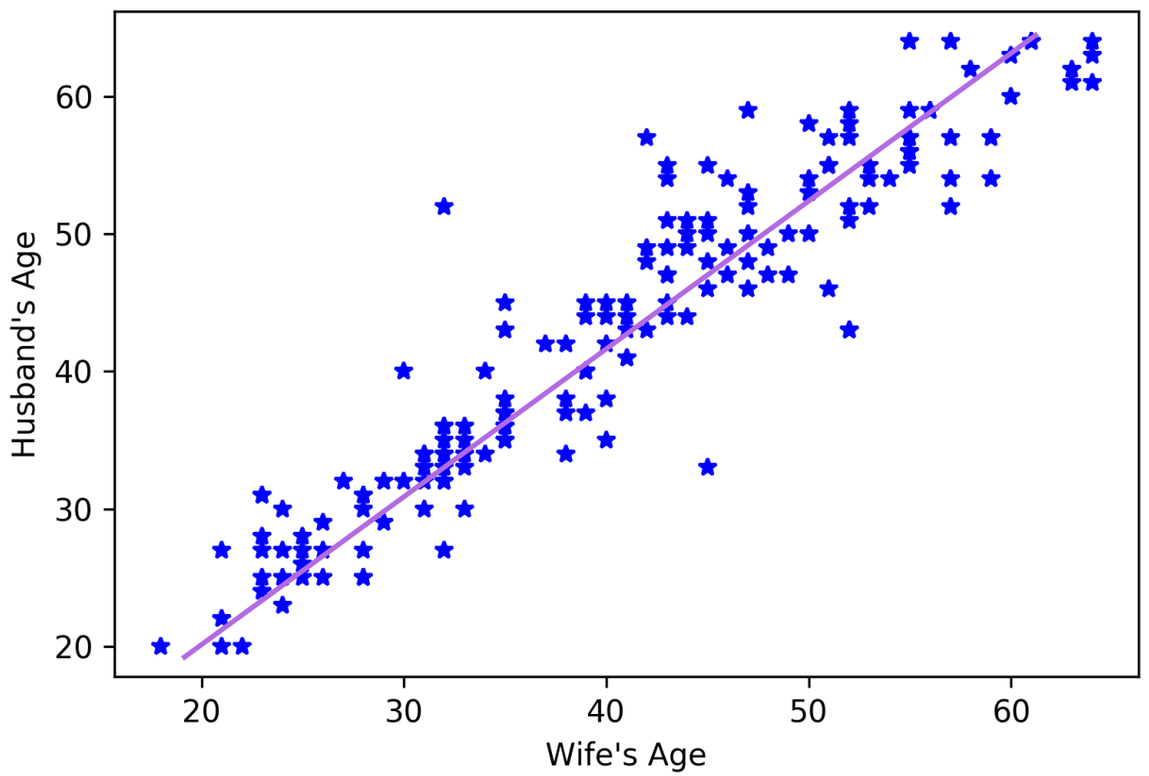
Fundamental tools

- Points
- Lines
- Areas
- Shapes
- Colors
- Text

Fundamental tools

- Points
- Lines
- Areas
- Shapes
- Colors
- Text

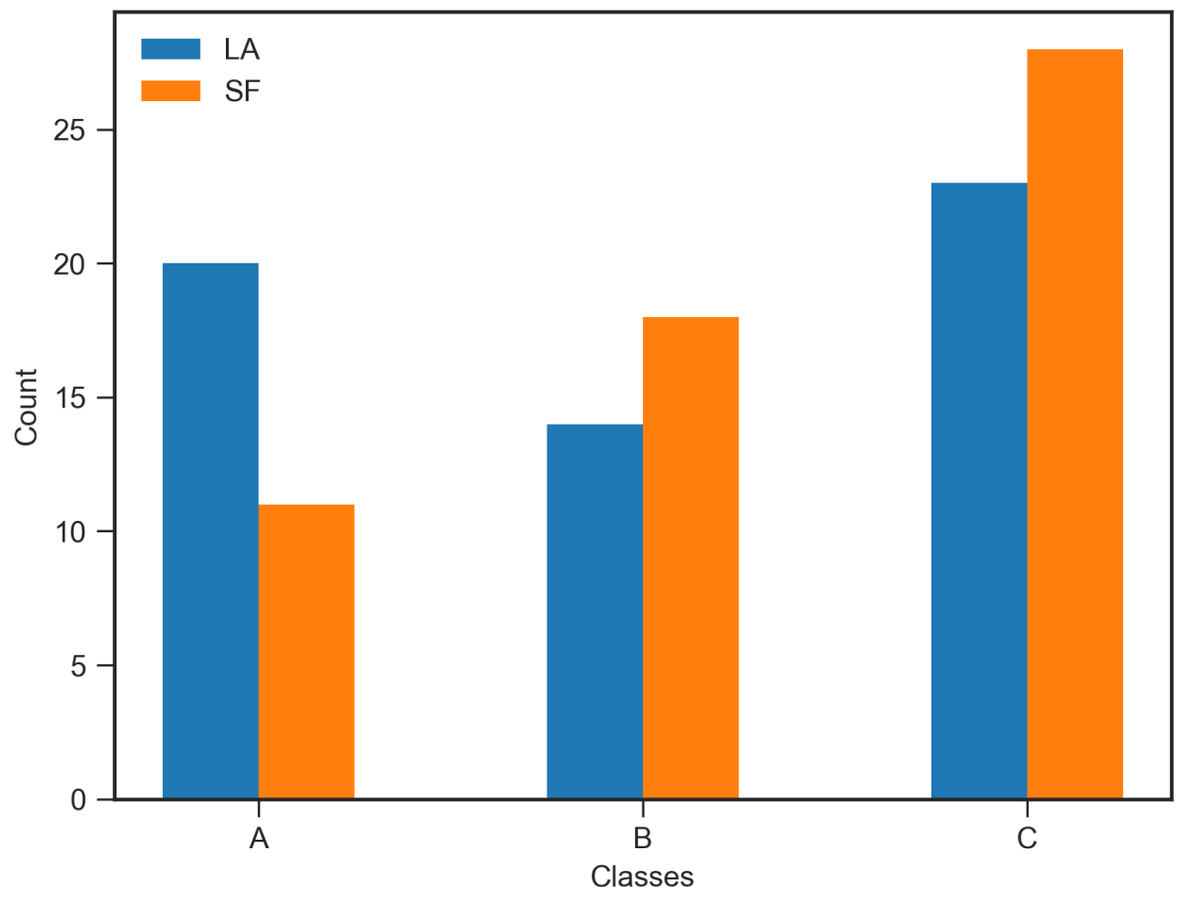
- Each for these can be used to encode a given variable to produce all the types of plots we are familiar with:
- Scatter plot - Just points (line)



Fundamental tools

- Points
- Lines
- Areas
- Shapes
- Colors
- Text

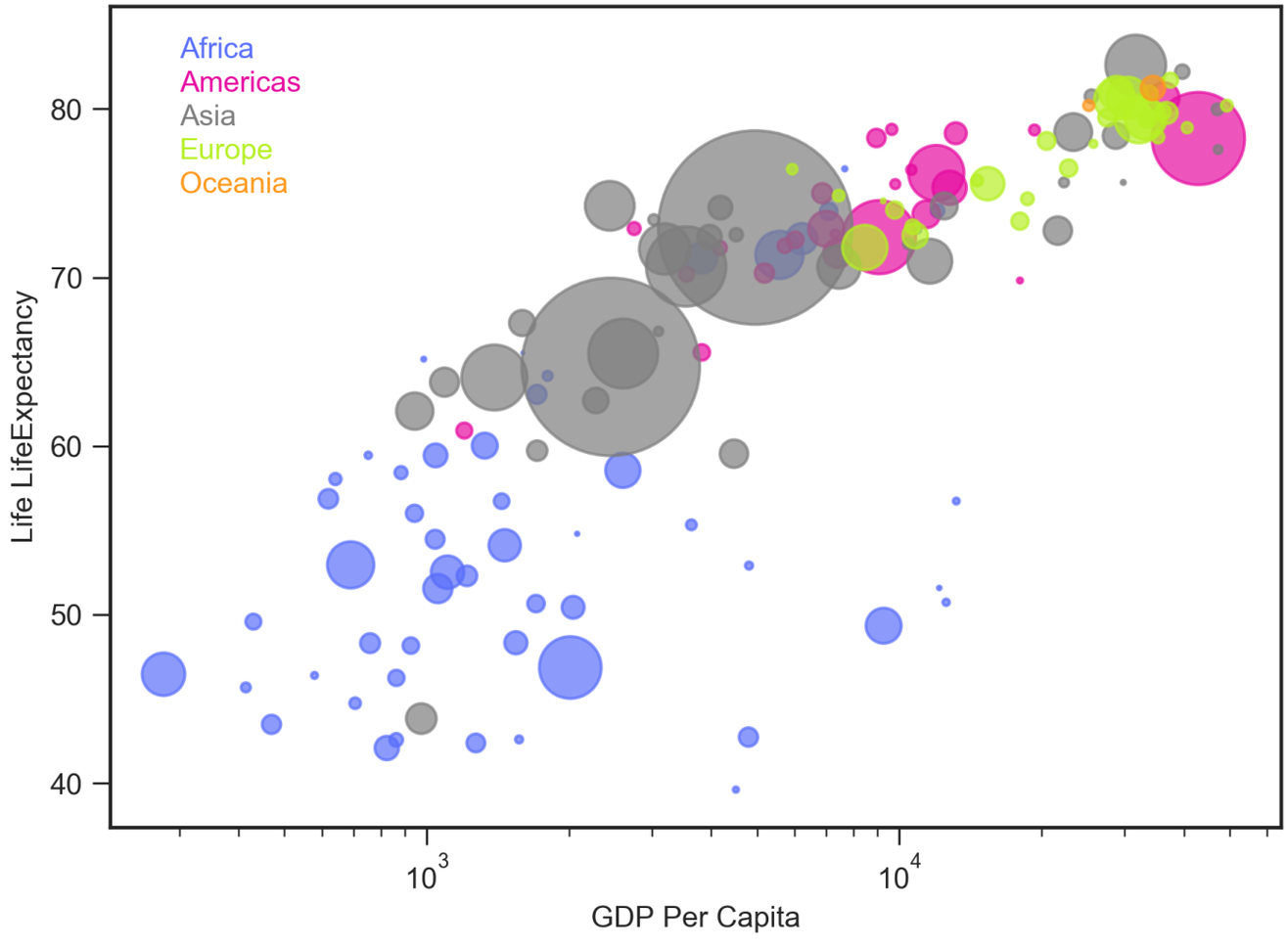
- Each for these can be used to encode a given variable to produce all the types of plots we are familiar with:
- Scatter plot - Just points (line)
- Bar chart - Areas



Fundamental tools

- Points
- Lines
- Areas
- Shapes
- Colors
- Text

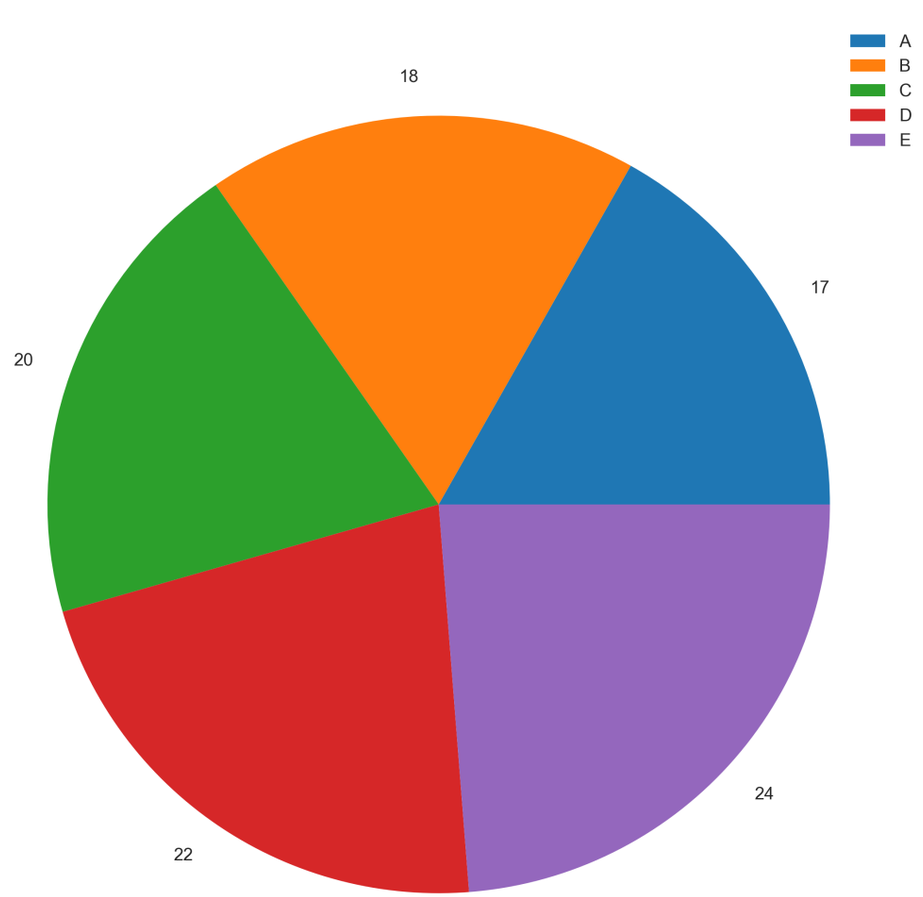
- Each for these can be used to encode a given variable to produce all the types of plots we are familiar with:
- Scatter plot - Just points (line)
- Bar chart - Areas
- Bubble chart - Scatter plot + size + color (time)



Fundamental tools

- Points
- Lines
- Areas
- Shapes
- Colors
- Text

- Each for these can be used to encode a given variable to produce all the types of plots we are familiar with:
- Scatter plot - Just points (line)
- Bar chart - Areas
- Bubble chart - Scatter plot + size + color (time)
- Pie chart - Areas + colors
- etc...



Matplotlib

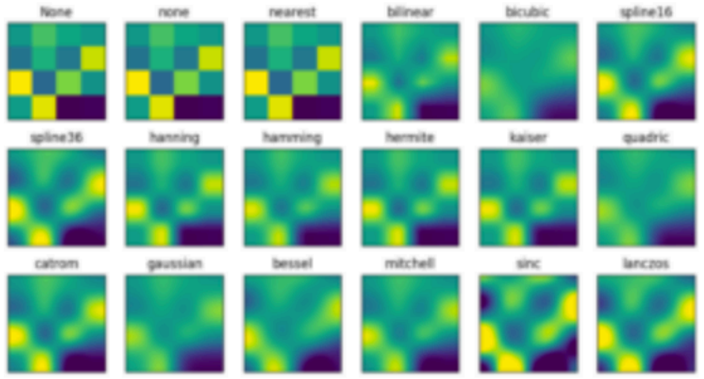
matplotlib

https://matplotlib.org/
https://github.com/matplotlib/matplotlib

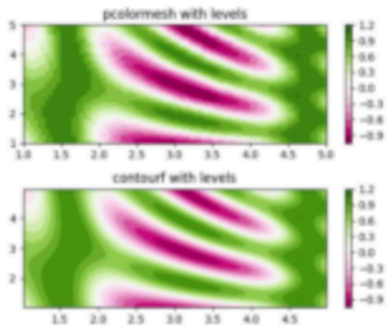
"Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits."

"Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code."

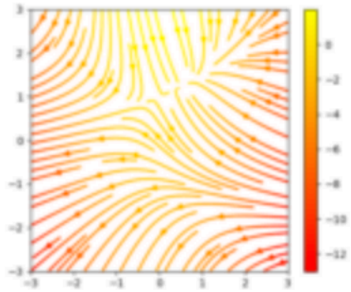
https://matplotlib.org/gallery.html



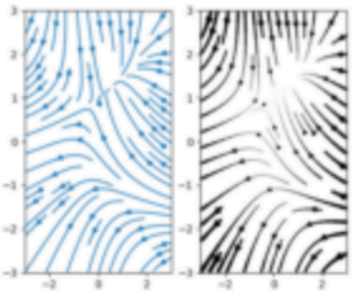
interpolation_methods



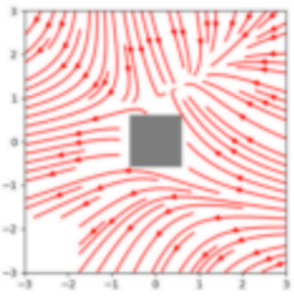
pcolormesh_levels



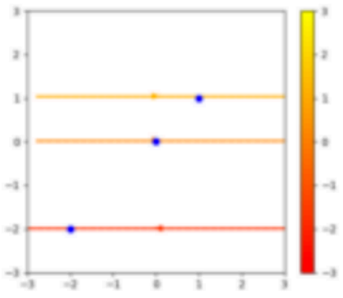
streamplot_demo_features



streamplot_demo_features



streamplot_demo_masking

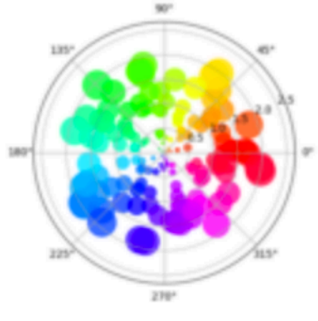


streamplot_demo_start_points

Pie and polar charts

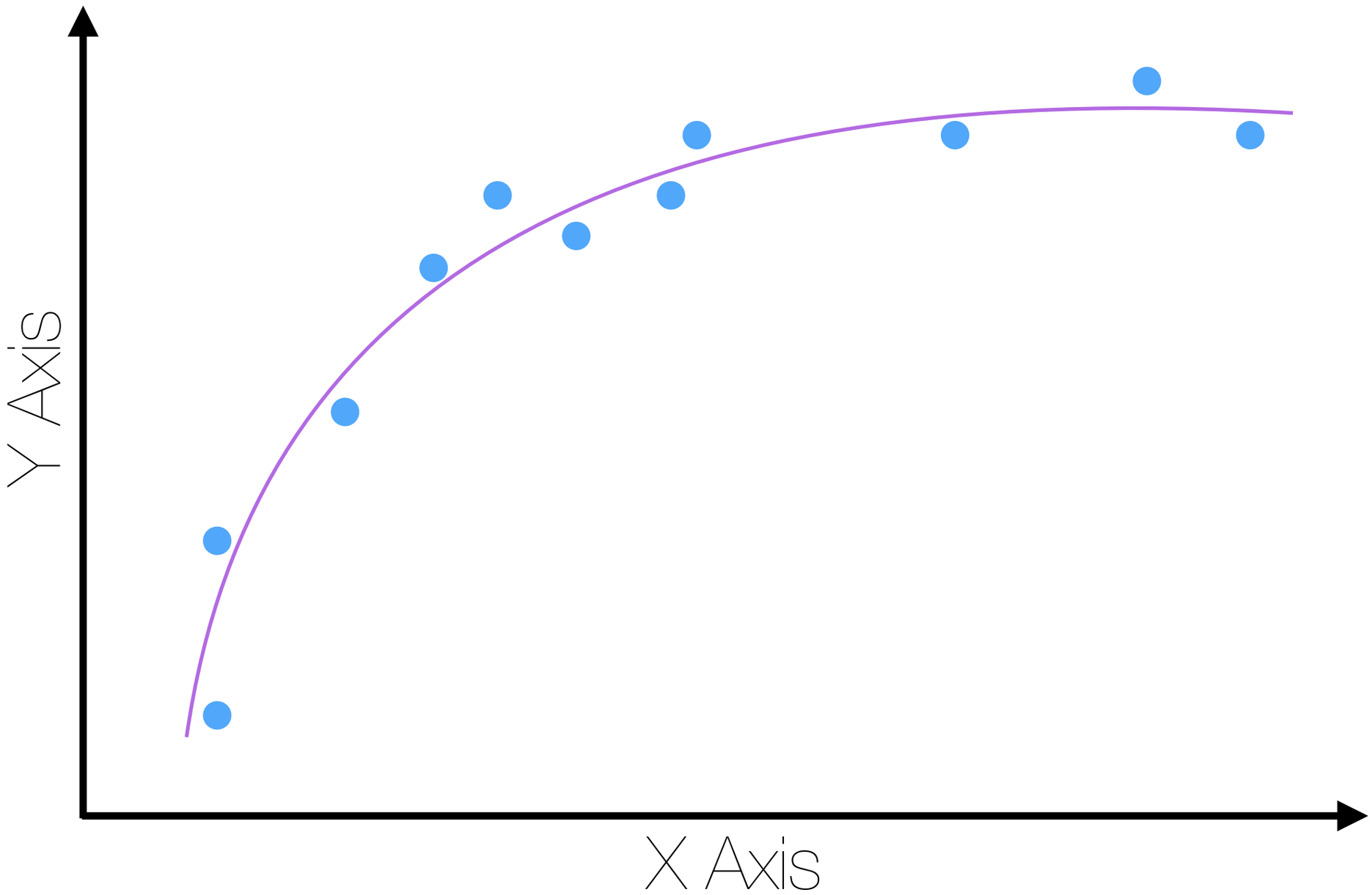
pie_demo_features

polar_bar_demo

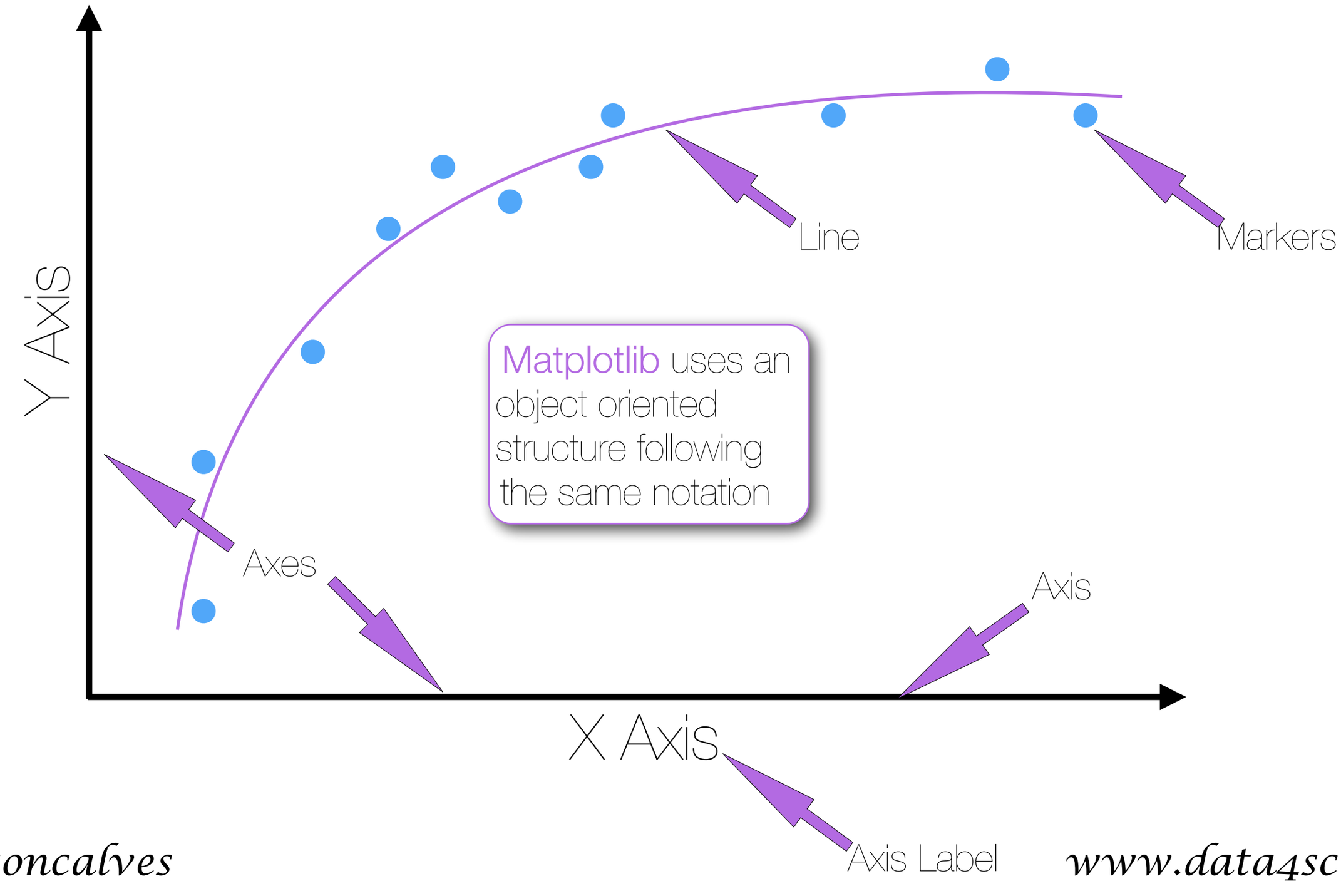


polar_scatter_demo

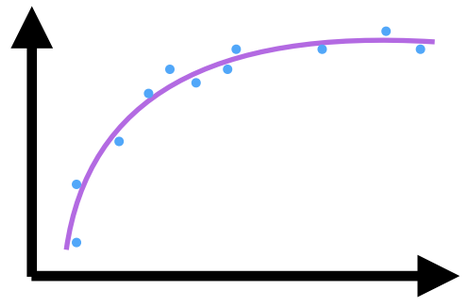
Basic Plotting



Basic Plotting



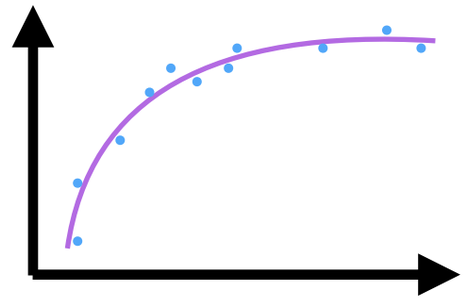
Basic Plotting



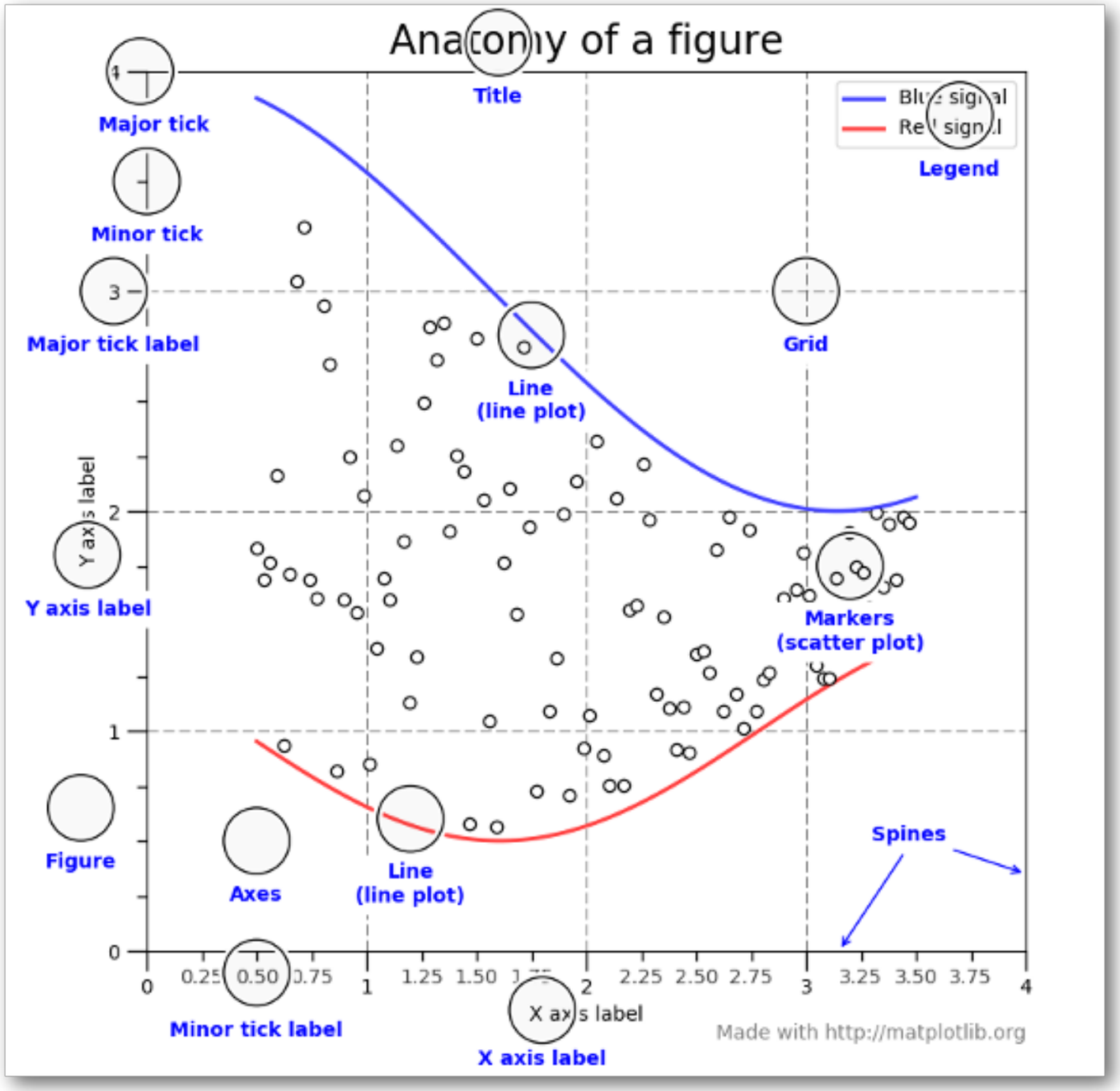
https://matplotlib.org/3.1.0/

- Matplotlib uses an object oriented structure following an intuitive notation
- Each Axes object contains one or more Axis objects.
- A Figure is a set of one or more Axes.
- Each Axes is associated with exactly one Figure and each set of Markers is associated with exactly one Axes.
- In other words, Markers/Lines represent a dataset that is plotted against one or more Axis. An Axes object is (effectively) a subplot of a Figure.

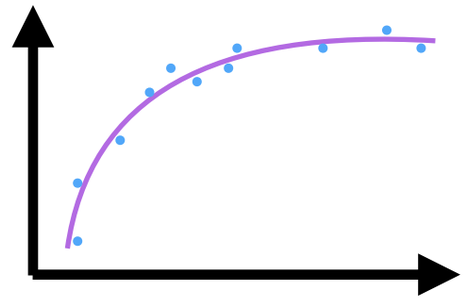
Basic Plotting - Programmatically!



https://matplotlib.org/3.1.0/



Basic Plotting - Programmatically!

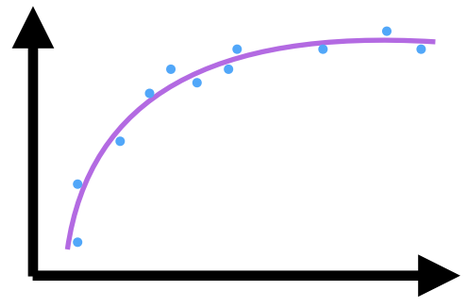


https://matplotlib.org/3.1.0/

- While the Figure object controls the way in which the figure is displayed.
  - .gca() - Get the current Axes, creating one if necessary
  - .show() - Show the final figure
  - .savefig("filename.ext", dpi=300) - Save the figure to "filename.ext" where ".ext" defines the format the saved image

```
filetypes = {'ps': 'Postscript', 'eps': 'Encapsulated Postscript', 'pdf': 'Portable Document Format',
'pgf': 'PGF code for LaTeX', 'png': 'Portable Network Graphics', 'raw': 'Raw RGBA bitmap', 'rgba': 'Raw
RGBA bitmap', 'svg': 'Scalable Vector Graphics', 'svgz': 'Scalable Vector Graphics', 'jpg': 'Joint
Photographic Experts Group', 'jpeg': 'Joint Photographic Experts Group', 'tif': 'Tagged Image File
Format', 'tiff': 'Tagged Image File Format'}
```

Basic Plotting - Programmatically!



https://matplotlib.org/3.1.0/

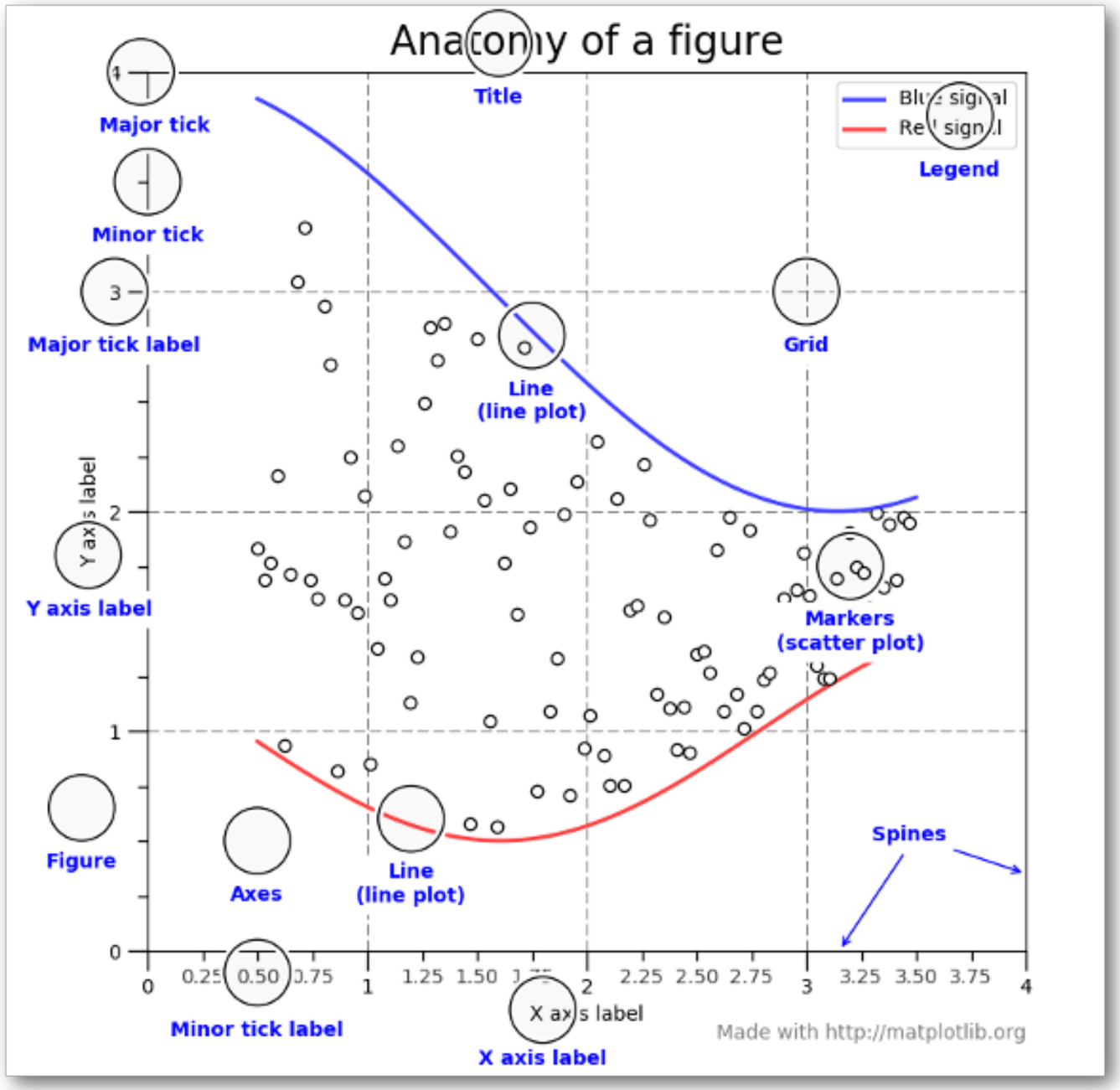
- The first step is to import the pyplot module from matplotlib and instanciating a Figure object:

```
import matplotlib.pyplot as plt
fig = plt.figure()
```

- The convention is to import pyplot as plt
- To create subplots (Axes) you use .subplots(nrows, ncols, sharex=False, sharey=False) instead of .figure(). set sharex and/or sharey to True to keep the same scale in both cases.
- .subplots - returns a (fig, ax_lst) tuple where ax_lst is a list of Axes and fig is the Figure.
- Axes have several methods of interest:
  - .plot(x, y) - Make a scatter or line plot from a list of x, y coordinates.
  - .imshow(mat) - Plot a matrix as if it were an image. Element 0,0 is plotted in the top right corner.
  - .bar(x, y) - Make a bar plot where x is a list of the lower left coordinates of each bar and y is the respective height.
  - .pie(values, labels=labels) - Produce a pie plot out of a list of values list and labeled with labels
  - .savefig(filename) - Write the current figure as an static image

matplotlib - decorations

https://matplotlib.org/3.1.0/



- The respective functions are named in an intuitive way, Every Axes object has as methods:
  - .set_xlabel(label)
  - .set_ylabel(label)
  - .set_title(title)
- And axis limits can be set using:
  - .set_xlim(xmin, xmax)
  - .set_ylim(ymin, ymax)
- Tick marks and labels are set using:
  - .set_xticks(ticks)/.set_yticks(ticks)
  - .set_xticklabels(labels)/.set_yticklabels(labels)

matplotlib - Images

https://matplotlib.org/3.1.0/

- plt.imshow(fig) - Display an image on a set of axes.
- plt.imshow(fig, extent=(xllcorner, xurcorner, yllcorner, yurcorner), zorder=-1)
- fig can be any matrix of numbers.
- Further plotting can occur by simply using the functions described above

Matplotlib Basemap

Basemap

https://matplotlib.org/basemap/

- The Basemap module is the workhorse and returns a Basemap object when instantiated.
- The Basemap object has many useful methods to assist in drawing a map:
  - .drawcoastlines() - To draw the coastlines of continents
  - .drawmapboundary() - To draw the boundary of the map
  - .fillcontinents() - add color to the continents
  - etc…
- The constructor for Basemap can take many different arguments to be able to handle different projections, but it defaults to the Plate Carrée projection centered at (0, 0)
- The minimal map is simply:

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt

map = Basemap()
map.drawcoastlines()
plt.savefig('basemap_demo.png')
```

- Please note that with the .drawcoastlines() call nothing is plotted as our map has no content.

Basemap

https://matplotlib.org/basemap/

- We can also visualize just specific regions by setting the bbox and center coordinates by setting

llcrnrlon, llcrnrlat, urcrnrlon, urcrnrlat

- And

lat_0, lon_0

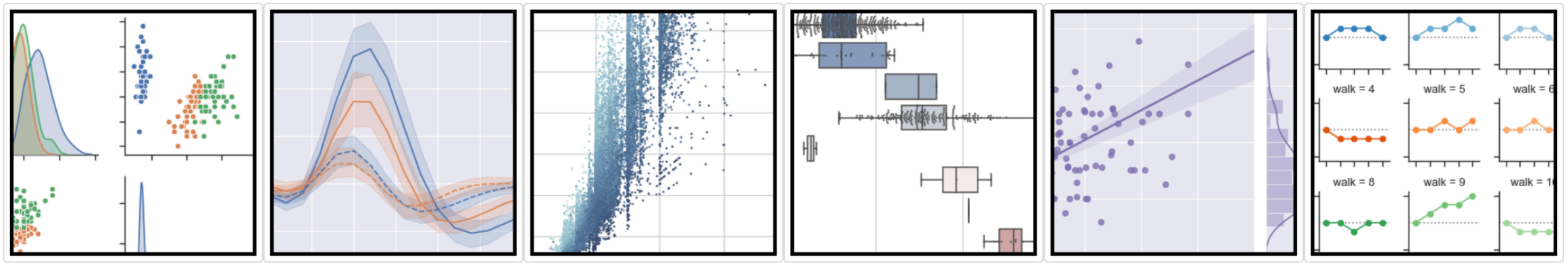
- Respectively.
- We can convert arbitrary lat, lon values to map coordinates by calling the map() object directly.
- After we obtain the map coordinates we can add them to the map by calling the .plot(x, y) method of the map object.

Seaborn

seaborn

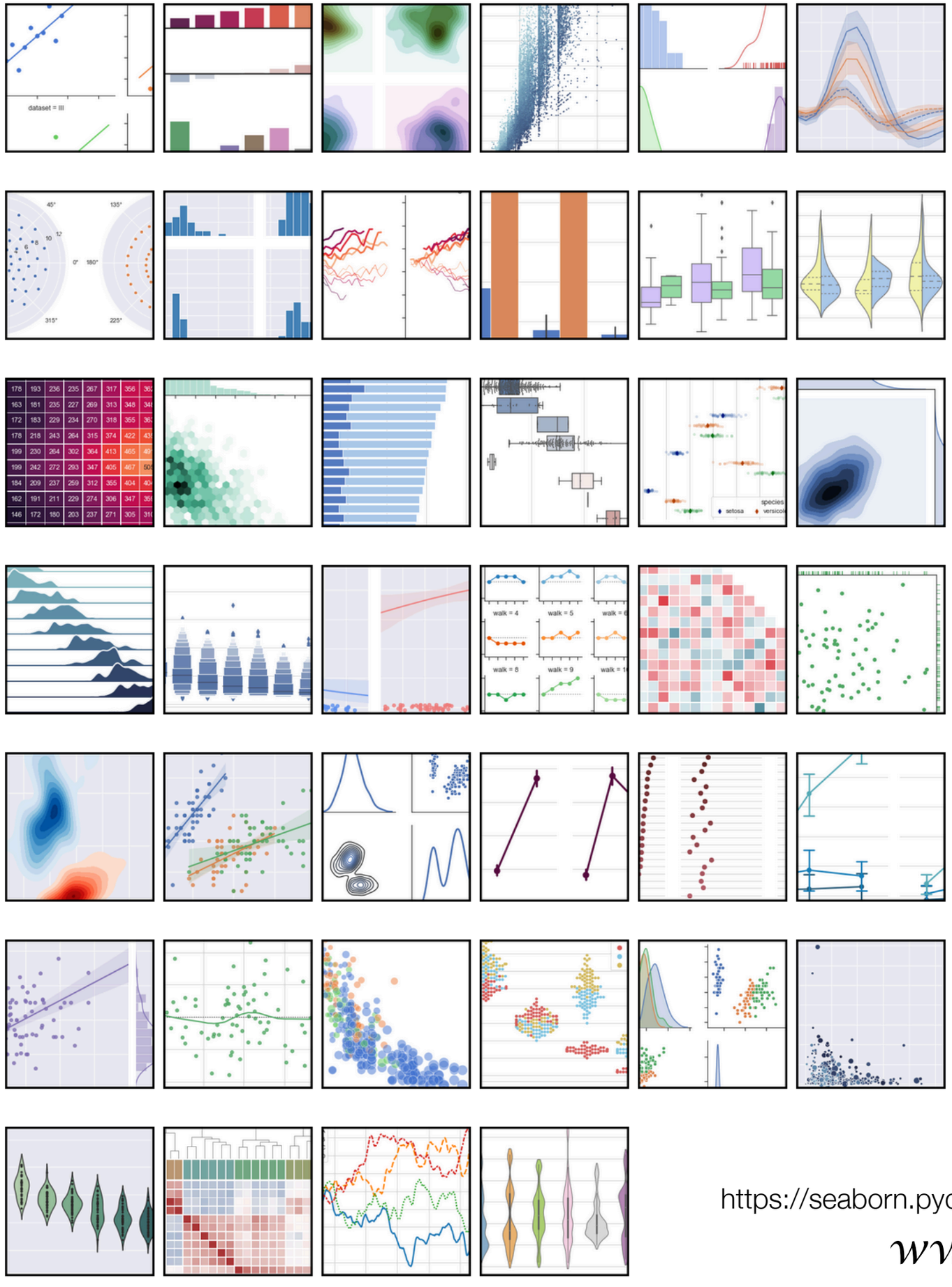
https://pypi.org/project/seaborn/
https://seaborn.pydata.org/index.html

seaborn: statistical data visualization



"Seaborn is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures.

Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots."



https://seaborn.pydata.org/examples/index.html

seaborn

https://seaborn.pydata.org/index.html
https://github.com/mwaskom/seaborn-data

- The first step is to import the seaborn module:

```
import seaborn as sns
```

- The convention is to import seaborn as sns (sns is a geeky reference to Sam Norman Seaborn, a fictional character on the television show The West Wing)
- The sns module contains all the functions we will use as direct members.

seaborn - datasets

https://seaborn.pydata.org/index.html
https://github.com/mwaskom/seaborn-data

- The first step is to import the seaborn module:

```
import seaborn as sns
```

- The convention is to import seaborn as sns (sns is a geeky reference to Sam Norman Seaborn, a fictional character on the television show The West Wing)
- The sns module contains all the functions we will use as direct members.
- For ease of learning and demonstration, seaborn makes it easy to access a small set of standard datasets.
- The datasets can be access easily by calling the load_dataset function with the file name (sans extension) as a parameter:
- .load_dataset(name) - where name is “iris”, “mpg”, etc…
- datasets are returned as "tidy" pandas dataframes

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

- anscombe.csv
- attention.csv
- brain_networks.csv
- car_crashes.csv
- diamonds.csv
- dots.csv
- exercise.csv
- flights.csv
- fmri.csv
- gammas.csv
- iris.csv
- mpg.csv
- planets.csv
- tips.csv
- titanic.csv

seaborn - types of plots

- seaborn can handle a large number of different kinds of plots

Relationship plots

- sns.scatterplot()
- sns.lineplot()

Categorical plots

- sns.stripplot()
- sns.swarmplot()
- sns.boxplot()
- sns.violinplot()
- sns.boxenplot()
- sns.pointplot()
- sns.barplot()
- sns.countplot()

seaborn - types of plots

Axes level	Figure level	
• sns.scatterplot() • sns.lineplot()	Relationship plots sns.relplot()	• kind="scatter" • kind="line"
• sns.stripplot() • sns.swarmplot() • sns.boxplot() • sns.violinplot() • sns.boxenplot() • sns.pointplot() • sns.barplot() • sns.countplot()	Categorical plots sns.catplot()	• kind="strip" • kind="swarm" • kind="box" • kind="violin" • kind="boxen" • kind="point" • kind="bar" • kind="count"

Figure level functions are equivalent to the axes level ones, (with the right kind setting) but more adapted for ease of exploration

For convenience, we will focus on figure level functions

seaborn - basic function structure

- The basic plotting functions are designed to work directly with pandas data frames
- Every plotting function takes a data parameter that is used to pass the correct data frame to the function
- There is no limitation on the number of columns that the dataframe can contain.
- The specific columns to be plotted are passed by setting other parameters to the respective column names. In particular:
    - x - column to plot along the x-axis
    - y - column to plot along the y axis
- Plot properties such as hue, size, style, etc can also be set using column names
- In the case of figure level functions, the kind parameter determines what specific type of plot is produced.

seaborn - Figure level functions

- sns.relplot(x, y, hue, size, style, data, kind)
  - x, y - column names to plot in each axes
  - hue - column name specifying how to color each data element
  - size - column name to use to determine the size of each element
  - style - column name to use to determine the style of each element
  - data - dataframe containing the data to plot
  - kind - string specifying the type of plot to produce
- sns.catplot(x, y, hue, order, data, orient, kind)
  - orient - specifies the orientation of the plot ('v' or 'h' for vertical or horizontal)
  - order - specifies the order in which the categorical variable (x or y) is plotted
  - *_order - family of parameters that determiner the order in which the respective categorical value (hue_order, style_order, size_order, etc…) is plotted

seaborn - Figure level functions

- As we saw, axes level and figure level functions are equivalent. Every plot you can generate with an axes level function can also be generated by a figure level function
- The converse is, however, not true.
- One of the main advantages of figure level functions is their ability to easily generate subplots by just passing a couple of extra parameters:
    - row, col - specifies the column names to be used to split the dataset and plot along rows and columns
    - col_wrap - width at which to wrap the column. Incompatible with a row setting.
    - row_order, col_order - work similarly to the other *_order parameters
- The return types of **axes level** functions and **figure level** function are also different:
    - axes level functions return a matplotlib Axis object
    - figure level functions return a seaborn FacetGrid object
- Axes level functions can be easily added to a pre-existing matplotlib plot by passing an matplotlib Axis object to the ax parameter.

FacetGrid

- FacetGrid is the object that seaborn uses in the background to generate the subplots.

- You can easily use FacetGrid with any matplotlib plotting function.

- The process is divided into two steps:

  - First, you instantiate a FacetGrid object with the correct data frame and basic plotting parameters (these are the same parameters you would use with any other seaborn function).

  - Second, you use the FacetGrid.map method to initialize the FacetGrid object with the correct matplotlib function along with any extra matplotlib parameters you require.

https://github.com/DataForScience/DataViz

Events

Natural Language Processing (NLP) from Scratch

Jun 29, 2019 - ODSC NYC

Deep Learning from Scratch

Jul 2, 2019 5am-9pm (PST)

Natural Language Processing (NLP) from Scratch

Aug 5, 2019 5am-9pm (PST)

Deep Learning from Scratch

Sept 24, 2019 - Strata NYC

Natural Language Processing (NLP) from Scratch

http://bit.ly/LiveLessonNLP - On Demand