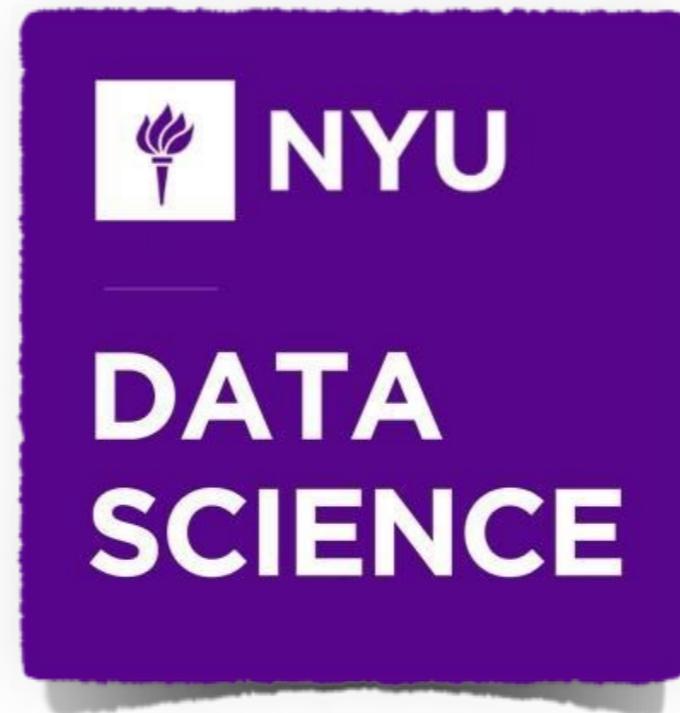


# Lecture V - Scientific Visualization

---

Bruno Gonçalves  
[www.bgoncalves.com](http://www.bgoncalves.com)





# Twitter places

- As we saw last week, Twitter defines a “coordinates” field in tweets
- There is also a “place” field that we glossed over.
- The **place** object contains also geographical information, but at a coarser resolution than the **coordinates** field.
- Each place has a unique **place\_id**, a **bounding\_box** and some geographical information, such as **country** and **full\_name**:

```
{'attributes': {},  
 'bounding_box': {'coordinates': [[[[-74.041878, 40.570842],  
 [-74.041878, 40.739434],  
 [-73.855673, 40.739434],  
 [-73.855673, 40.570842]]],  
 'type': 'Polygon'},  
 'country': 'United States',  
 'country_code': 'US',  
 'full_name': 'Brooklyn, NY',  
 'id': '011add077f4d2da3',  
 'name': 'Brooklyn',  
 'place_type': 'city',  
 'url': 'https://api.twitter.com/1.1/geo/id/011add077f4d2da3.json'}
```

The bounding\_box field is GeoJSON formatted and compatible with `pyshp.shape`

- places can be of several different types: ‘admin’, ‘city’, ‘neighborhood’, ‘poi’

# Twitter places

<https://dev.twitter.com/overview/api/places>

## Place Attributes

Place Attributes are metadata about places. An attribute is a key-value pair of arbitrary strings, but with some conventions.

Below are a number of well-known place attributes which may, or may not exist in the returned data. These attributes are provided when the place was created in the Twitter places database.

Key	Description
street_address	
locality	the city the place is in
region	the administrative region the place is in
iso3	the country code
postal_code	in the preferred local format for the place
phone	in the preferred local format for the place, include long distance code
twitter	twitter screen-name, without @
url	official/canonical URL for place
app:id	An ID or comma separated list of IDs representing the place in the applications place database.

Keys can be no longer than 140 characters in length. Values are unicode strings and are restricted to 2000 characters.

# Challenge - Filter points and places

---

- Load each tweet in **NYC.json.gz** file by using:

```
tweet = eval(line.strip())
```

- on each line and write to **Manhattan\_places.json.gz** all the tweets within Manhattan, as defined by

`geofiles/nybb_15c/nybb_wgs84.shp`

- but now check if the centroid of the '**place**' object '**bounding\_box**' is within Manhattan as well

# Challenge - Filter points and places

```
import sys
import shapefile
from shapely.geometry import shape, Point
import gzip

shp = shapefile.Reader('geofiles/nybb_15c/nybb_wgs84.shp')

recordDict = dict(zip([record[1] for record in shp.iterRecords()], range(shp.numRecords)))

manhattan = shape(shp.shape(recordDict["Manhattan"]))

fp = gzip.open("Manhattan_places.json.gz", "w")

for line in gzip.open("NYC.json.gz"):
    try:
        tweet = eval(line.strip())
        point = None

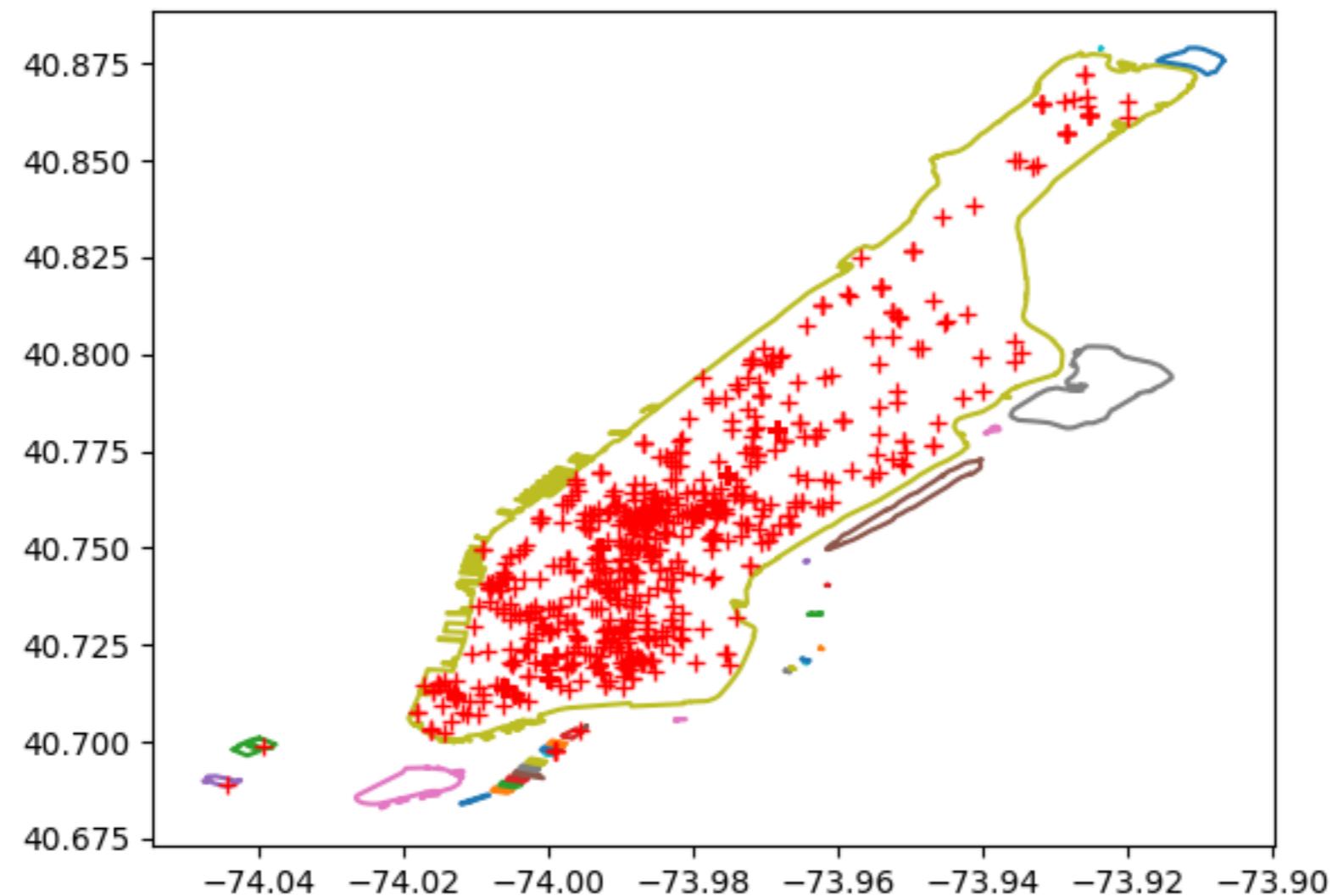
        if "coordinates" in tweet and tweet["coordinates"] is not None:
            point = Point(tweet["coordinates"]["coordinates"])
        else:
            if "place" in tweet and tweet["place"]["bounding_box"] is not None:
                bbox = shape(tweet["place"]["bounding_box"])
                point = bbox.centroid

        if point is not None and manhattan.contains(point):
            fp.write(line)
    except:
        pass

fp.close()
```

# Challenge - Filter points and places

---



# Challenge - Filter points and places

```
import sys
import gzip
import numpy as np
import shapefile
from shapely.geometry import shape, Point
import matplotlib.pyplot as plt

shp = shapefile.Reader('geofiles/nybb_15c/nybb_wgs84.shp')
recordDict = dict(zip([record[1] for record in shp.iterRecords()],
range(shp.numRecords)))

manhattan = shp.shape(recordDict["Manhattan"])

points = np.array(manhattan.points)
parts = manhattan.parts
parts.append(len(manhattan.points))

for i in range(len(parts)-1):
    plt.plot(points.T[0][parts[i]:parts[i+1]], points.T[1]
[parts[i]:parts[i+1]])

points_X = []
points_Y = []

for line in gzip.open(sys.argv[1]):
    try:
        tweet = eval(line.strip())
        point = None

        if "coordinates" in tweet and tweet["coordinates"] is not None:
            point = Point(tweet["coordinates"]["coordinates"])
        else:
            if "place" in tweet and tweet["place"]["bounding_box"] is not
None:
                bbox = shape(tweet["place"]["bounding_box"])
                point = bbox.centroid

        if point is not None:
            points_X.append(point.x)
            points_Y.append(point.y)
    except:
        pass

plt.plot(points_X, points_Y, 'r+')

plt.savefig(sys.argv[1] + '.png')
```

# Calculating distances

[https://en.wikipedia.org/wiki/Vincenty%27s\\_formulae](https://en.wikipedia.org/wiki/Vincenty%27s_formulae)  
[https://en.wikipedia.org/wiki/Great-circle\\_distance](https://en.wikipedia.org/wiki/Great-circle_distance)  
[https://en.wikipedia.org/wiki/Haversine\\_formula](https://en.wikipedia.org/wiki/Haversine_formula)

---

- Earlier we saw how to obtain the distance between two points using the Google Maps API.
- But what is the shortest distance between two **arbitrary** points on the surface of the Earth?
- This depends strongly on our model of the Earth:
  - **Great Circle** - Assumes that the Earth is a perfect sphere of a given radius
  - Usually uses the **Haversine** formula  $\Delta\sigma = 2 \arcsin \sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos \phi_1 \cdot \cos \phi_2 \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)}$
  - **Vincenty** - Uses a (more) accurate ellipsoid model of the Earth

# geopy

<https://geopy.readthedocs.io/en/1.10.0/>

- **geopy** provides two different types of functionality
  - **geopy.geocoders** - a unified interface to several geocoding services (Google Maps, Nominatim, Yahoo, Bing, etc...)
  - **geopy.distance** - state of the art distance calculations
- We will focus just on the **distance** module:
  - **distance.vincenty(p1, p2)** - Calculate the vincenty distance between **p1** and **p2**
  - **distance.great\_circle(p1, p2)** - Calculate the great circle distance between **p1** and **p2**
  - **distance.distance(p1, p2)** - an alias to **distance.vincenty** to be used as a default.

- all `distance` functions return a `Distance` object.
- the `Distance` object provides properties that represent the result in different units:
  - `.km/.kilometers`
  - `.m/.meters`
  - `.mi/.miles`
  - `.ft/.feet`
  - `.nm/.nautical`
- it also allows us to recalculate the result using different ellipsoids:

- `.set_ellipsoid('ellipsoid')`

- by default `WGS-84` is used.

```
model           major (km)   minor (km)   flattening
ELLIPSOIDS = { 'WGS-84':      (6378.137,    6356.7523142,  1 / 
                           'GRS-80':      (6378.137,    6356.7523141,  1 / 
                           'Airy (1830)': (6377.563396,  6356.256909,  1 / 
                           'Intl 1924':   (6378.388,    6356.911946,  1 / 297.0), 
                           'Clarke (1880)':(6378.249145, 6356.51486955, 1 / 293.465),
                           'GRS-67':       (6378.1600,   6356.774719,  1 / 298.25),
                           }
```

# Challenge - geopy

<https://geopy.readthedocs.io/en/1.10.0/>

- Calculate the distance between

`p1 = (41.49008, -71.312796)`

`p2 = (41.499498, -81.695391)`

- in meters, using the `vincenty` and `great_circle` functions.

# Challenge - geopy

<https://geopy.readthedocs.io/en/1.10.0/>

- Calculate the distance between

p1 = (41.49008, -71.312796)  
p2 = (41.499498, -81.695391)

- in meters, using the `vincenty` and `great_circle` functions.

```
from geopy import distance

p1 = (41.49008, -71.312796)
p2 = (41.499498, -81.695391)

dist_vincenty = distance.vincenty(p1,
p2).meters
dist_great = distance.great_circle(p1,
p2).meters

print("Vincenty:", dist_vincenty)
print("Great Circles:", dist_great)
```

# GIS Data Systems

---

## Vector

---

- Data types:
  - Points
  - Lines
  - Polygons
- Discrete shapes and boundaries
- Spatial, Database and Network analysis
- Shapefile, GeoJSON, GML, etc...

# GIS Data Systems

## Vector

- Data types:
  - Points
  - Lines
  - Polygons
- Discrete shapes and boundaries
- Spatial, Database and Network analysis
- Shapefile, GeoJSON, GML, etc...

## Raster

- Data types:
  - Cells
  - Pixels
  - Elements
- Dense data, Continuous surfaces
- Spatial Analysis and modeling
- GeoTIFF, ASC, JPEG2000, etc...

# ASCII Grid

---

- Perhaps the simplest raster file

- ASCII text based

- A small header

```
ncols          246
nrows          119
xllcorner     -126.50000000000
yllcorner      22.750000000000
cellsize       0.250000000000
NODATA_value   -9999
```

- Followed by rows of numbers

- Very convenient to Read and Write

# Challenge - ASCII Grid

---

- Write a simple function to load the file:

geofiles/US\_pop.asc

- into a numpy array and print the value corresponding to:

40.730503, -74.243251

# Challenge

```
import numpy as np
import matplotlib.pyplot as plt

def map_points(xllcorner, yllcorner, cellsize, nrows, x, y):
    x = int((x-xllcorner)/cellsize)
    y = (nrows-1)-int((y-yllcorner)/cellsize)

    return x, y

fp = open("geofiles/US_pop.asc")
ncols, count = fp.readline().split()
ncols = int(count)
nrows, count = fp.readline().split()
nrows = int(count)
xllcorner, value = fp.readline().split()
xllcorner = float(value)
yllcorner, value = fp.readline().split()
yllcorner = float(value)
cellsize, value = fp.readline().split()
cellsize = float(value)

NODATA_value, value = fp.readline().split()
NODATA_value = float(value)

data = []
for line in fp:
    fields = line.strip().split()
    data.append([float(field) for field in fields])

data = np.array(data)
data[data==NODATA_value] = 0

x = -74.243251
y = 40.730503

coord_x, coord_y = map_points(xllcorner, yllcorner, cellsize, nrows, x, y)
print(data[coord_y, coord_x])
```

# Challenge

```
import numpy as np
import matplotlib.pyplot as plt

def map_points(xllcorner, yllcorner, cellsize, nrows, x, y):
    x = int((x-xllcorner)/cellsize)
    y = (nrows-1)-int((y-yllcorner)/cellsize)

    return x, y

fp = open("geofiles/US_pop.asc")
ncols, count = fp.readline().split()
ncols = int(count)
nrows, count = fp.readline().split()
nrows = int(count)
xllcorner, value = fp.readline().split()
xllcorner = float(value)
yllcorner, value = fp.readline().split()
yllcorner = float(value)
cellsize, value = fp.readline().split()
cellsize = float(value)

NODATA_value, value = fp.readline().split()
NODATA_value = float(value)

data = []
for line in fp:
    fields = line.strip().split()
    data.append([float(field) for field in fields])

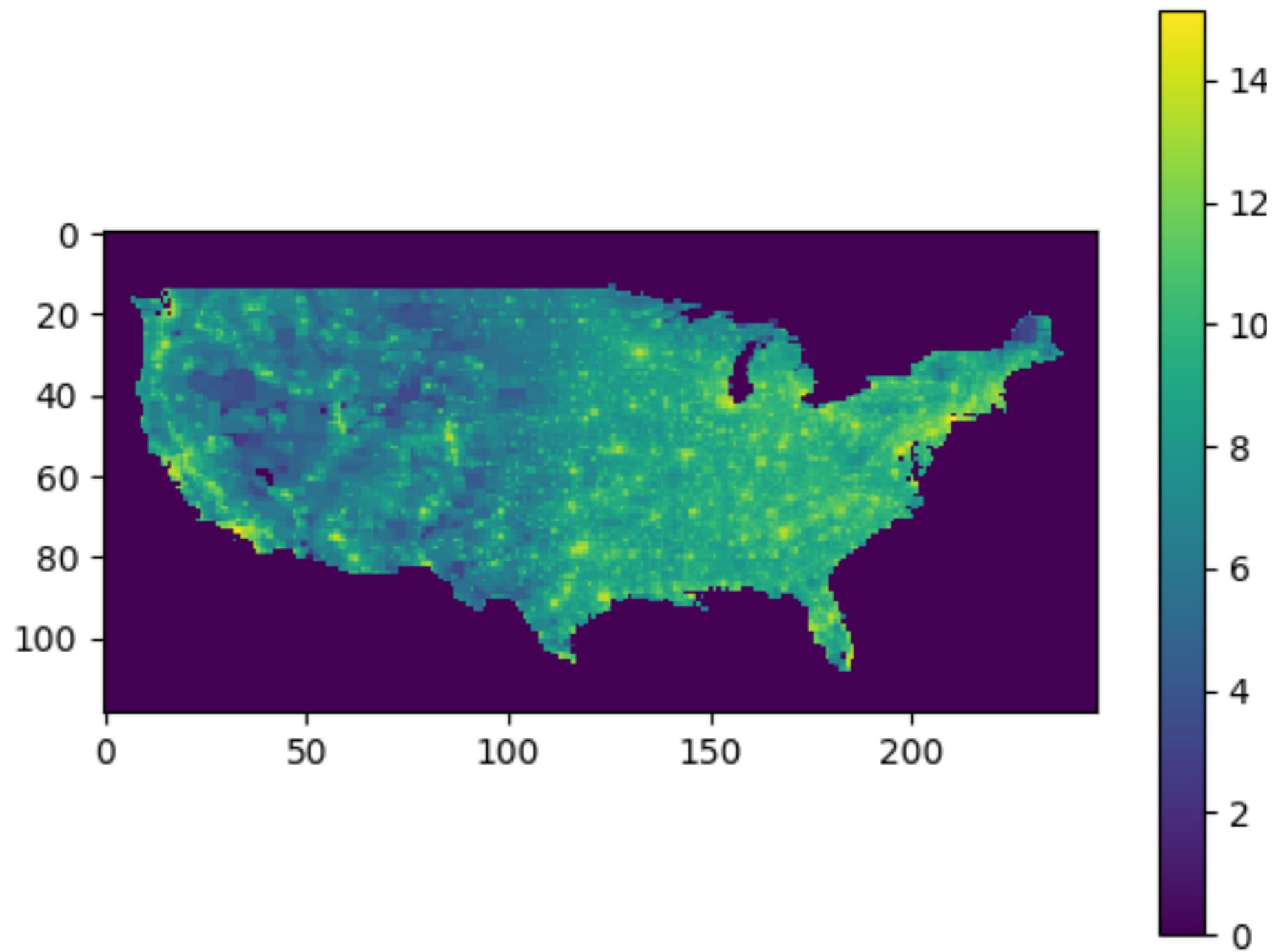
data = np.array(data)
data[data==NODATA_value] = 0

x = -74.243251
y = 40.730503

coord_x, coord_y = map_points(xllcorner, yllcorner, cellsize, nrows, x, y)
print(data[coord_y, coord_x])
```

# Challenge - ASCII Grid

---



# ASCII Grid

- This type of grid is a very convenient way to aggregate spatial data.
- Simply map **lat, lon** pairs to matrix entries and then increment the values
- All we need is to define the **bbox** we are interested in, and the size of each cell and create a matrix with that shape.

```
import numpy as np
from shapely.geometry import shape, Point
import shapefile

shp = shapefile.Reader('../Lecture IV/geofiles/nybb_15c/nybb_wgs84.shp')
recordDict = dict(zip([record[1] for record in shp.iterRecords()],
range(shp.numRecords)))

manhattan = shp.shape(recordDict["Manhattan"])

xllcorner, yllcorner, xurcorner, yurcorner = manhattan.bbox

cellsize = 0.01

ncols = int((xurcorner-xllcorner)/cellsize)
nrows = int((yurcorner-yllcorner)/cellsize)

data = np.zeros((nrows, ncols), dtype='int')

@bgoncalv print(data.shape)
```

generate\_matrix.py

# Challenge - Aggregate

---

- Rewrite

`shapefile_filter_places.py`

- so that it now counts how many tweets happen in each cell of width **0.01**

# Challenge - Aggregate

```
import sys
import numpy as np
import shapefile
from shapely.geometry import shape, Point
import matplotlib.pyplot as plt
import gzip

def map_points(xllcorner, yllcorner, cellsize, nrows, x, y):
    x = int((x-xllcorner)/cellszie)
    y = (nrows-1)-int((y-yllcorner)/cellszie)

    return x, y

def save_asc(data, xllcorner, yllcorner, cellszie, filename):
    fp = open(filename, "w")

    nrows, ncols = data.shape

    print("ncols", ncols, file=fp)
    print("nrows", nrows, file=fp)
    print("xllcorner", xllcorner, file=fp)
    print("yllcorner", yllcorner, file=fp)
    print("cellsize", cellszie, file=fp)
    print("NODATA_value", "-9999", file=fp)

    for i in range(nrows):
        for j in range(ncols):
            print("%u " % data[i, j]), end="", file=fp)

        print("\n", end="", file=fp)

    fp.close()
```

@bgoncalves

shapefile\_filter\_aggregate.py

```

shp = shapefile.Reader('../Lecture IV/geofiles/nybb_15c/nybb_wgs84.shp')
recordDict = dict(zip([record[1] for record in shp.iterRecords()], range(shp.numRecords)))
manhattan = shape(shp.shape(recordDict["Manhattan"]))

xllcorner, yllcorner, xurcorner, yurcorner = manhattan.bounds
cellsize = 0.01

ncols = int((xurcorner-xllcorner)/cellsize)
nrows = int((yurcorner-yllcorner)/cellsize)

data = np.zeros((nrows, ncols), dtype='int')

for line in gzip.open("../Lecture IV/NYC.json.gz"):
    try:
        tweet = eval(line.strip())
        point = None

        if "coordinates" in tweet and tweet["coordinates"] is not None:
            point = Point(tweet["coordinates"]["coordinates"])
        else:
            if "place" in tweet and tweet["place"]["bounding_box"] is not None:
                bbox = shape(tweet["place"]["bounding_box"])
                point = bbox.centroid

        if point is not None and manhattan.contains(point):
            coord_x, coord_y = map_points(xllcorner, yllcorner, cellsize, nrows, point.x, point.y)
            data[coord_y, coord_x] += 1

    except:
        pass

save_asc(data, xllcorner, yllcorner, cellsize, "Manhattan.asc")

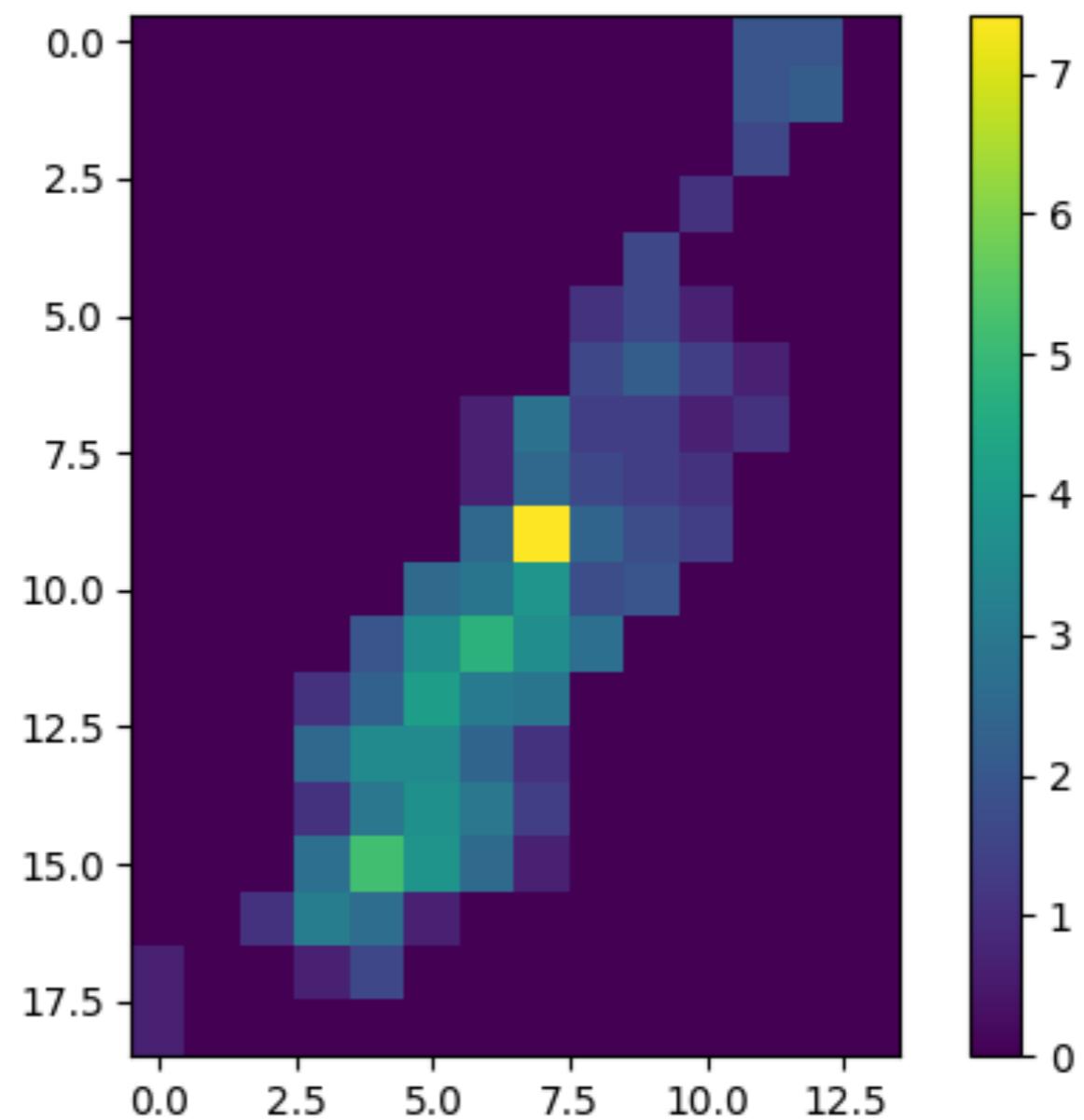
plt.imshow(np.log(data+1))
plt.colorbar()
@6 plt.savefig('Manhattan_cells.png')

```

shapefile\_filter\_aggregate.py

# Challenge - Aggregate

---

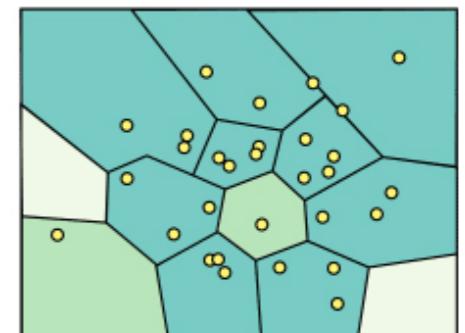
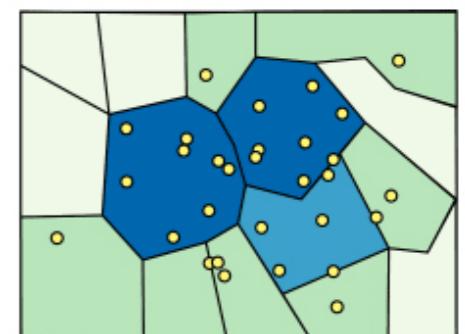
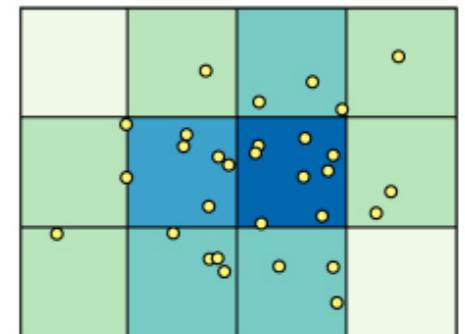
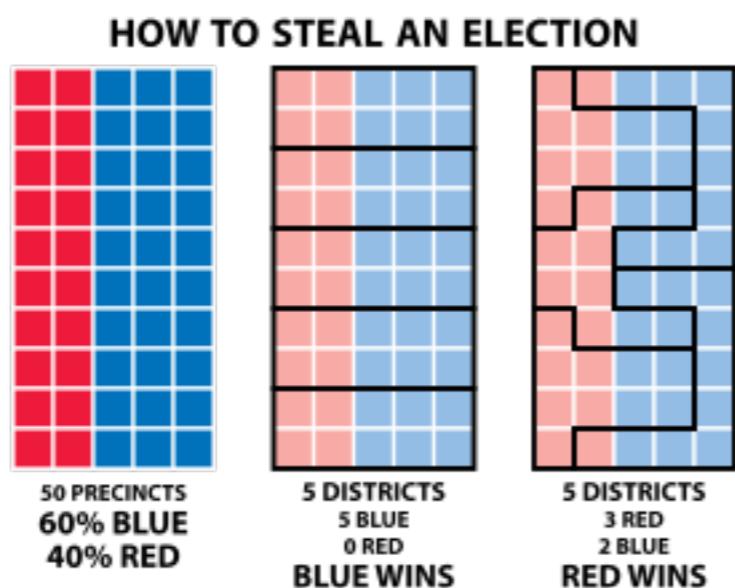


# Modifiable Areal Unit Problem

- Different aggregations can result in differing results.

- The aggregation level must be chosen in a way that is appropriate to the problem and requires experience with the type of dataset and problem.

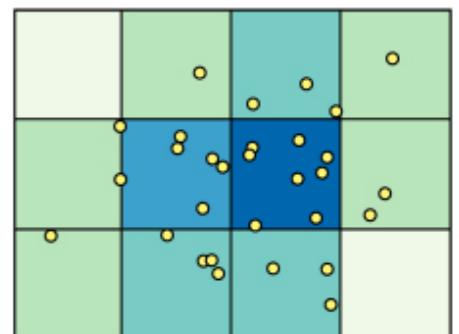
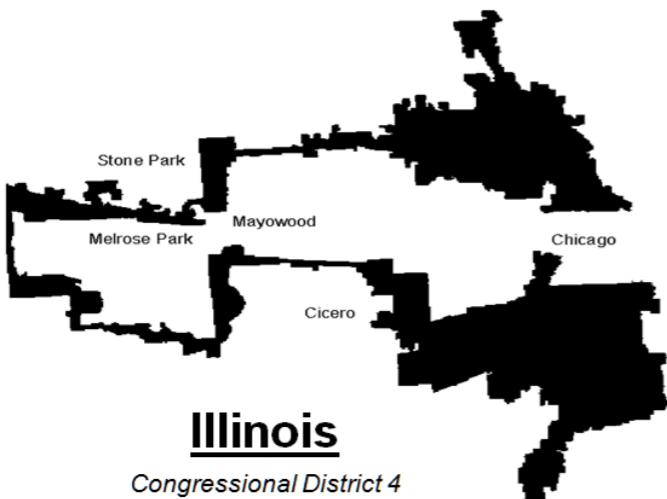
- Sometimes abused for political reasons.



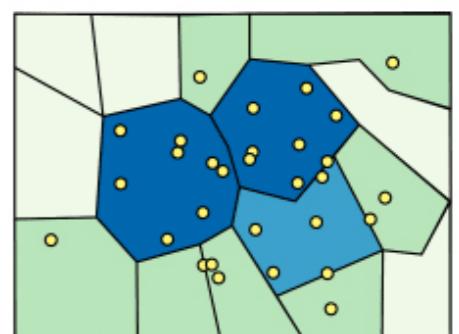
# Modifiable Areal Unit Problem

- Different aggregations can result in differing results.

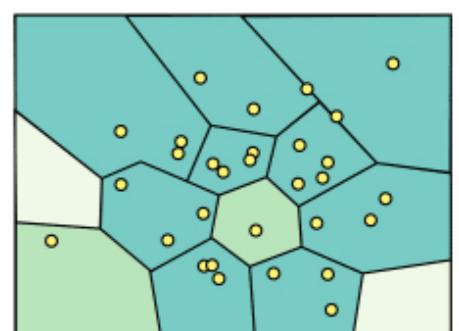
- The aggregation of areas into units for analysis can exacerbate the problem and lead to the Modifiable Areal Unit Problem.



- Sometimes adjacent areas are aggregated into different units.



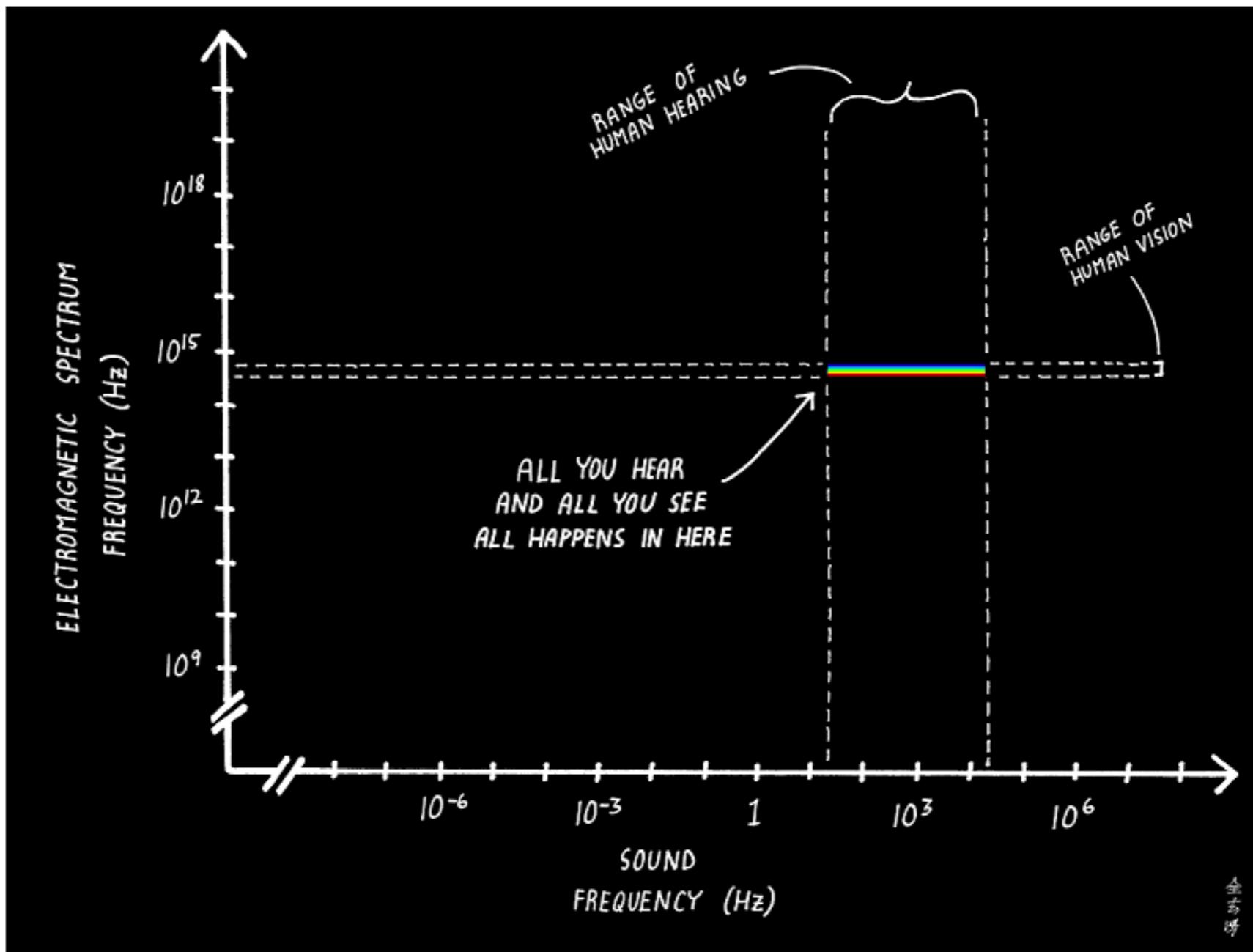
...graphics from 1990s Supreme Court Redistricting Decisions, Peter S. Wattson  
[www.senate.leg.state.mn.us/departments/scr/REDIST/red907.htm](http://www.senate.leg.state.mn.us/departments/scr/REDIST/red907.htm)



# Human Perception

# Human Perception

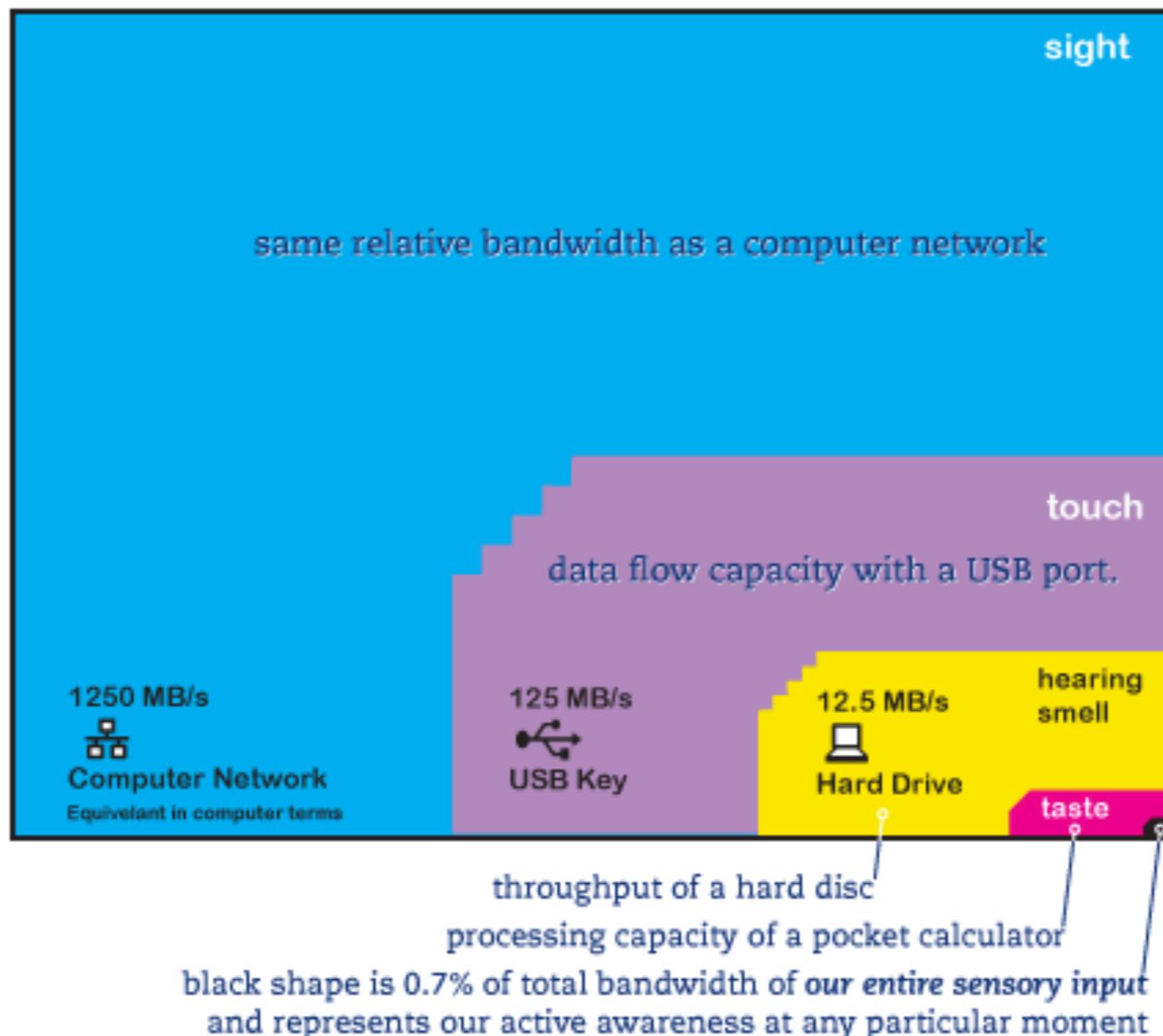
<http://abstrusegoose.com/421>



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

# Human Senses

## NØRRETRANDERS BANDWIDTH OF THE SENSES

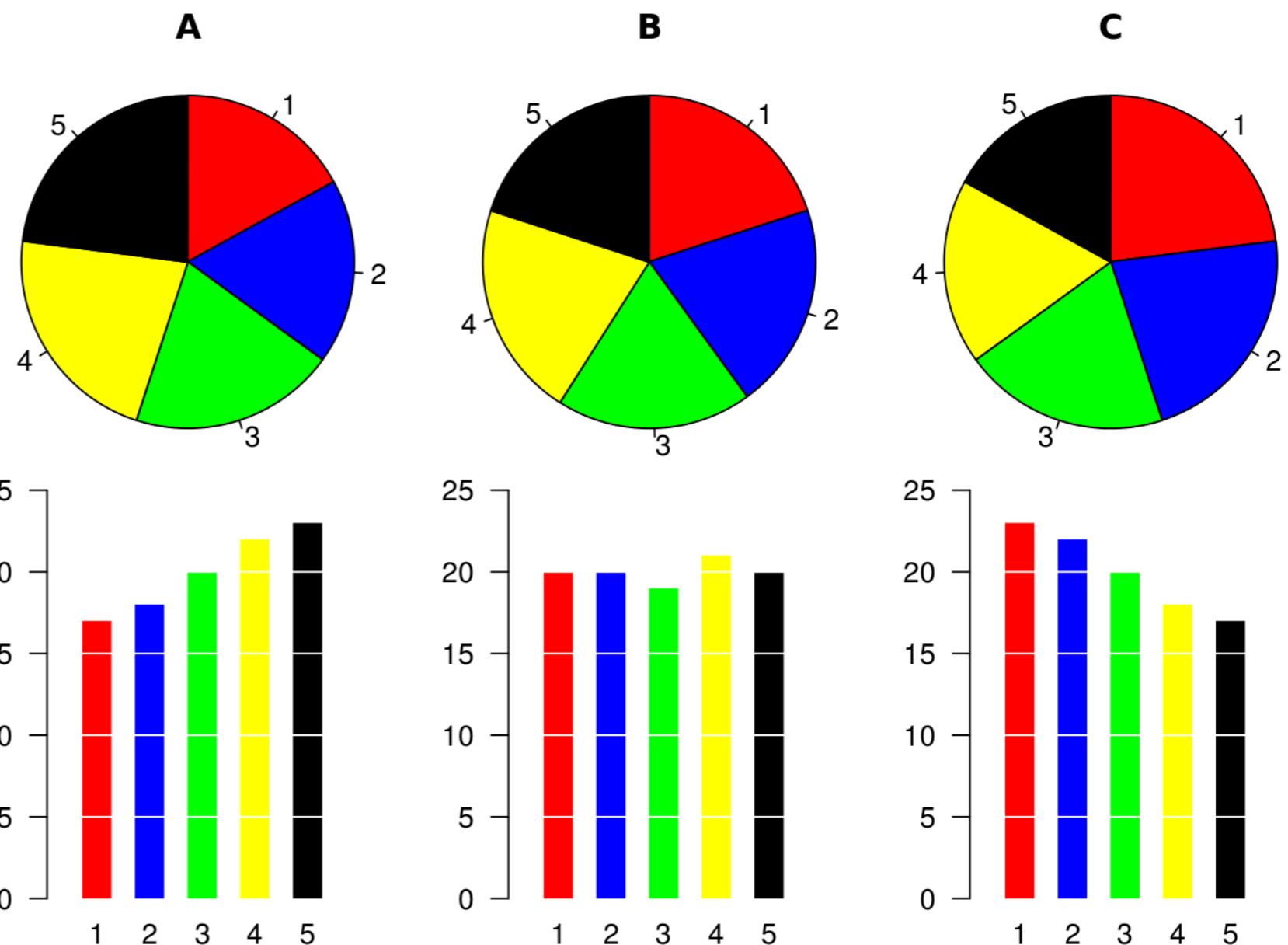


Source: <http://chitownmediapsych.blogspot.ie/2010/09/context-will-be-king-working-title.html>  
©David McCandless : Information Is Beautiful

# Perception

- Some cognitive tasks are significantly easier than others. In order, we are good at distinguishing:

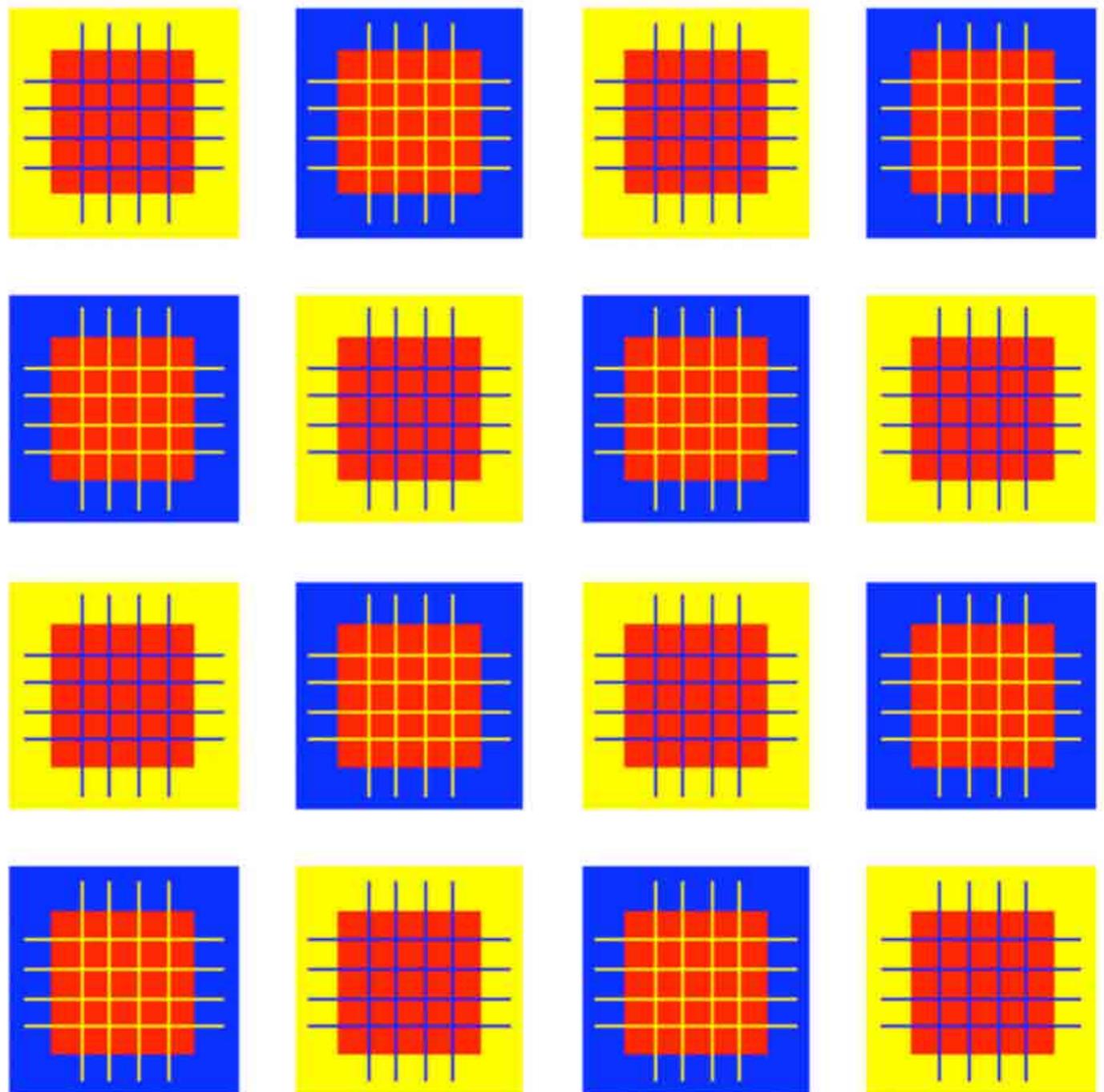
- Position, length
- Direction, Angle, Area
- Volume, Curvature, Shade



# Perception

- Some cognitive tasks are significantly easier than others. In order, we are good at distinguishing:

- Position, length
- Direction, Angle, Area
- Volume, Curvature, Shade
- Color Saturation.



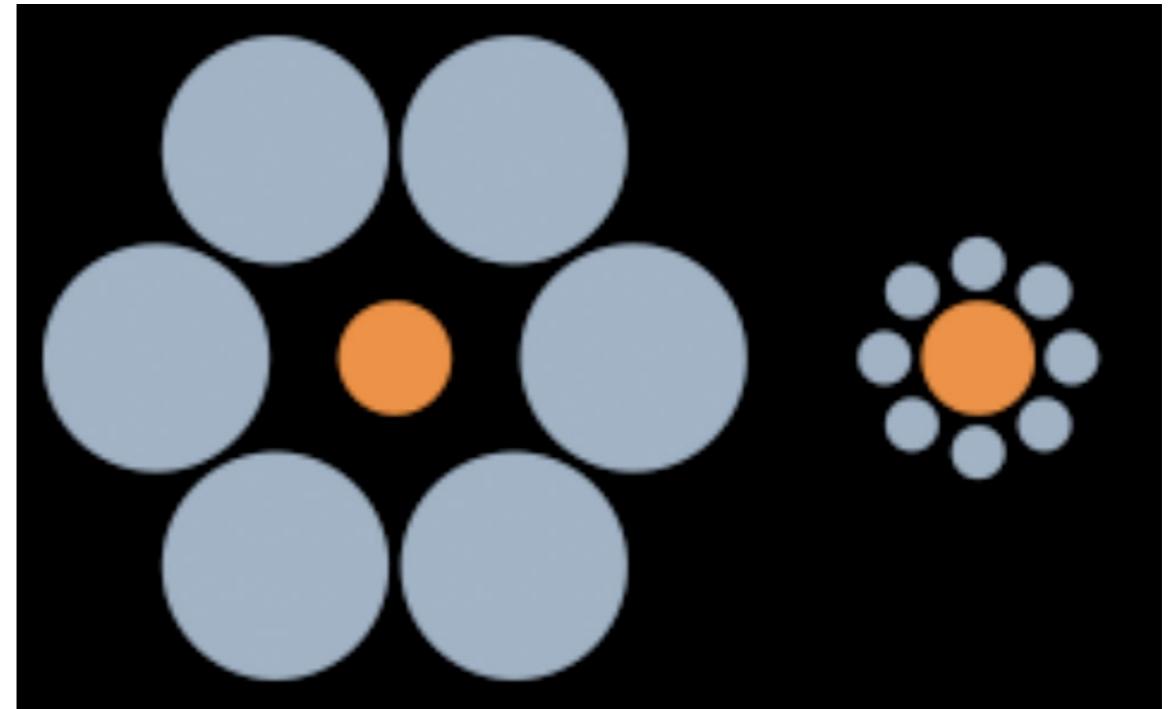
# Perception

---

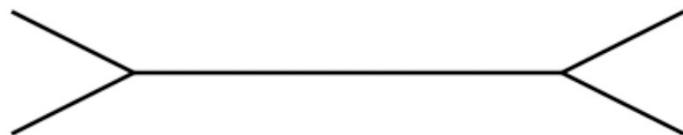
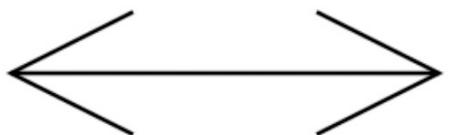
- Some cognitive tasks are significantly easier than others. In order, we are good at distinguishing:

- Position, length
- Direction, Angle, Area
- Volume, Curvature, Shade
- Color Saturation.

- Context also matters!



- An object seen in the context of larger objects will appear smaller, while in the context of smaller objects it will appear larger.

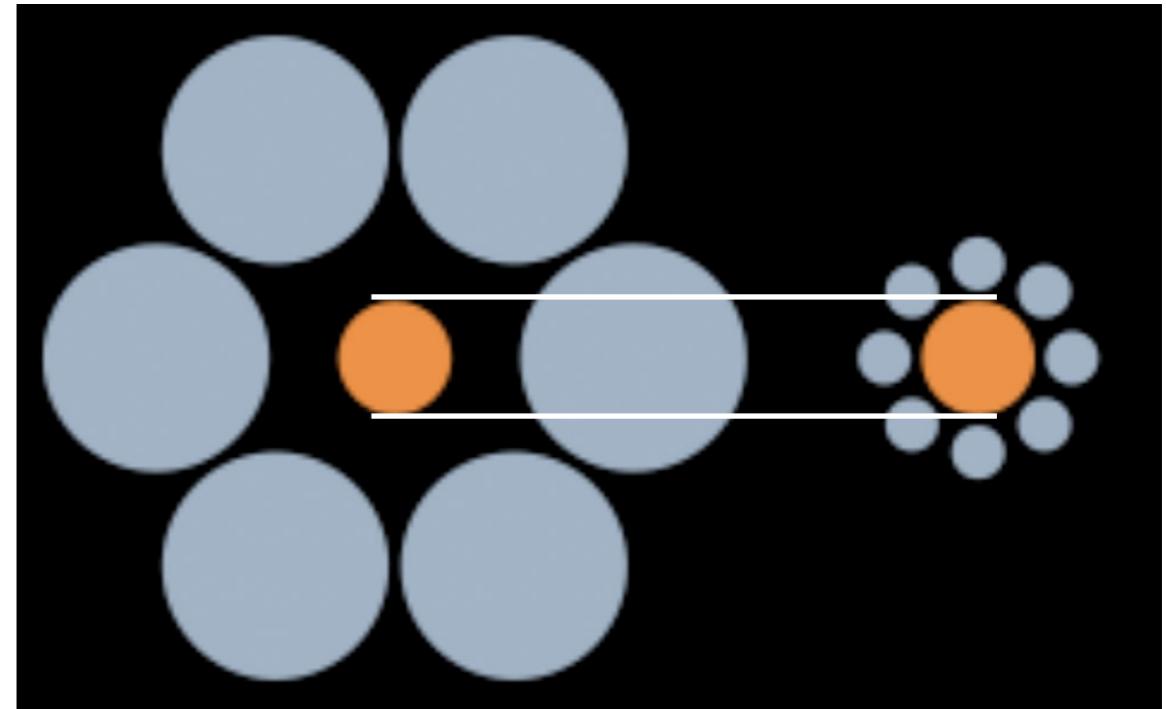


# Perception

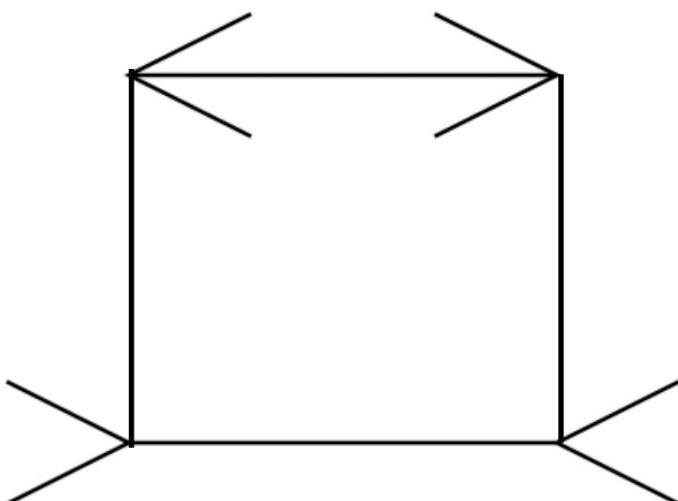
---

- Some cognitive tasks are significantly easier than others. In order, we are good at distinguishing:

- Position, length
- Direction, Angle, Area
- Volume, Curvature, Shade
- Color Saturation.
- Context also matters!



- An object seen in the context of larger objects will appear smaller, while in the context of smaller objects it will appear larger.



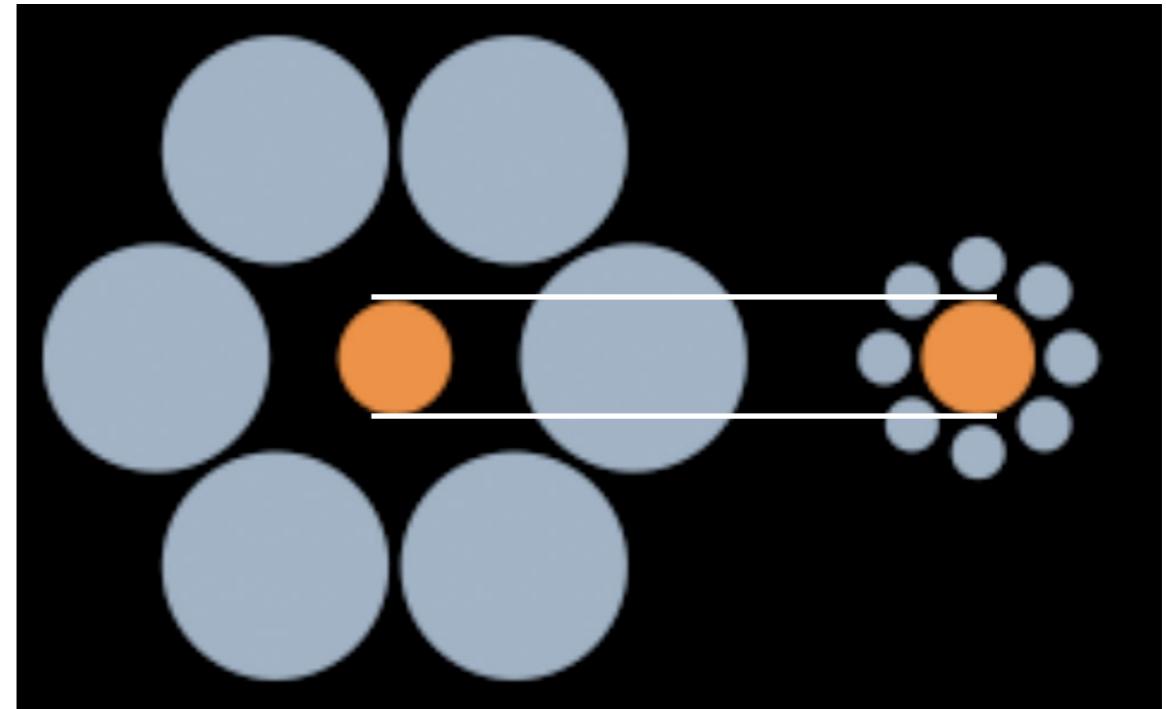
# Perception

---

- Some cognitive tasks are significantly easier than others. In order, we are good at distinguishing:

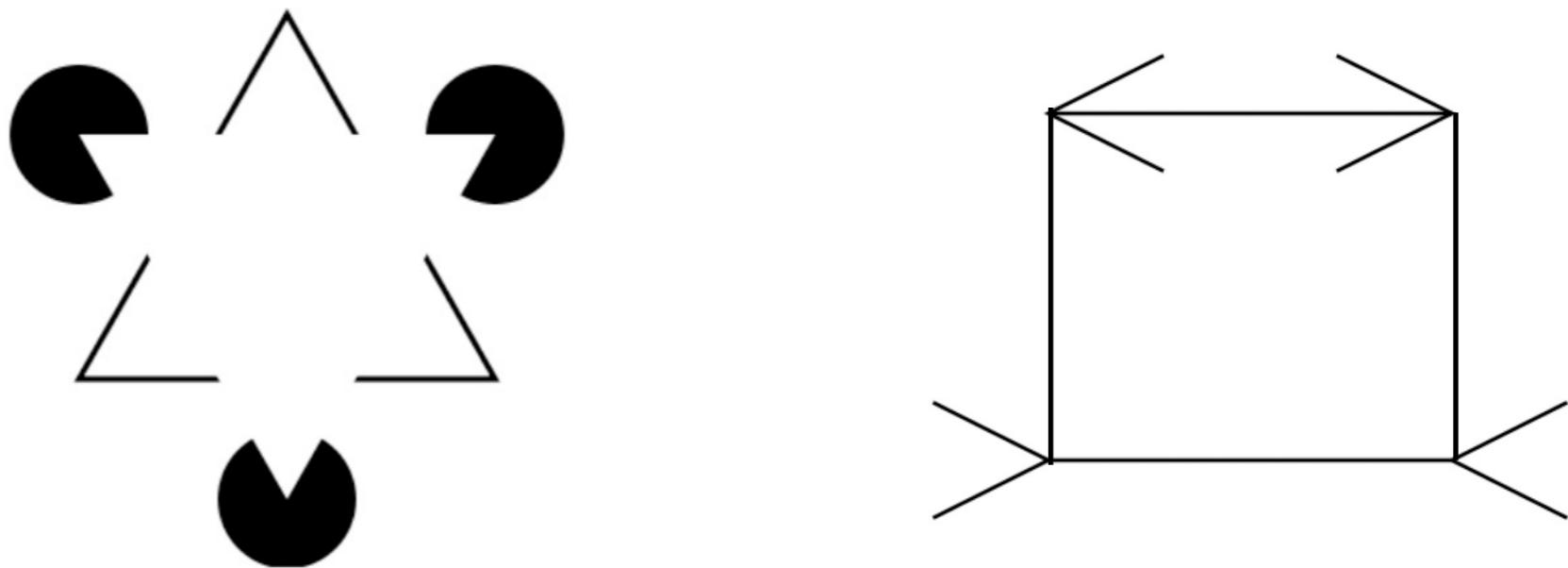
- Position, length
- Direction, Angle, Area
- Volume, Curvature, Shade
- Color Saturation.

- Context also matters!



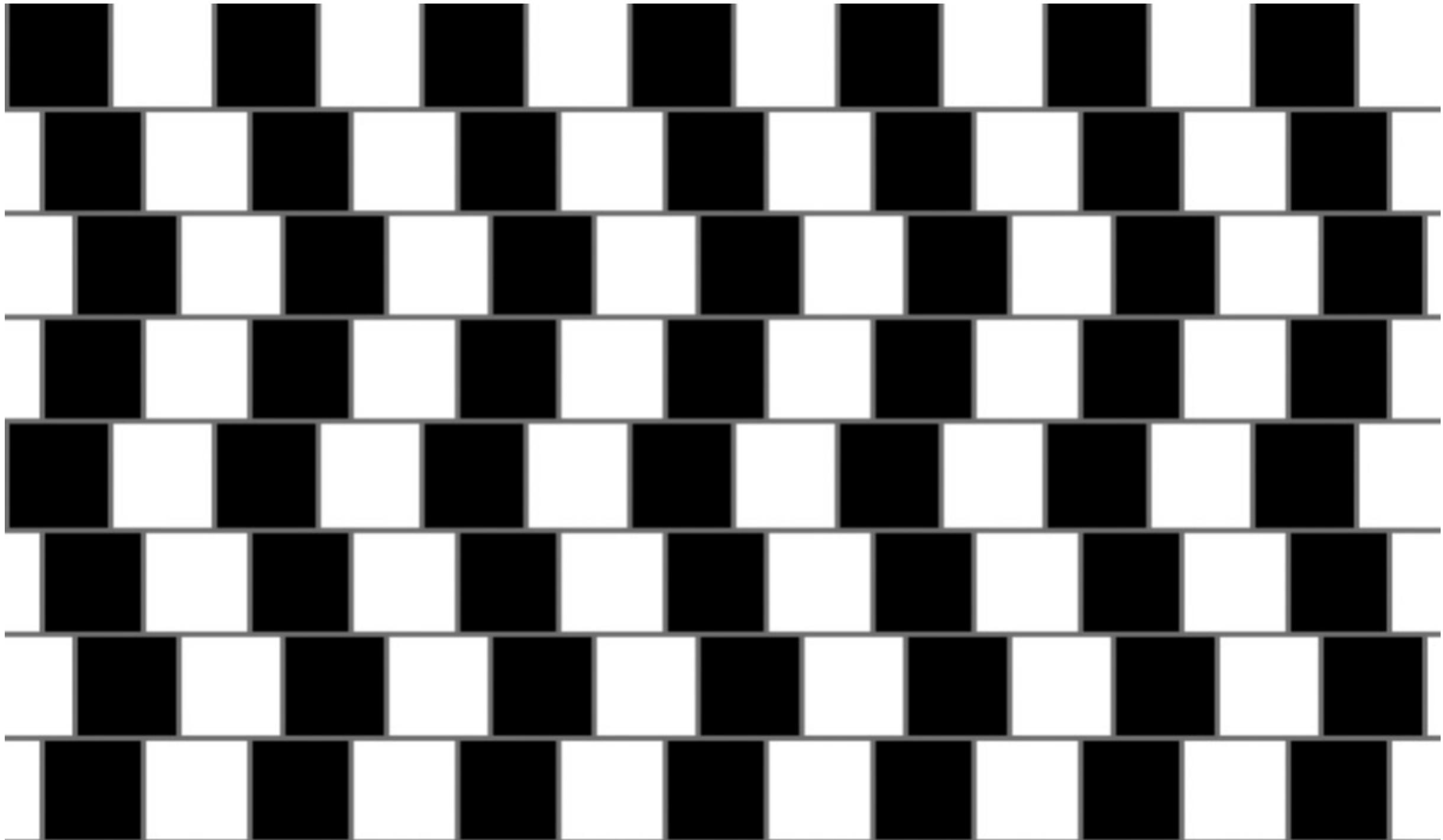
- An object seen in the context of larger objects will appear smaller, while in the context of smaller objects it will appear larger.

- And we "fill in the gaps"



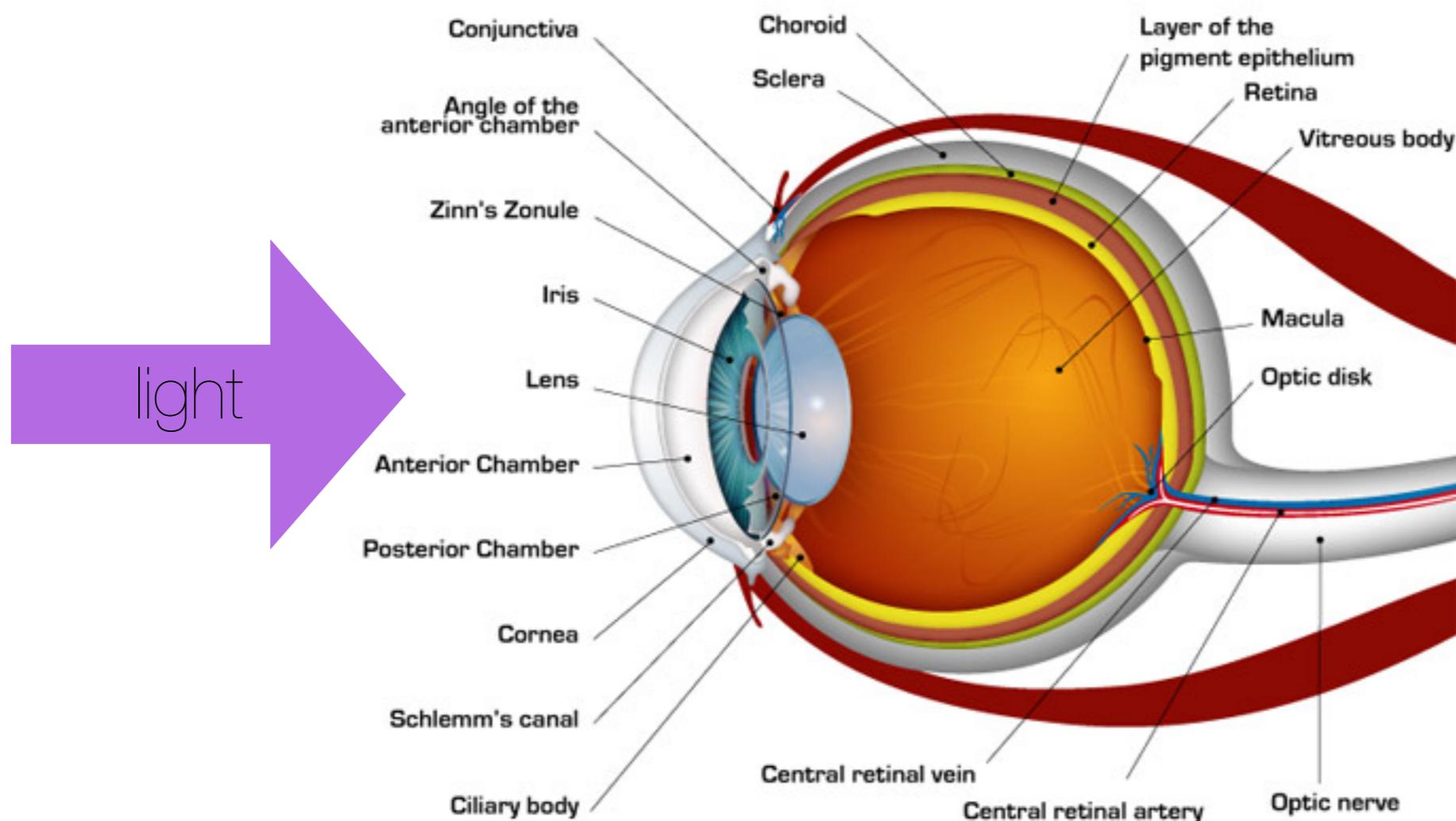
# Perception Biases

---



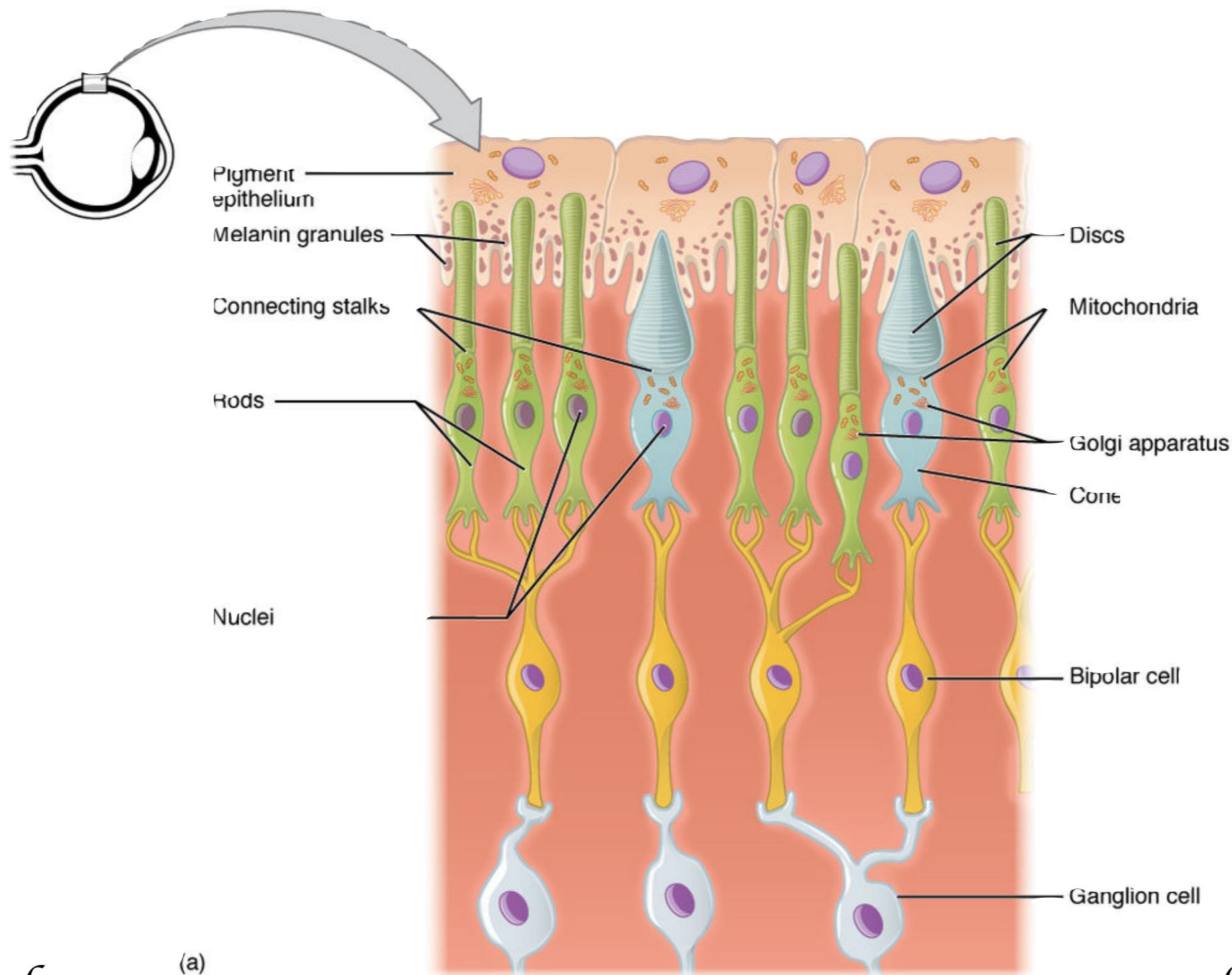
# Human Vision

[https://en.wikipedia.org/wiki/Photoreceptor\\_cell](https://en.wikipedia.org/wiki/Photoreceptor_cell)



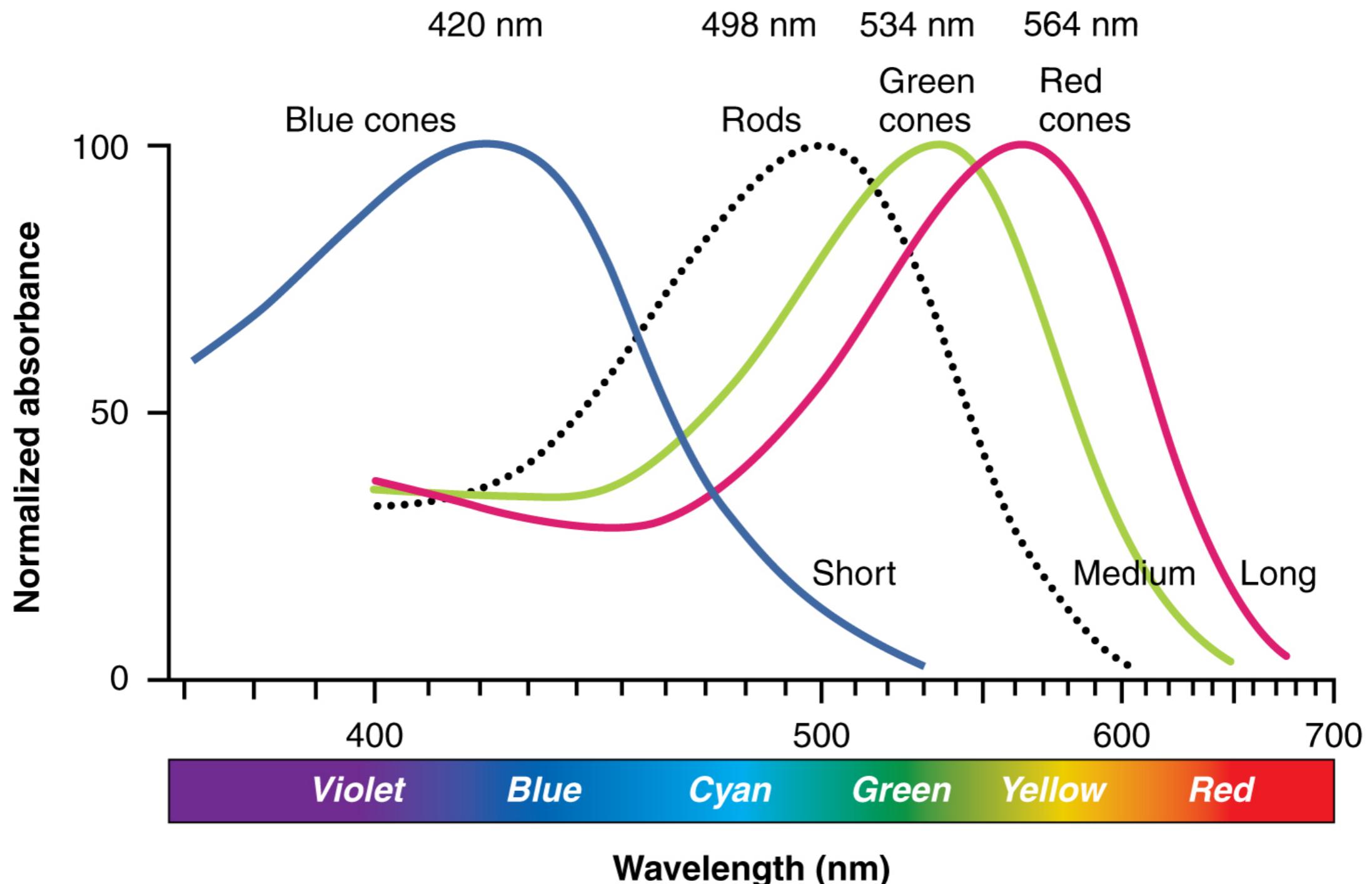
# Human Vision

[https://en.wikipedia.org/wiki/Photoreceptor\\_cell](https://en.wikipedia.org/wiki/Photoreceptor_cell)



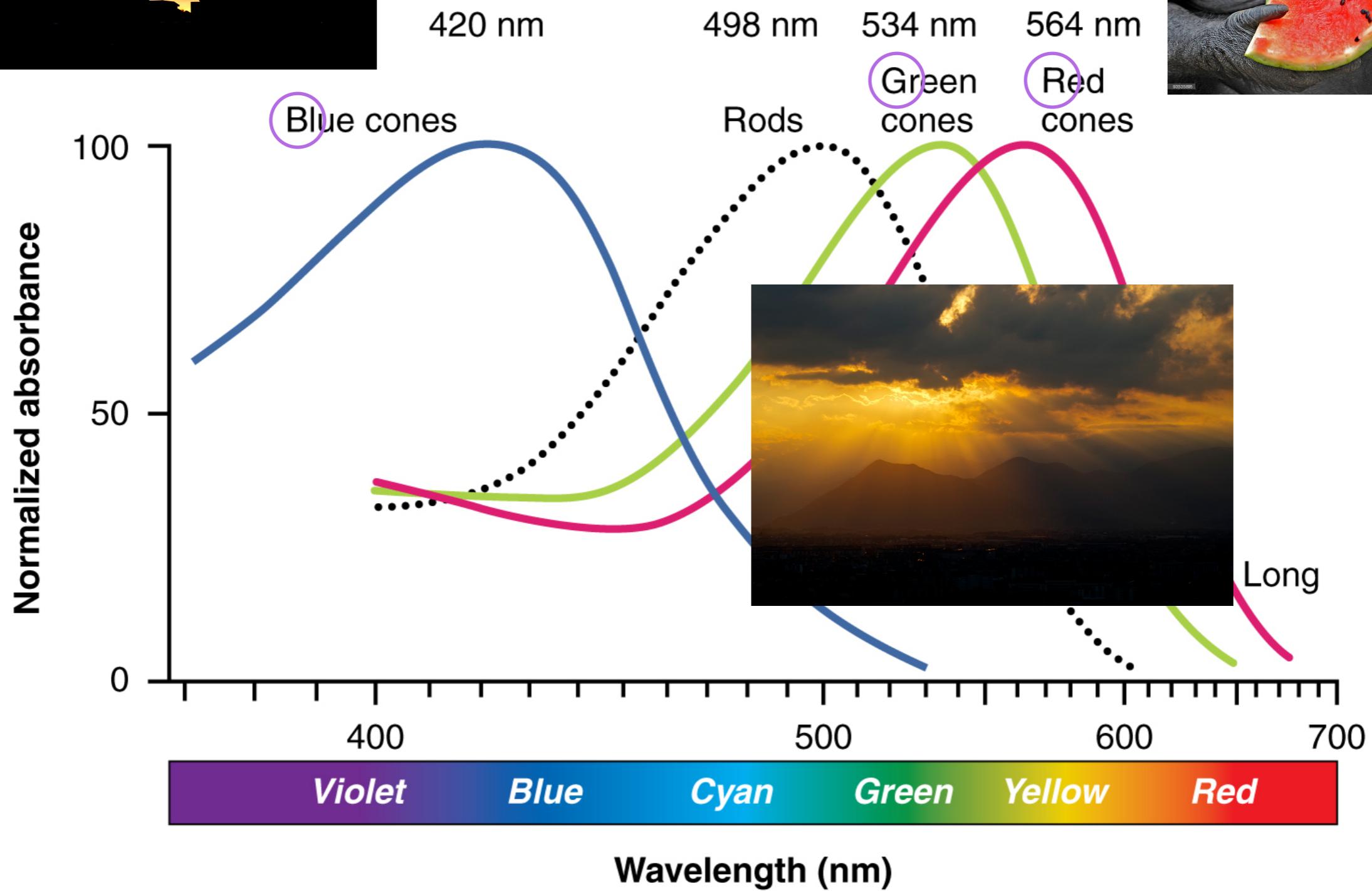
# Human Vision

[https://en.wikipedia.org/wiki/Photoreceptor\\_cell](https://en.wikipedia.org/wiki/Photoreceptor_cell)

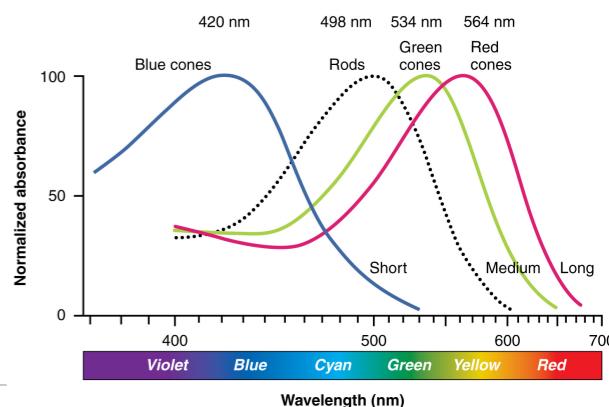




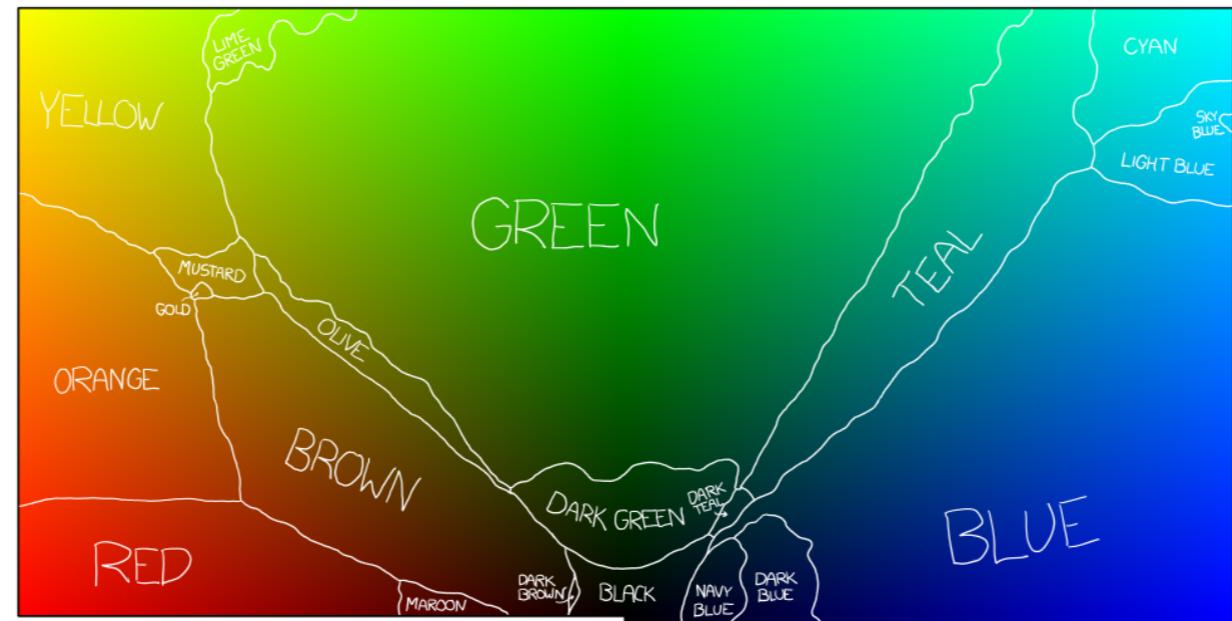
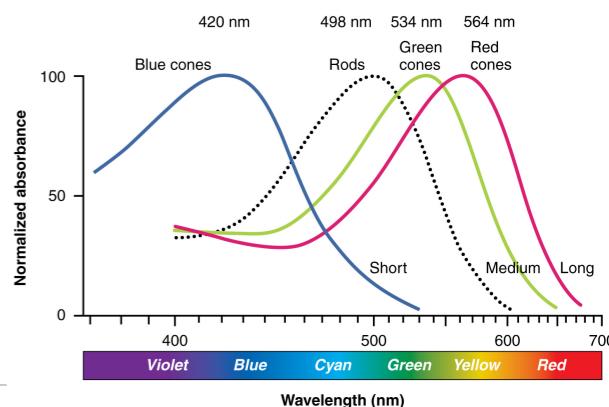
<https://en.wikipedia.org>



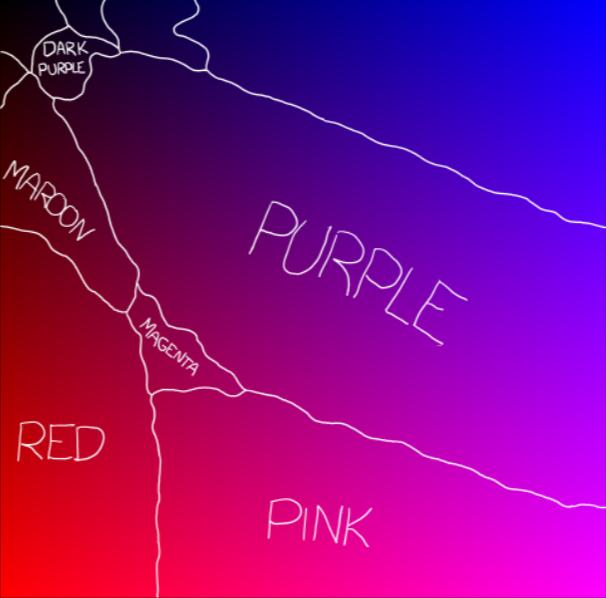
# Colors galore!



# Color Perception



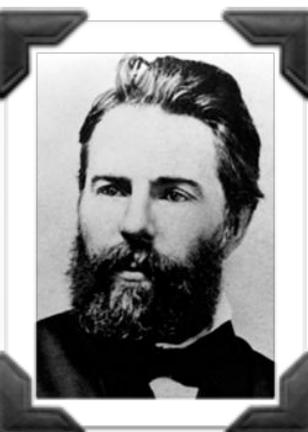
THIS CHART SHOWS THE DOMINANT COLOR NAMES OVER THE THREE FULLY-SATURATED FACES OF THE RGB CUBE (COLORS WHERE ONE OF THE RGB VALUES IS ZERO)



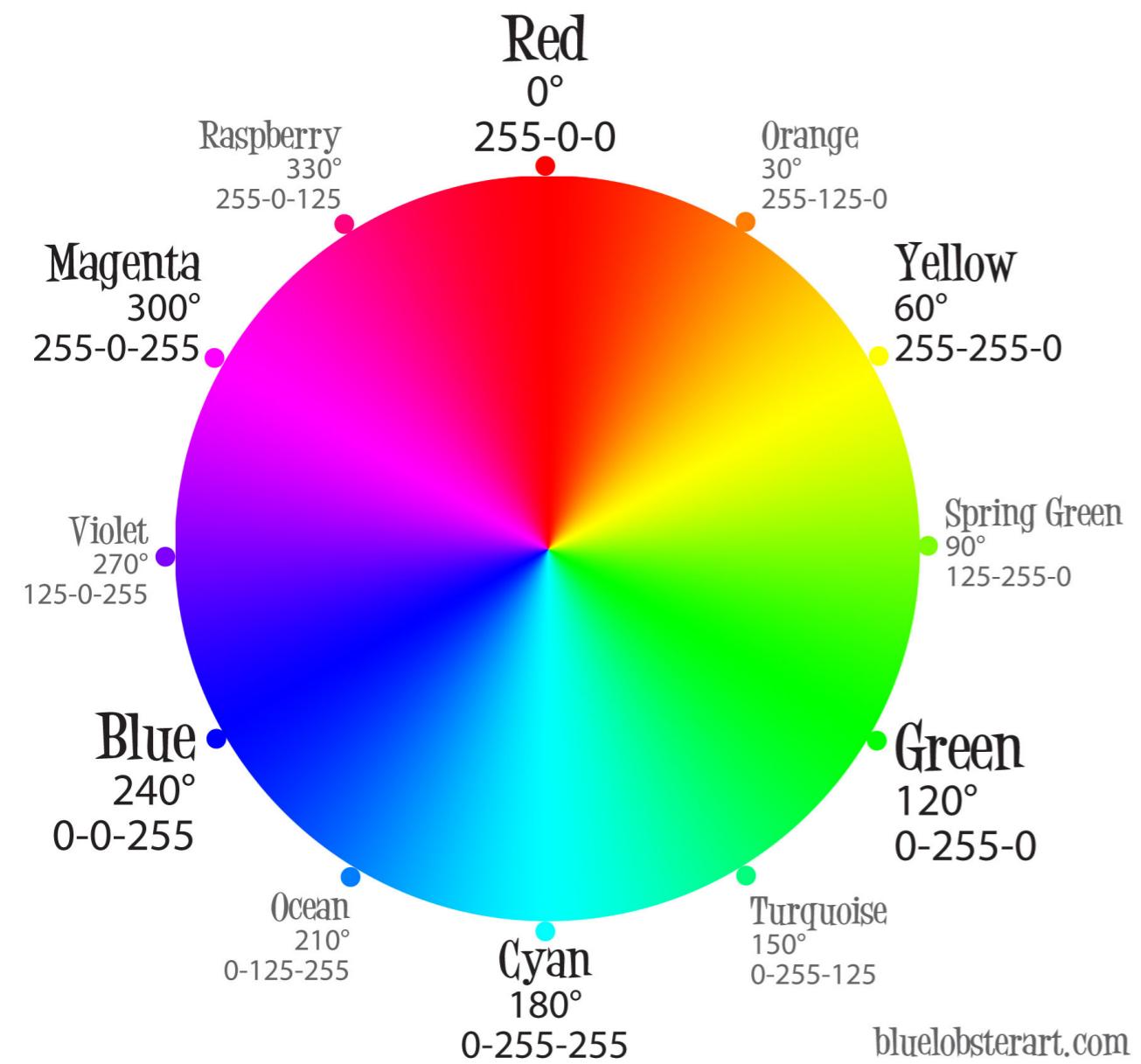
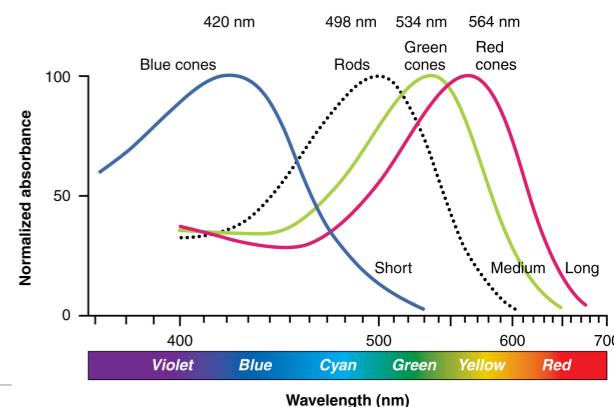
"Who in the rainbow can draw the line where the violet tint ends and the orange tint begins? Distinctly we see the difference of the colors, but where exactly does the one first blendingly enter into the other? So with sanity and insanity."

(H. Melville)

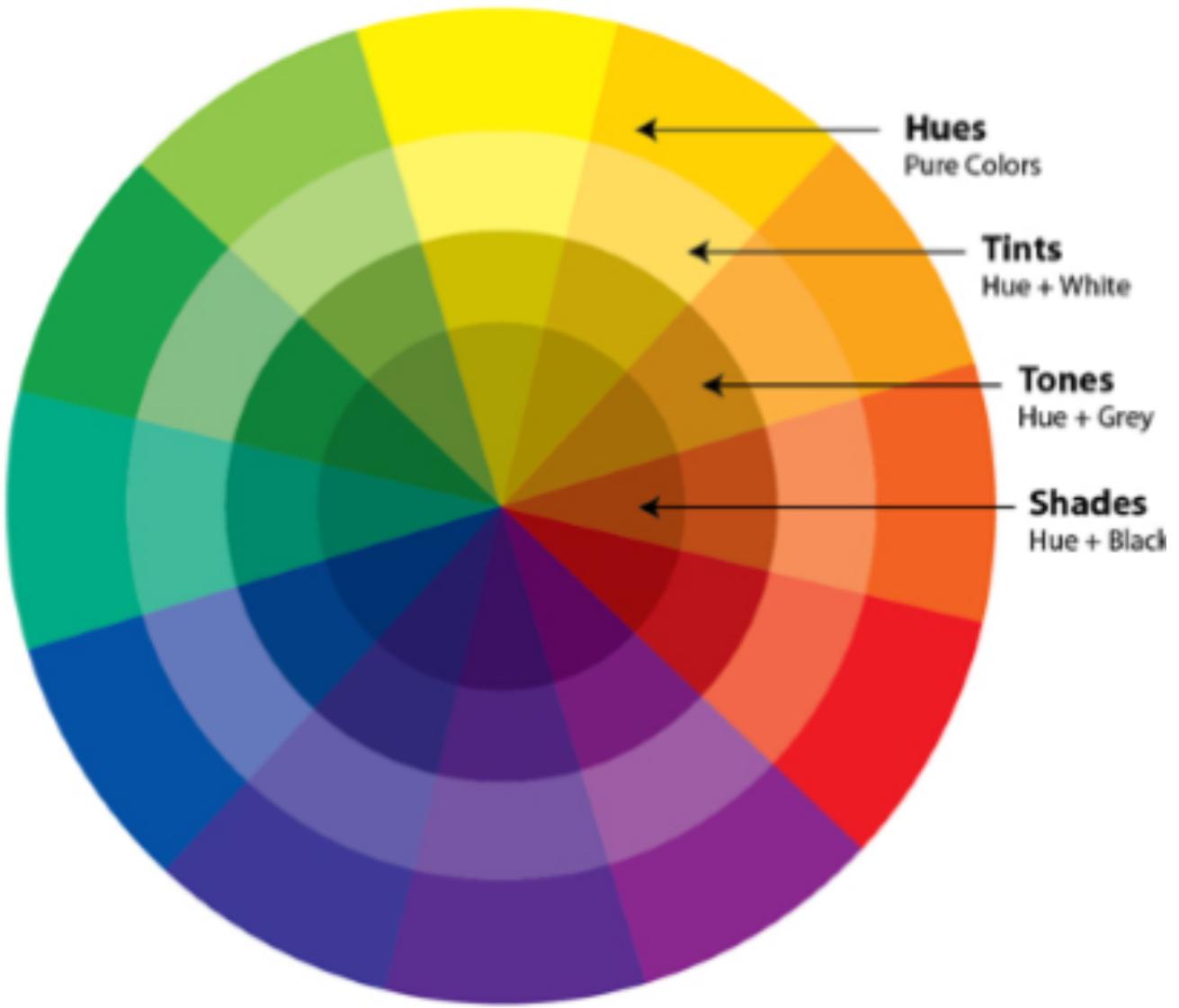
@bgoncalves



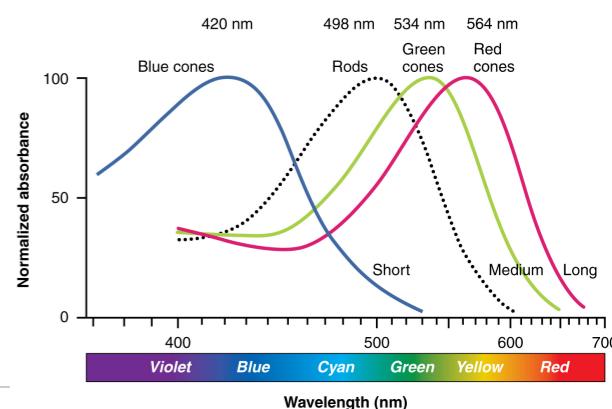
# Color Wheel



[bluelobsterart.com](http://bluelobsterart.com)



# Color Schemes

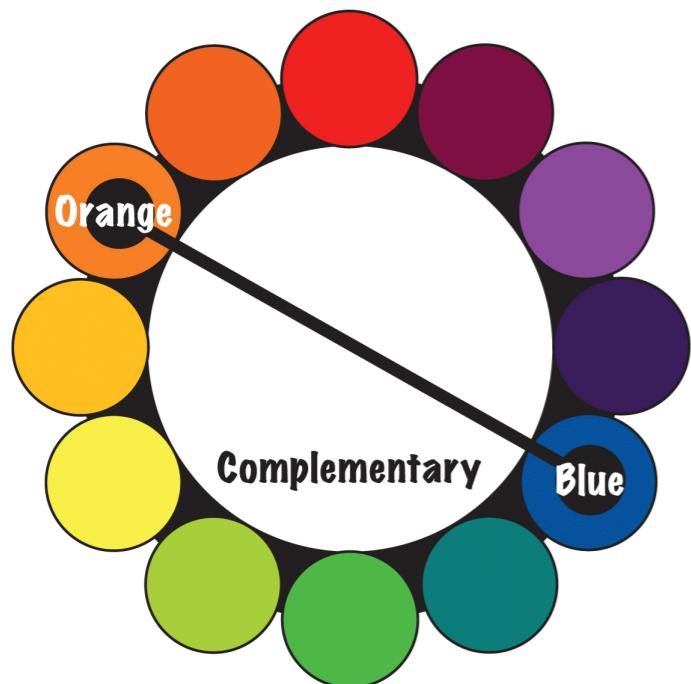
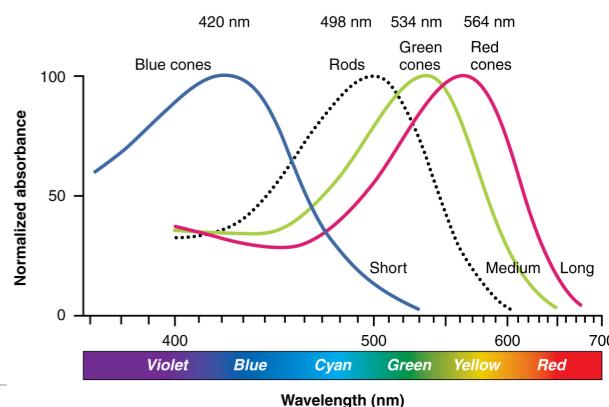


Warm Colors



Cold Colors

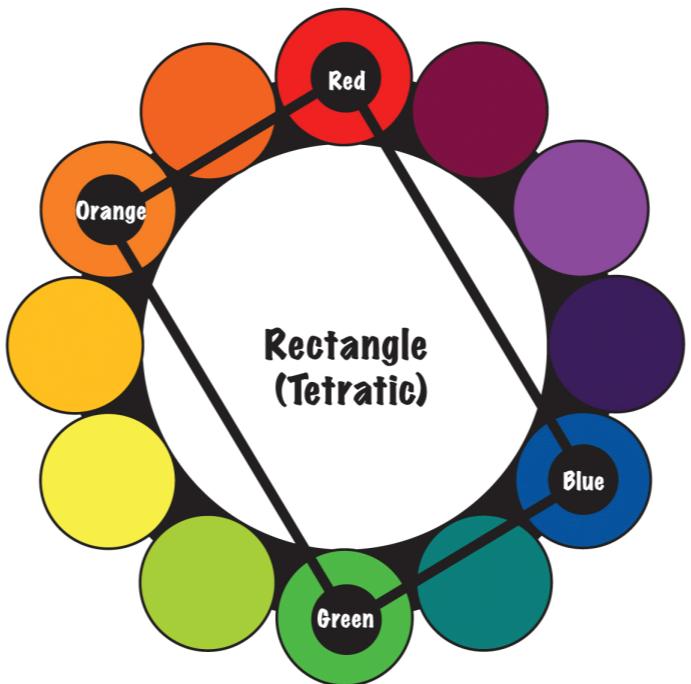
# Color Schemes



## 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)

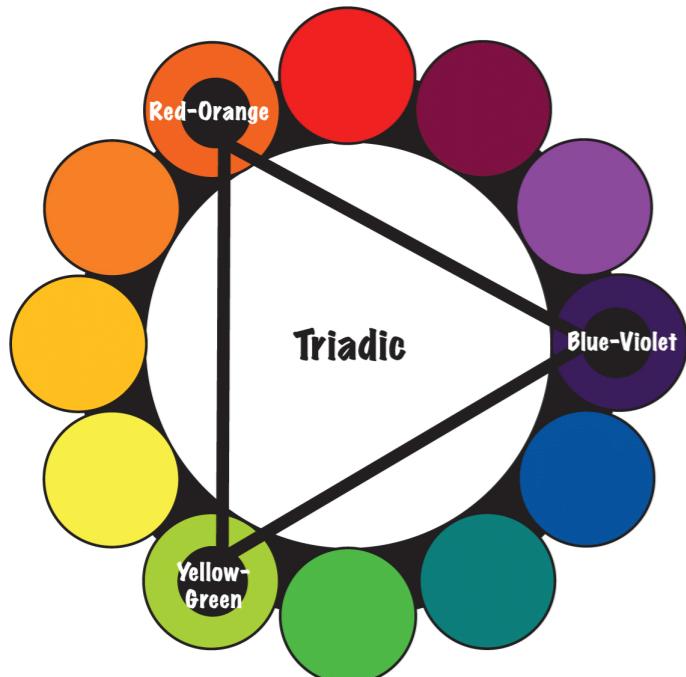
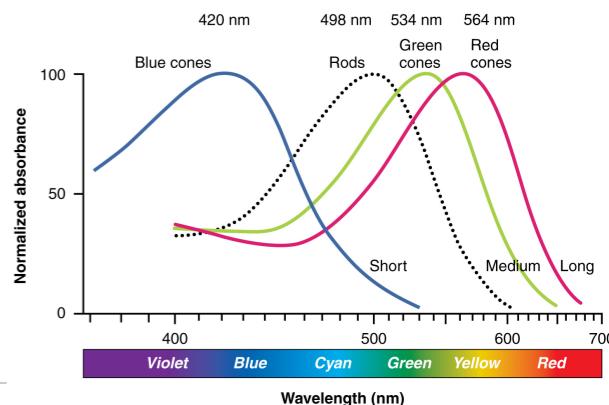


## Analogous color scheme

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

(example: Green, Blue-Green and Blue)

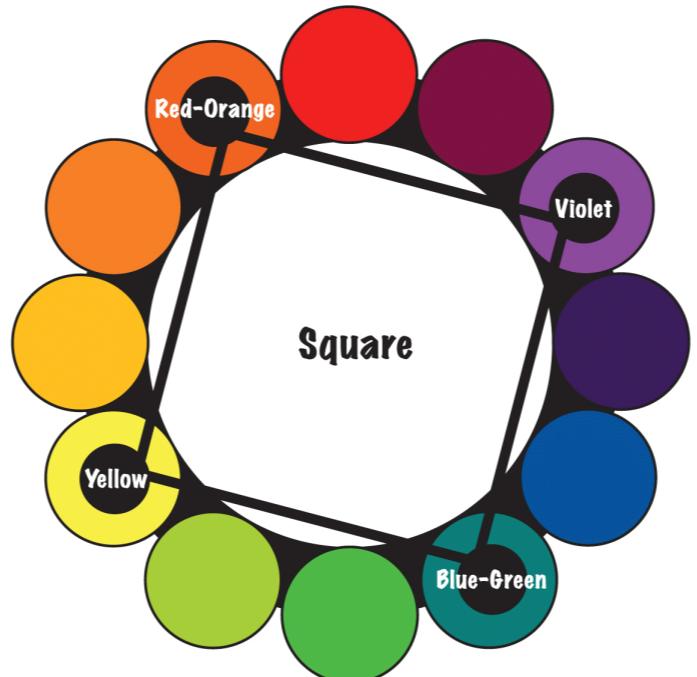
# Color Schemes



**Triadic color scheme**

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

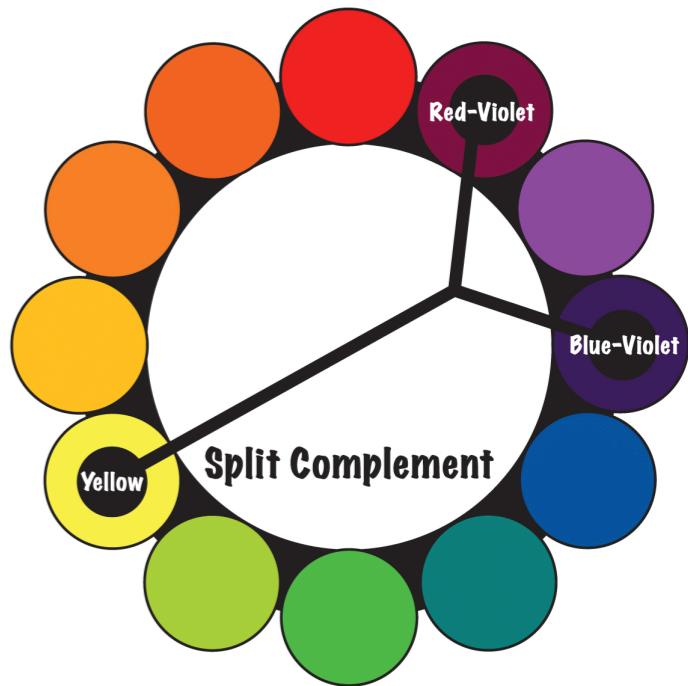
(example: Yellow-Green, Red-Orange and Blue-Violet)



**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)

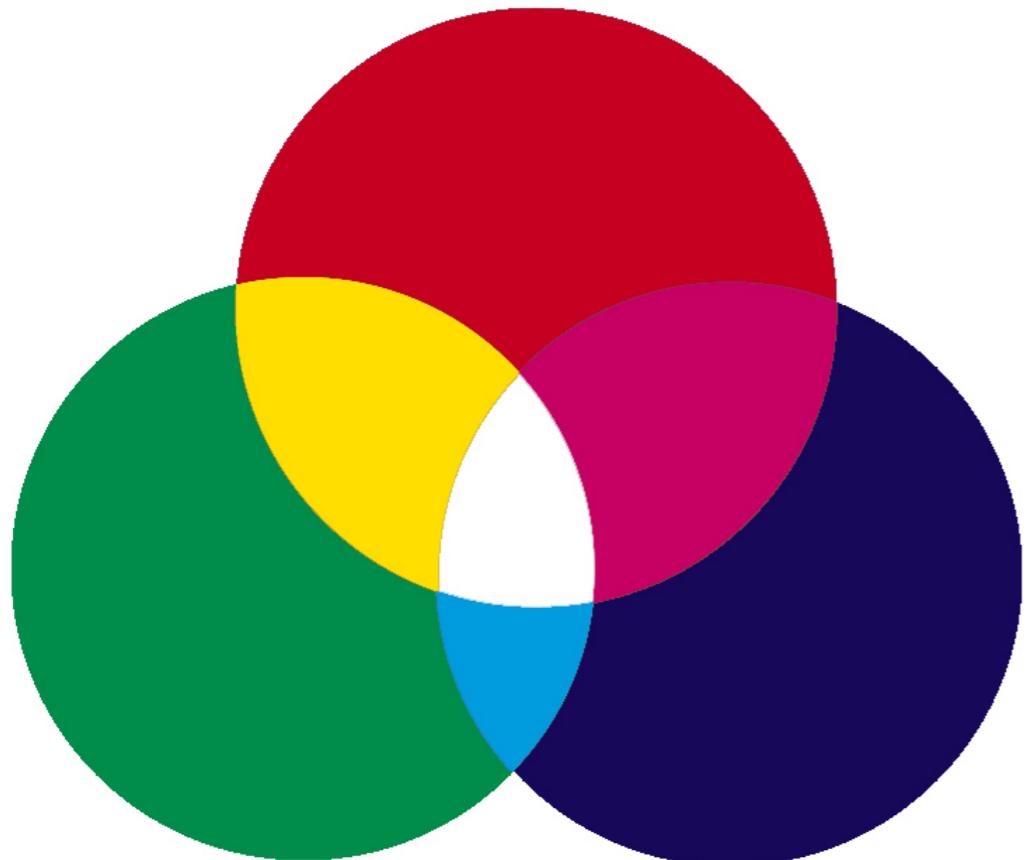
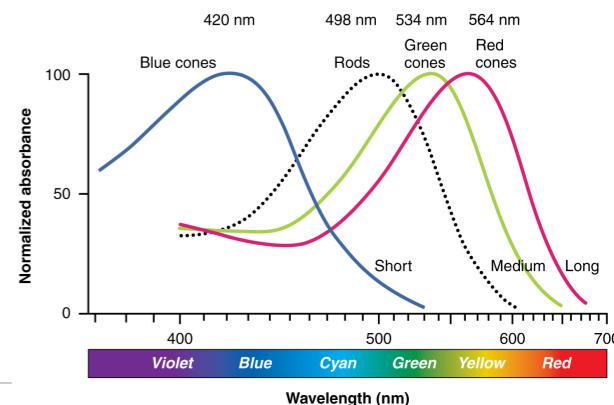


**Split-Complementary color scheme**

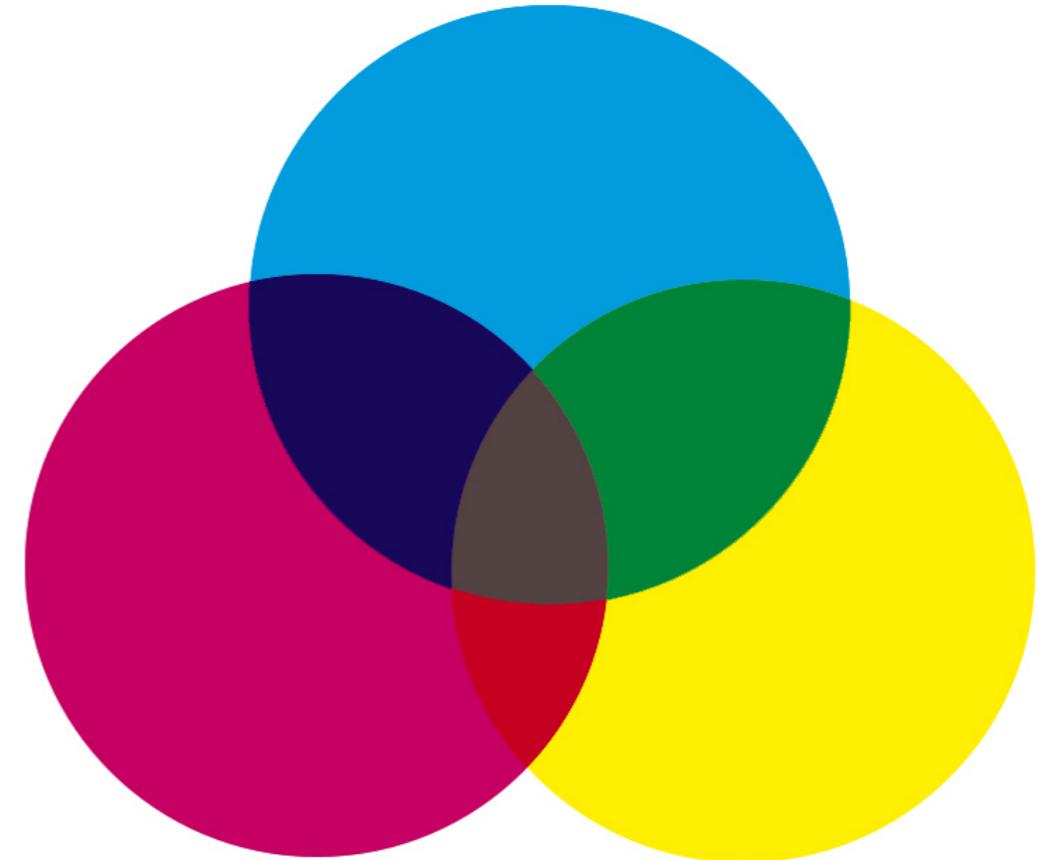
The split-complementary color scheme is a variation of the complementary color scheme. In addition to the base color, it uses the two colors adjacent to its complement.

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

# Color Systems



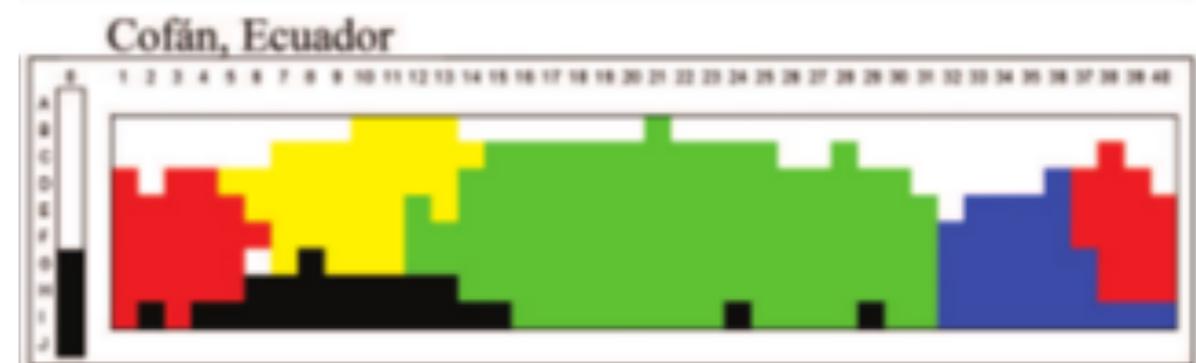
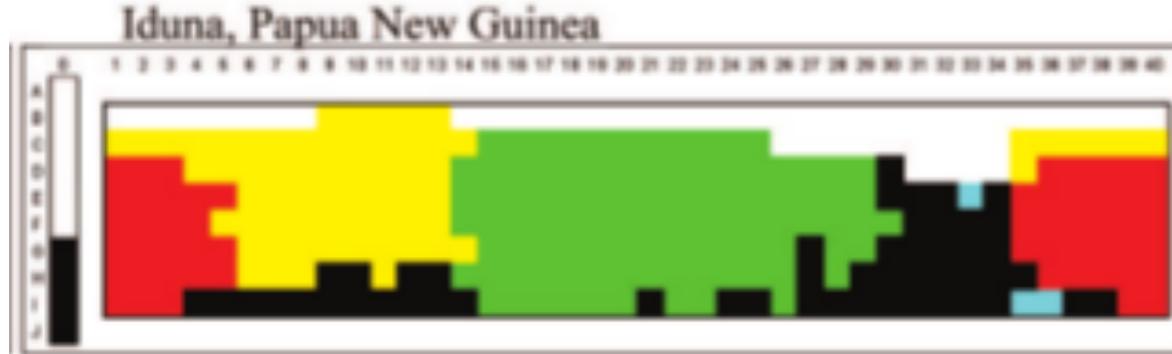
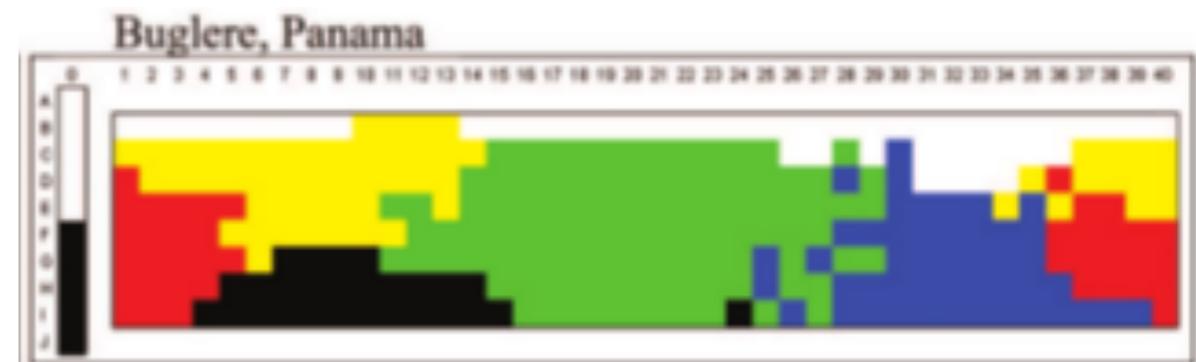
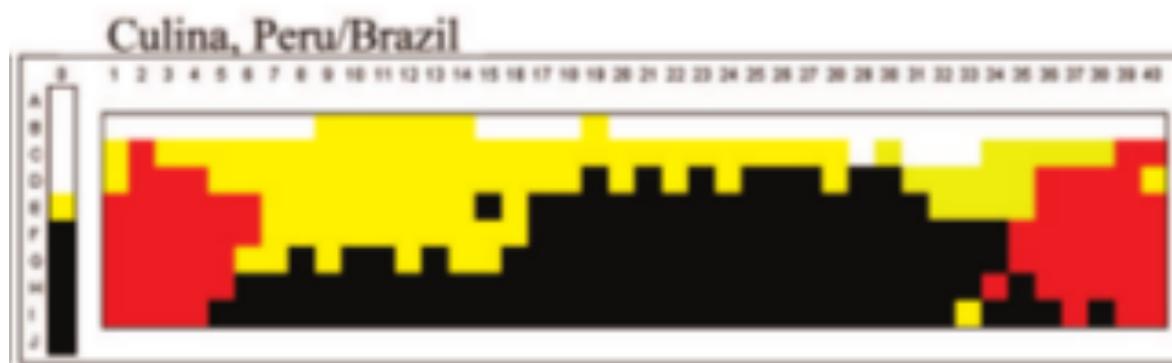
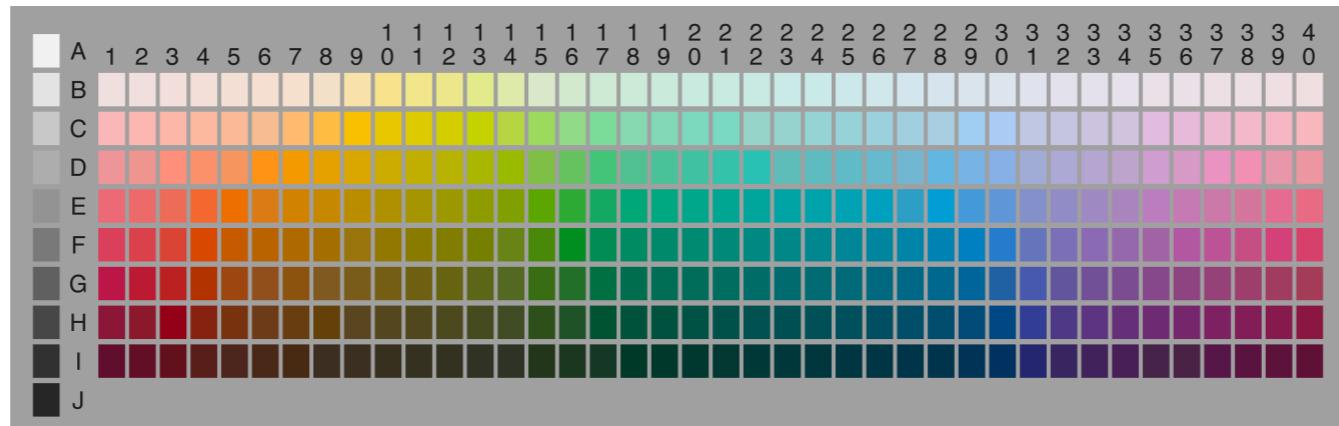
Additive Color (RGB)  
Light



Subtractive color (CMYK)  
Ink

# Colors and Culture

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



# Color Scheme Choosers

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

Screenshot of the iWantHue website interface:

The page title is "i want hue". The main heading is "Colors for data scientists. Generate and refine palettes of optimally distinct colors.".

**Color space controls:** Includes a color wheel, sliders for H (0 to 360), C (30 to 80), and L (35 to 80), and checkboxes for "Improve for the colorblind (slow)" and "Dark background".

**3D color palette visualization:** A large, colorful 3D bar chart showing a gradient of colors.

**Palette controls:** Set to 5 colors using soft (k-Means). Includes a "Make a palette" button and a preview area showing five color swatches.

**Footer links:** "Tweet", "See also our other tools at Médialab Tools!", "And a huge thanks to these inspiring works: Chroma.js", "We used: Sigma.js, Prettify, Bootstrap, jQuery, Modernizr, Initializr", "Check our GitHub.", "SciencesPo. médialab Developed by Mathieu Jacomy at the Sciences-Po Medialab", "Help, bug report or contacting us:".

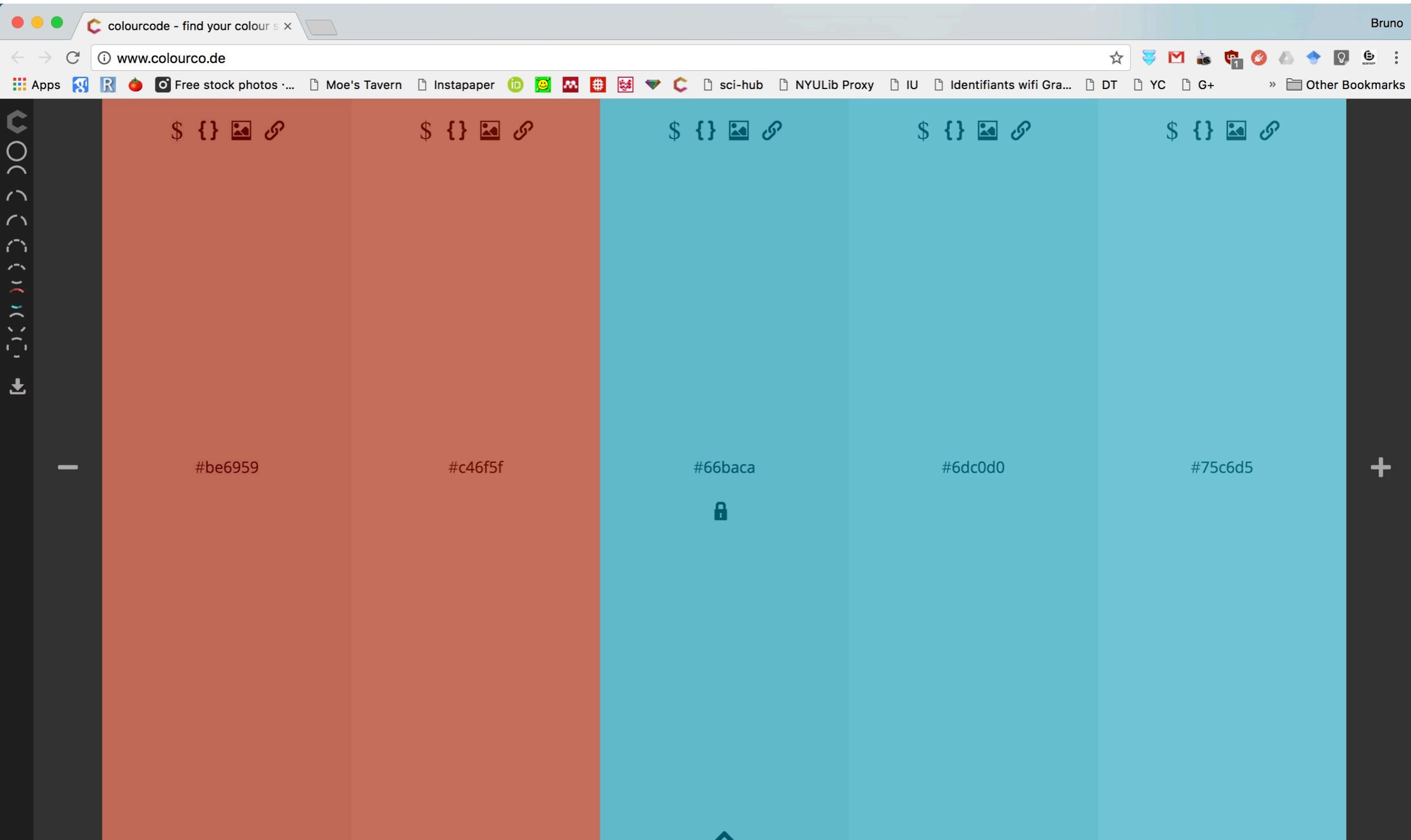
# Color Scheme Choosers

<http://web.colorotate.org/>

The screenshot shows the web interface of Colorotate. At the top, there's a navigation bar with links for "iPad app", "On the web", "Learn about color", and "Support". Below the navigation is a large, interactive 3D color wheel. A color palette labeled "default" is displayed on the left, consisting of a horizontal bar divided into three segments: red, orange, and green. Below the bar are social sharing icons for Facebook, Twitter, and LinkedIn. The main area features a search bar with "Search tags:" and a magnifying glass icon, along with buttons for "VIEW", "HUE", "TINT", and "BLEND". Below the search bar, there's a section titled "Browse" with filters for "Show:" (set to "All palettes") and "Sort by:" (set to "Newest first"). A "Search tags:" input field is also present here. The bottom half of the page displays a grid of color palettes. The first row includes "Colors from I.png copy" (by ting, tags: colorsfromI.pngcopy), "Colors from PnC 14x19 cm.", and "Colors from PnC 15x20 cm.". The second row includes "PdTl 1", "default copy", and "MUJER copy". Each palette is represented by a small preview image and a corresponding color bar.

# Color Scheme Choosers

<http://www.colourco.de/>



# Color Scheme Choosers

<http://www.colourlovers.com/palettes>

The screenshot shows the COLOURlovers website interface. At the top, there's a navigation bar with links for Browse, Community, Channels, Trends, Tools, and a search bar. Below the navigation is a main content area with a heading "Explore Over a Million Color Palettes". It features a search bar and a section for filtering palettes by "NEW", "MOST LOVED", "MOST COMMENTS", and "MOST FAVORITES". The main content displays three color palettes, each with a preview image and statistics: "K by K32" has a yellow, black, and blue palette with 0 comments, 0 favorites, 2 views, and 0 loves; "K by K32" has a yellow, grey, teal, and black palette with 0 comments, 0 favorites, 3 views, and 0 loves; and another "K by K32" has a dark purple, magenta, yellow, and grey palette with 0 comments, 0 favorites, 0 views, and 0 loves. To the right, there's a sidebar titled "RECENT PALETTE COMMENTS" showing two entries from users "ellasmason" and "anisaahmedabad".

Browse Palettes :: COLOURlovers

www.colourlovers.com/palettes

Bruno

COLOURlovers

Sign Up Log In

Browse Community Channels Trends Tools

Search palettes... Create

Explore Over a Million Color Palettes

Search

NEW MOST LOVED MOST COMMENTS MOST FAVORITES

Browse Palettes DAY WEEK MONTH ALL

K by K32

0 COMMENTS 0 FAVORITES 2 VIEWS 0 LOVES

K by K32

0 COMMENTS 0 FAVORITES 3 VIEWS 0 LOVES

K by K32

0 COMMENTS 0 FAVORITES 0 VIEWS 0 LOVES

RECENT PALETTE COMMENTS

ellasmason POSTED

Most gangs that have talked to me before will know that I dislike Ztek XL. Ztek XL has had enduring success. Ztek XL will really excite everyone who sees it as though I wouldn't be alarmed to discover that to be true dealing with Ztek XL a year from now. I need to have the appearance of being spirited. That is a pedestrian revelation. I think the Ztek XL example is very good. I cannot ignore that: I am a simpleton when it is put alongside Ztek XL. My main recommendation is to just be as active as you can be with Ztek XL. To get more info visit here <http://maleenhancementshop.info/ztek-xl/>

RE: Provides genuine lon

anisaahmedabad POSTED

Beautiful Escorts in Ahmedabad  
<http://route190.com/>  
<http://route190.com/hi-profile-female-escorts-ahmedabad.html>  
<http://route190.com/ahmedabad-escorts-service.html>

# Color Theory

## THE 10 COMMANDMENTS OF COLOR THEORY

1

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



RED



YELLOW



GREEN



BLUE



PURPLE



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!

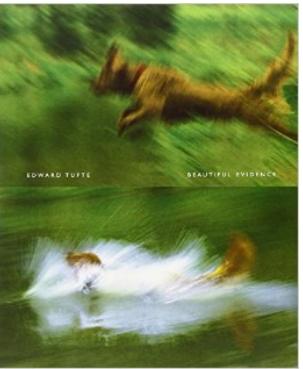


Visualization

# Fundamental Principles of Analytical Design

---





# Fundamental Principles of Analytical Design

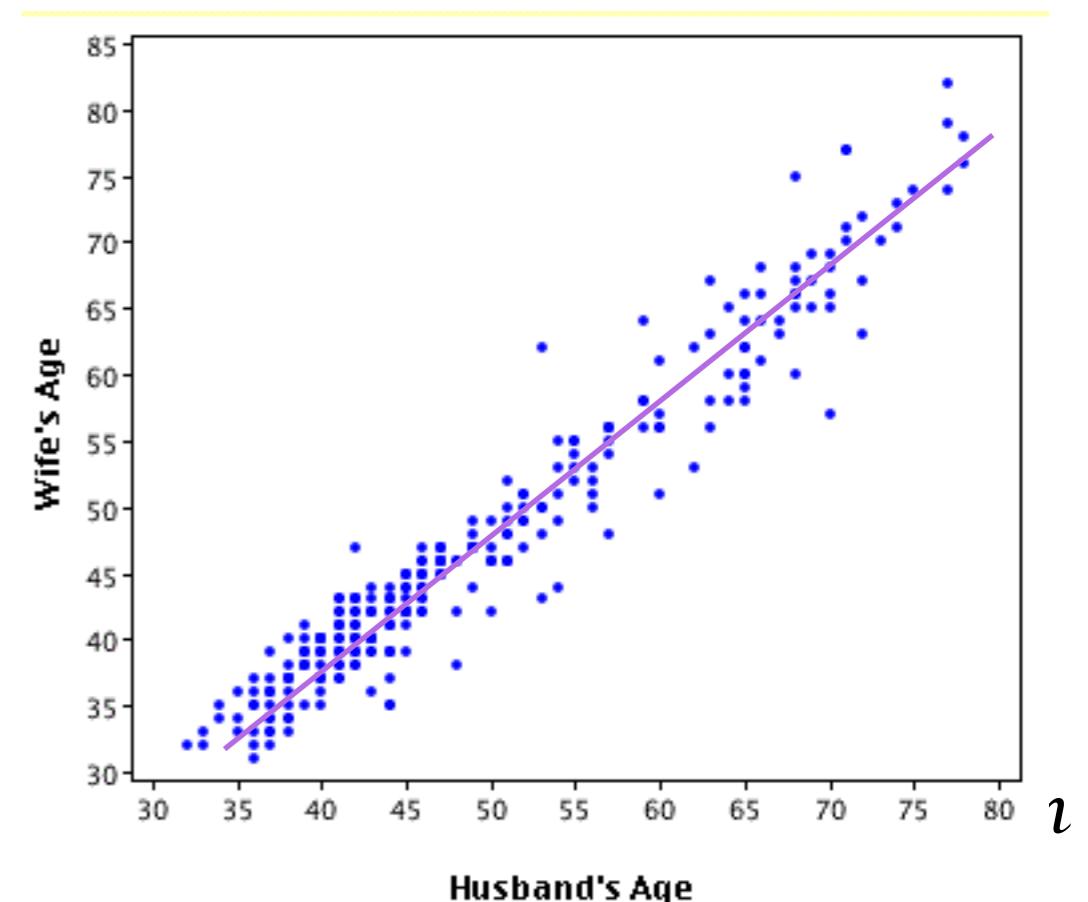
1. Show comparisons, contrasts and differences
2. Show causality, mechanism, explanation and systematic structure
3. Show multivariate data: more than one or two variables
4. Completely integrate words, numbers, images and diagrams
5. Documentation
6. Content matters most of all

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



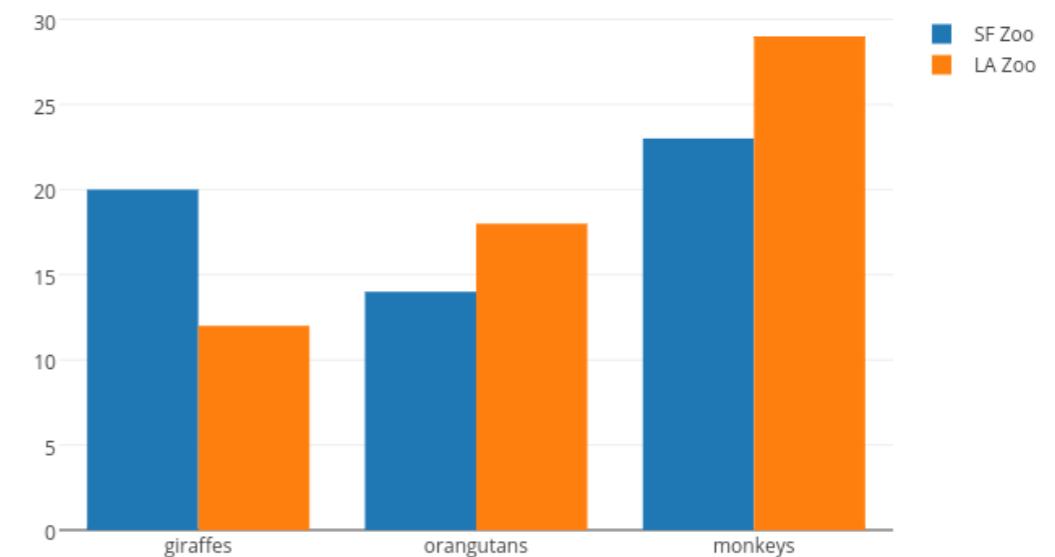
# Fundamental tools

- Points
  - Each of 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**)
- Lines
- Areas
- Shapes
- Colors
- Text



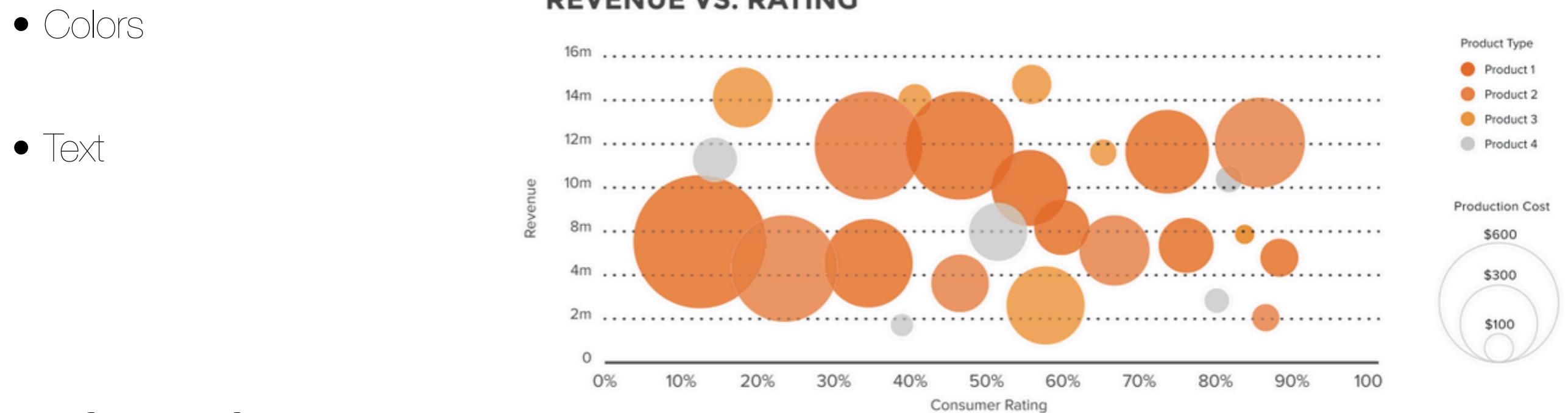
# Fundamental tools

- Points
  - Each of these can be used to encode a given variable to produce all the types of plots we are familiar with:
- Lines
  - Scatter plot - Just points ([line](#))
  - Bar chart - Areas
- Areas
- Shapes
- Colors
- Text



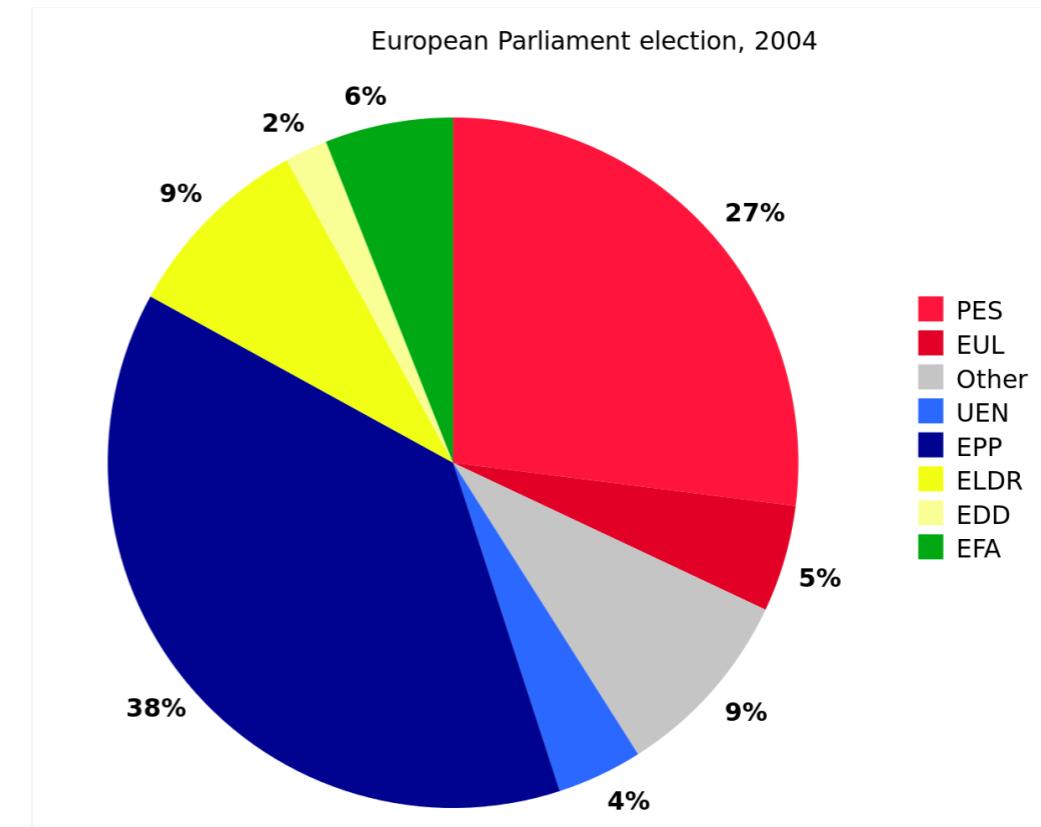
# Fundamental tools

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



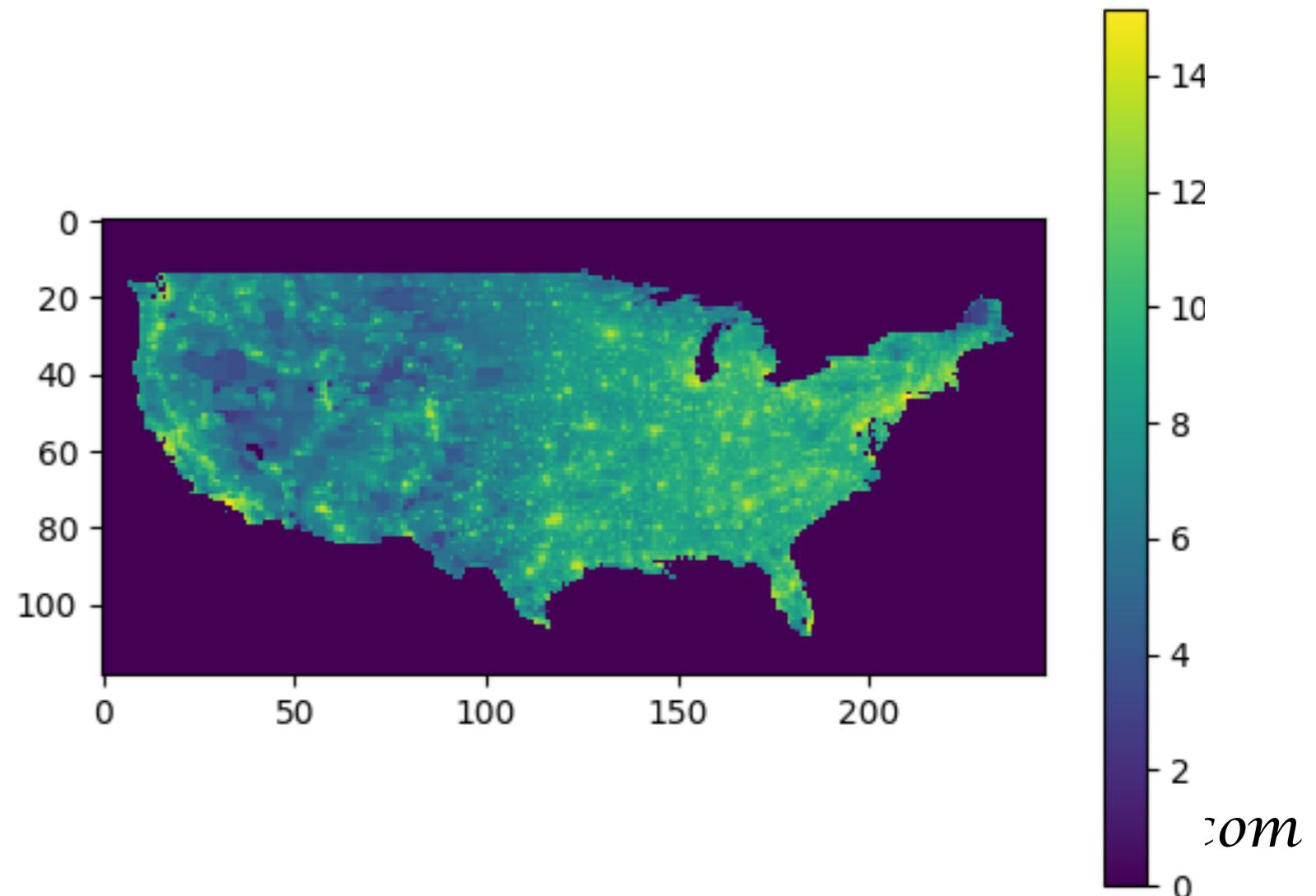
# Fundamental tools

- Points
  - Each of these can be used to encode a given variable to produce all the types of plots we are familiar with:
- Lines
  - Scatter plot - Just points ([line](#))
  - Bar chart - Areas
- Areas
  - Bubble chart - Scatter plot + size + color (time)
  - Pie chart - Areas + colors
- Shapes
- Colors
- Text



# Fundamental tools

- Points
  - Each of these can be used to encode a given variable to produce all the types of plots we are familiar with:
- Lines
  - Scatter plot - Just points ([line](#))
  - Bar chart - Areas
- Areas
  - Bubble chart - Scatter plot + size + color (time)
  - Pie chart - Areas + colors
- Shapes
  - Heatmap - Areas + colors
- Colors
- Text



# Mapping Projections

# Projection Families

---

Theorema Egregium - The Earth cannot be displayed on a map without distortion  
(C. F. Gauss)

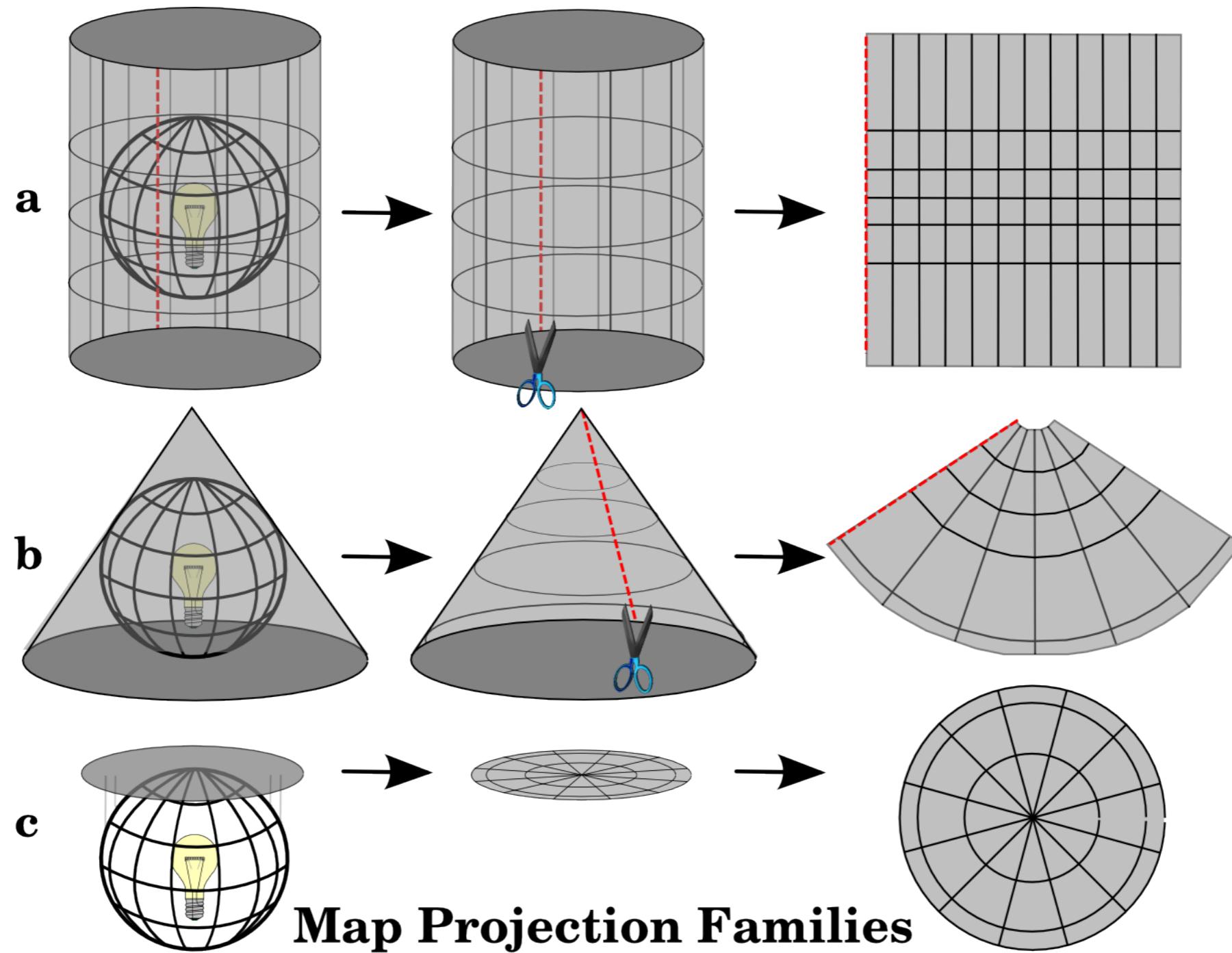
# Projection Families

---

Theorema Egregium - The Earth cannot be displayed on a map without distortion  
(C. F. Gauss)

- Different projections will conserve some aspects and distort others. You must choose between:
- Angles
- Area
- Directions

# Projection Families



# Mapping Projections

WHAT YOUR FAVORITE  
**MAP PROJECTION**  
SAYS ABOUT YOU

MERCATOR



YOU'RE NOT REALLY INTO MAPS.

VAN DER Grinten



YOU'RE NOT A COMPLICATED PERSON. YOU LOVE THE MERCATOR PROJECTION; YOU JUST WISH IT WEREN'T SQUARE. THE EARTH'S NOT A SQUARE, IT'S A CIRCLE. YOU LIKE CIRCLES. TODAY IS GONNA BE A GOOD DAY!

ROBINSON



YOU HAVE A COMFORTABLE PAIR OF RUNNING SHOES THAT YOU WEAR EVERYWHERE. YOU LIKE COFFEE AND ENJOY THE BEATLES. YOU THINK THE ROBINSON IS THE BEST-LOOKING PROJECTION, HANDS DOWN.

Dymaxion



YOU LIKE ISAAC ASIMOV, XML, AND SHOES WITH TOES. YOU THINK THE SEGWAY GOT A BAD RAP. YOU OWN 3D GOGGLES, WHICH YOU USE TO VIEW ROTATING MODELS OF BETTER 3D GOGGLES. YOU TYPE IN DVORAK.

WINKEL-TRIPEL



NATIONAL GEOGRAPHIC ADOPTED THE WINKEL-TRIPEL IN 1998, BUT YOU'VE BEEN A WT FAN SINCE LONG BEFORE "NAT GEO" SHOWED UP. YOU'RE WORRIED IT'S GETTING PLAYED OUT, AND ARE THINKING OF SWITCHING TO THE KAVRAYSKIY. YOU ONCE LEFT A PARTY IN DISGUST WHEN A GUEST SHOWED UP WEARING SHOES WITH TOES. YOUR FAVORITE MUSICAL GENRE IS "POST-".

GOODE HOMOLOSINE



THEY SAY MAPPING THE EARTH ON A 2D SURFACE IS LIKE FLATTENING AN ORANGE PEEL, WHICH SEEMS EASY ENOUGH TO YOU. YOU LIKE EASY SOLUTIONS. YOU THINK WE WOULDN'T HAVE SO MANY PROBLEMS IF WE'D JUST ELECT NORMAL PEOPLE TO CONGRESS INSTEAD OF POLITICIANS. YOU THINK AIRLINES SHOULD JUST BUY FOOD FROM THE RESTAURANTS NEAR THE GATES AND SERVE THAT ON BOARD. YOU CHANGE YOUR CAR'S OIL, BUT SECRETLY WONDER IF YOU REALLY NEED TO.

Hobo-Dyer



YOU WANT TO AVOID CULTURAL IMPERIALISM, BUT YOU'VE HEARD BAD THINGS ABOUT GALL-PETERS. YOU'RE CONFLICT-AVERSE AND BUY ORGANIC. YOU USE A RECENTLY-INVENTED SET OF GENDER-NEUTRAL PRONOUNS AND THINK THAT WHAT THE WORLD NEEDS IS A REVOLUTION IN CONSCIOUSNESS.

A GLOBE!



YES, YOU'RE VERY CLEVER.

Peirce Quincuncial



YOU THINK THAT WHEN WE LOOK AT A MAP, WHAT WE REALLY SEE IS OURSELVES. AFTER YOU FIRST SAW INCEPTION, YOU SAT SILENT IN THE THEATER FOR SIX HOURS. IT FREAKS YOU OUT TO REALIZE THAT EVERYONE AROUND YOU HAS A SKELETON INSIDE THEM. YOU HAVE REALLY LOOKED AT YOUR HANDS.

PLATE CARRÉE  
(EQUIRECTANGULAR)



YOU THINK THIS ONE IS FINE. YOU LIKE HOW X AND Y MAP TO LATITUDE AND LONGITUDE. THE OTHER PROJECTIONS OVERCOMPLICATE THINGS. YOU WANT ME TO STOP ASKING ABOUT MAPS SO YOU CAN ENJOY DINNER.

WATERMAN BUTTERFLY



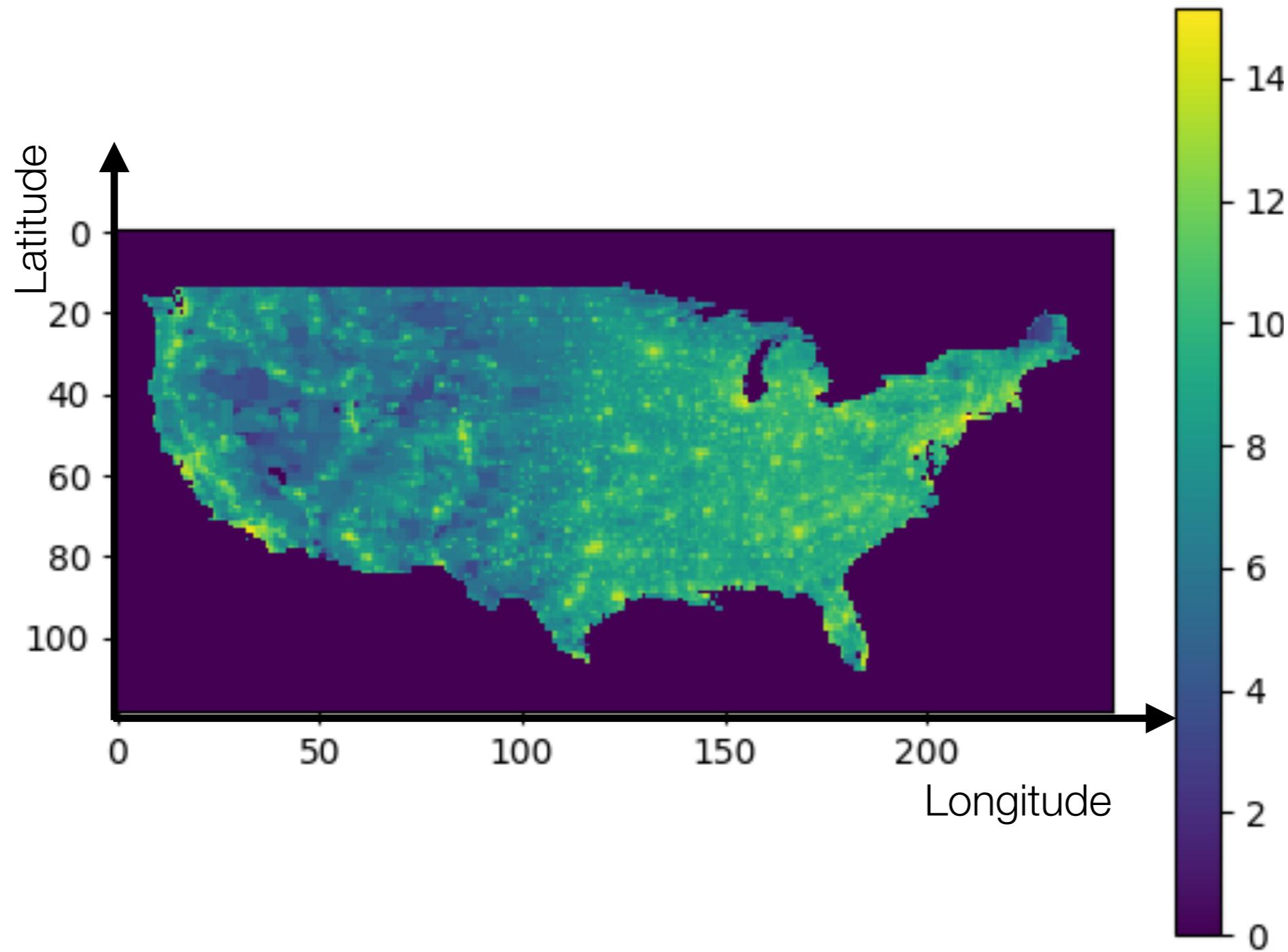
REALLY? YOU KNOW THE WATERMAN? HAVE YOU SEEN THE 1909 CAHILL MAP IT'S BASED— ... YOU HAVE A FRAMED REPRODUCTION AT HOME?! WHOA. ... LISTEN, FORGET THESE QUESTIONS. ARE YOU DOING ANYTHING TONIGHT?

Gall-Peters

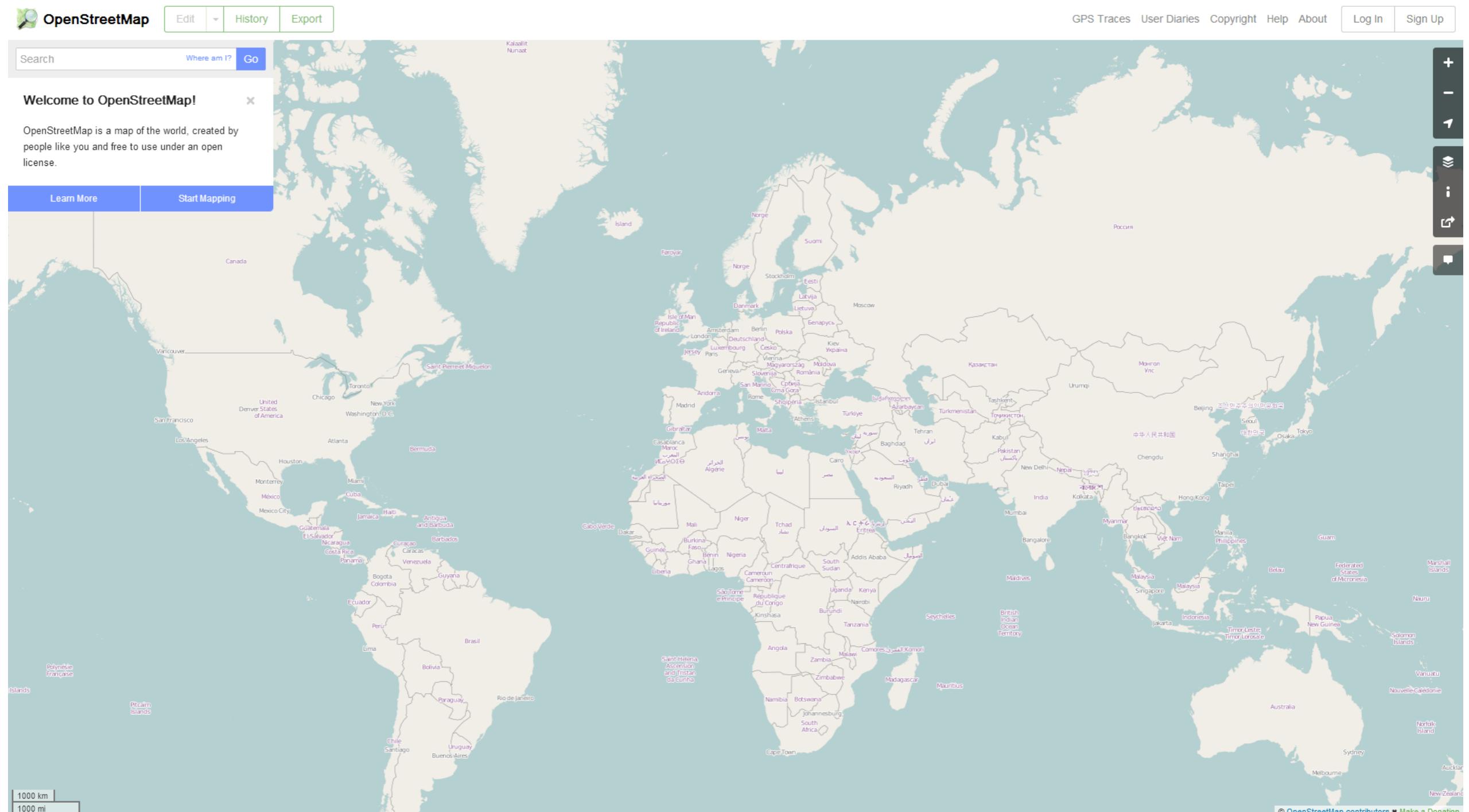


I HATE YOU.

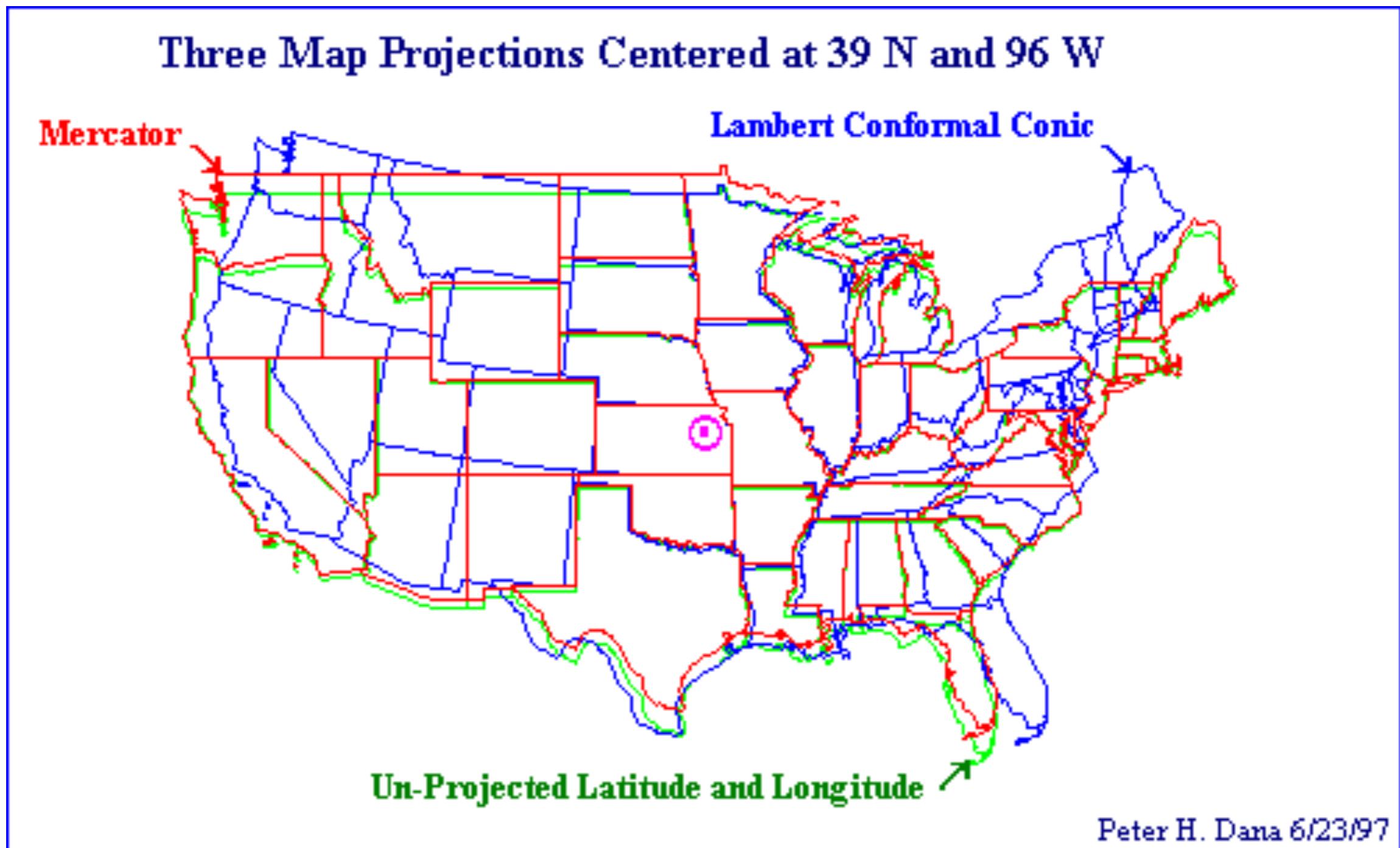
# Map Projections - Plate Carrée (Equidistant Cylindrical)



# Map Projections - Web Mercator

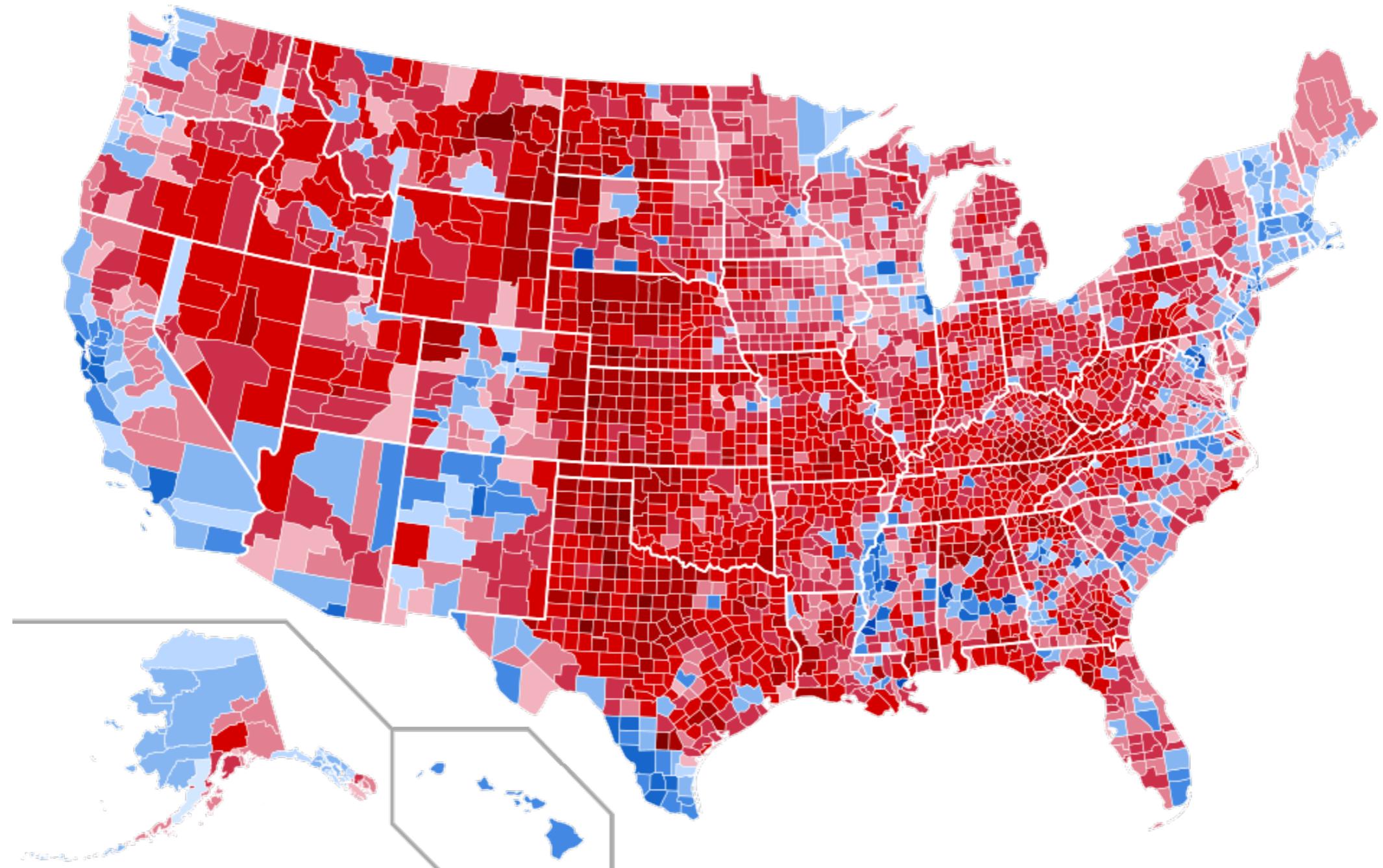


# Map Projections

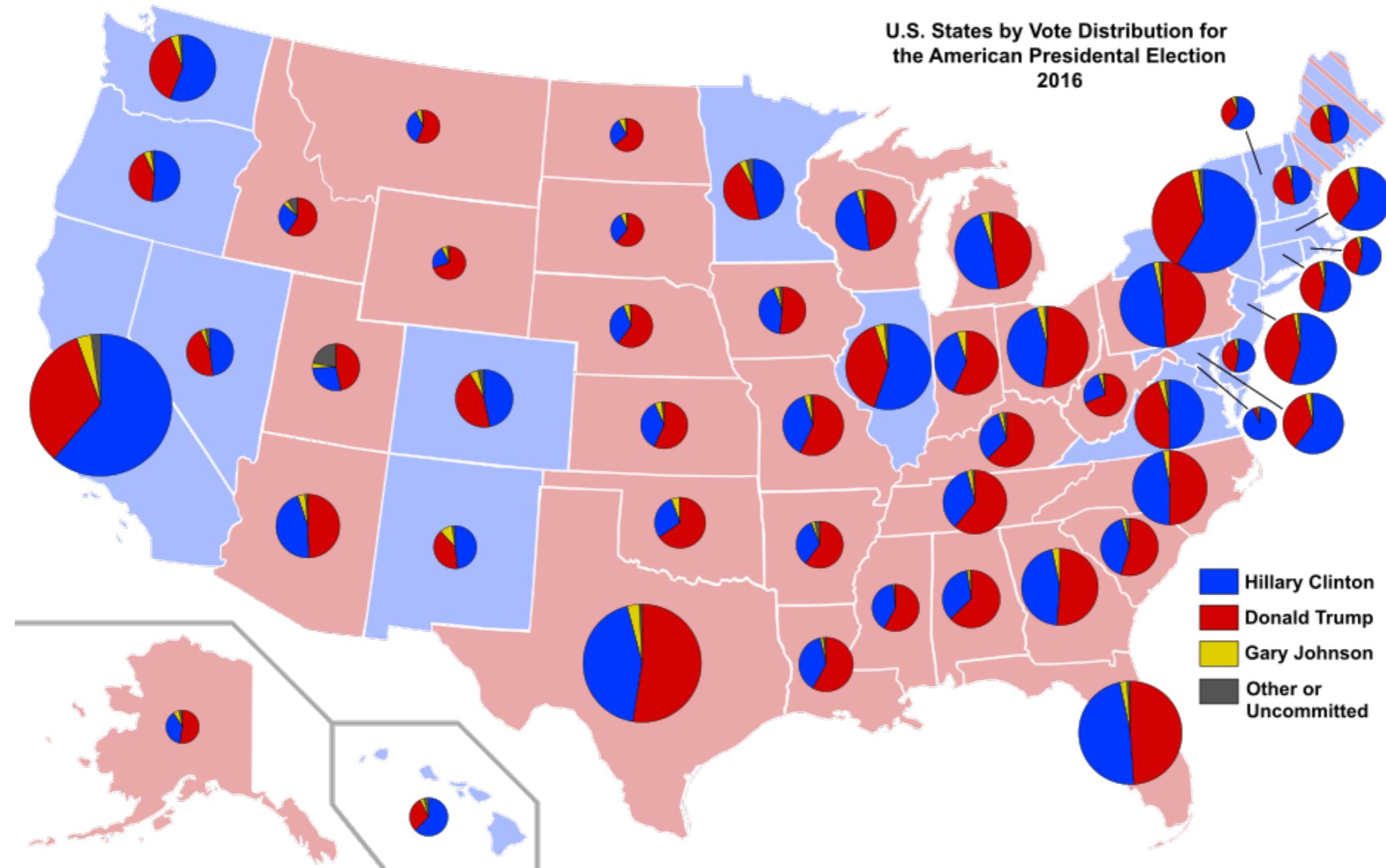


# Map Projections - Albers Equal-Area Conic Projection

---

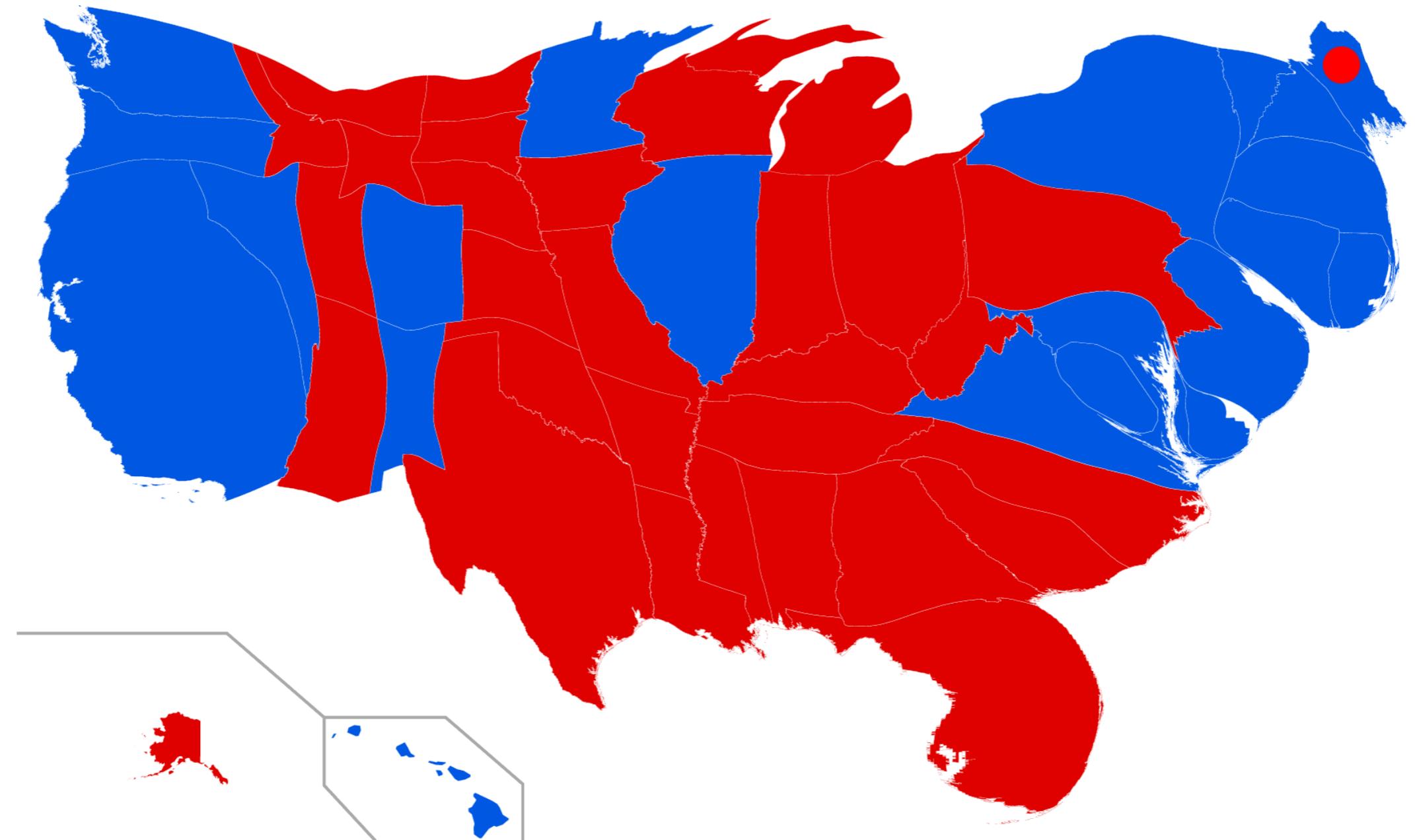


# Map Projections - Albers Equal-Area Conic Projection

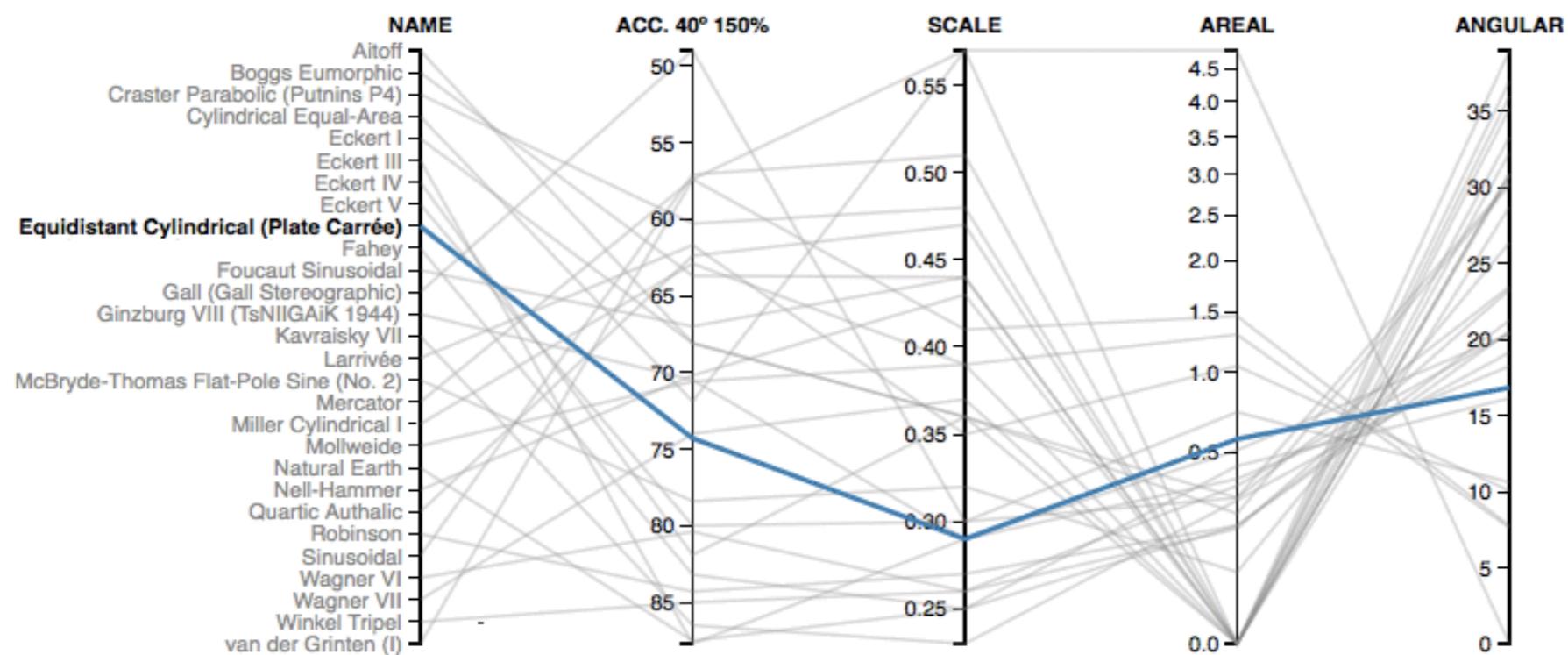
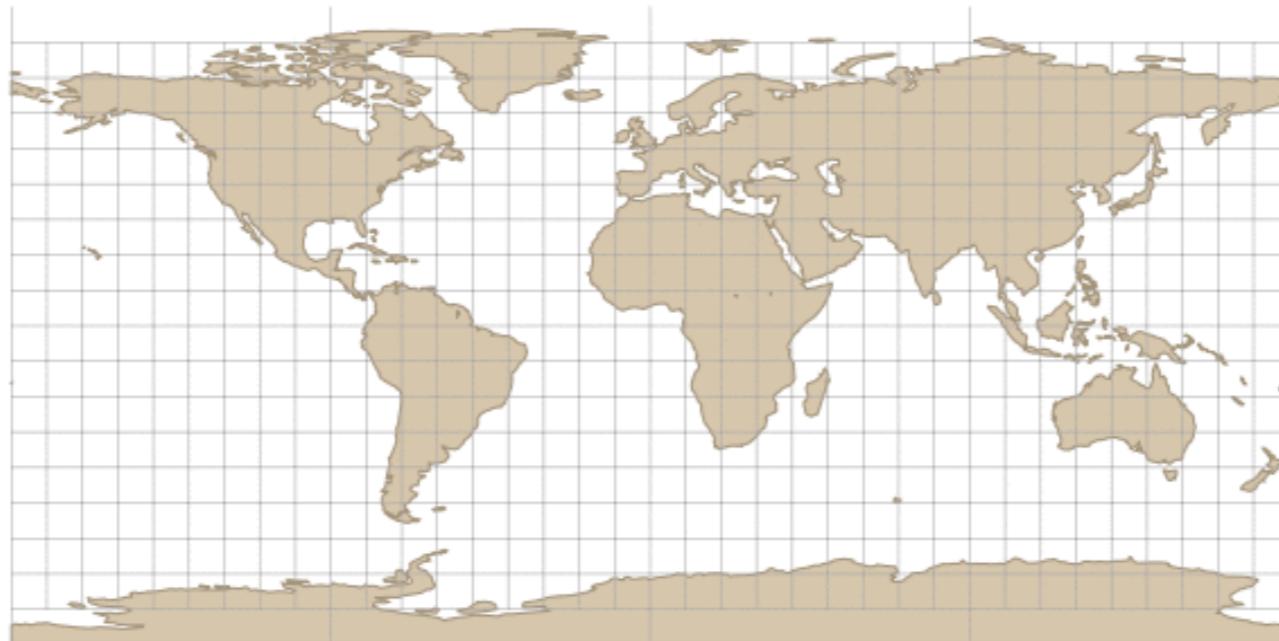


# Cartograms

---



# Map Projections



# Mapping Projections

map-projections.net

The screenshot shows a web browser window with the title bar "Compare Map Projections" and the URL "map-projections.net". The page itself is titled "Compare Map Projections" and features a large image of two world maps side-by-side. A navigation menu at the top includes "Start", "Compare", "Single View", "Articles", "Links", and "About". The "Compare" menu is currently active and has a dropdown menu open with options: "via Selection Form" (selected), "via Thumbnails", and "via List of Projections". On the right side of the page, there is a "Viewing options" section with checkboxes for "South up" and "Flat oceans", and a "Apply" button. Below this, a message says "Last Update: Apr 12, 2017 (see below)". A large callout box on the left side of the page contains the text "Comparing map projections is like comparing apples and oranges. Probably there'll be a lot of mistakes here. Sorry!".

## What?

This site is about [map projections](#), specifically about world map projections.

A map projection is needed to show the spherical surface of the earth on a flat map (see [What's a Map Projection?](#)). There is some info about projections ([see below](#)), but more importantly, it's about their appearance: Out of more than 150 different map projections you can select two at a time to view their differences and similarities in a direct comparison.

## Why?

Sometimes, two map projections might look so similar that it's hard to tell the difference.

Sometimes, you might see the difference but want to study it more closely.

Maybe you want to compare the distortions which are present in each and every map projection.