



Data Visualization with matplotlib and seaborn

Bruno Gonçalves

*www.data4sci.com/newsletter
graphs4sci.substack.com*

<https://github.com/DataForScience/DataViz>



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)

Question

- How did you hear about this webinar?

- O'Reilly Platform
- Newsletter
- Website
- Previous event
- Other?



Table of Contents

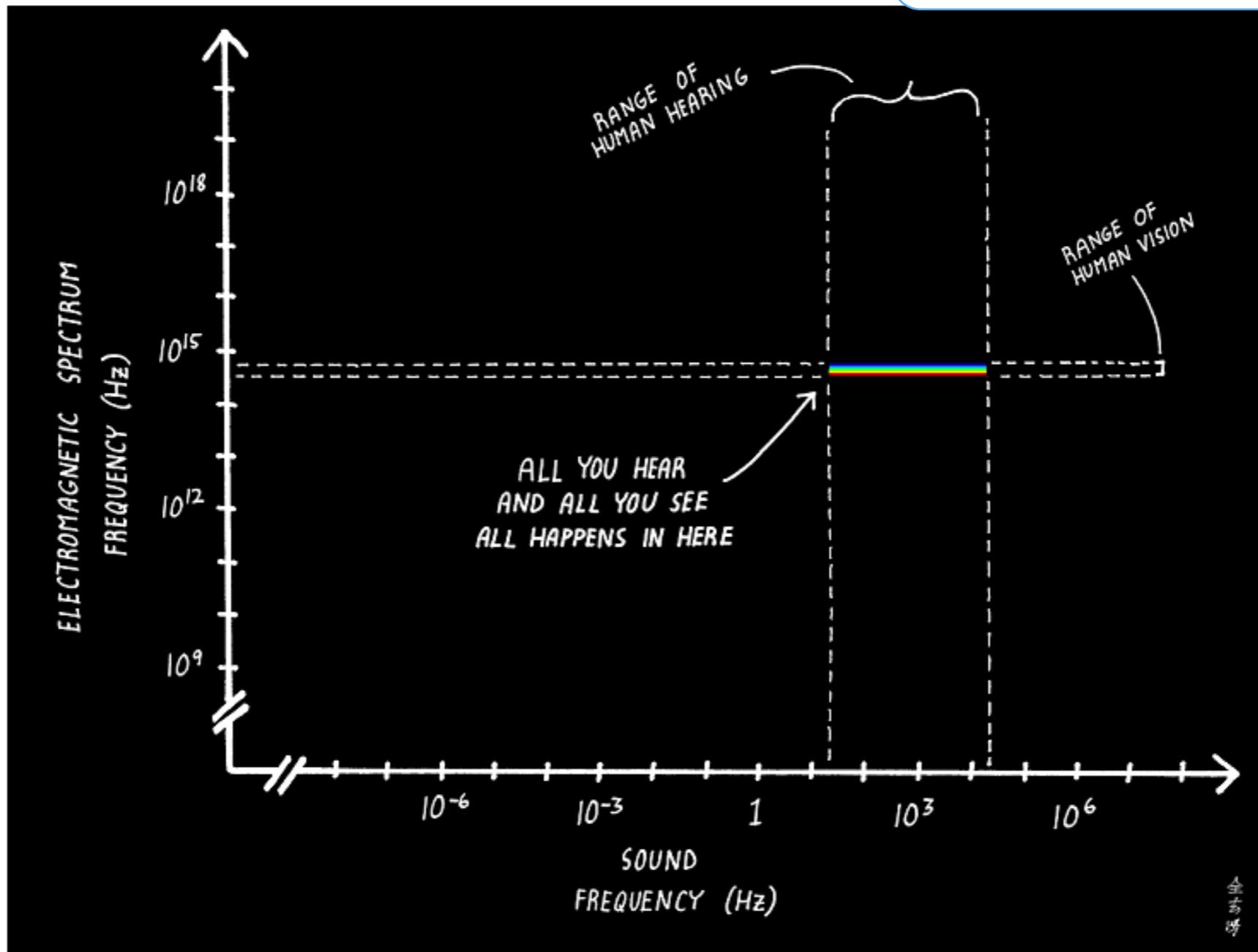
1. Human Perception
2. Visualization
3. Matplotlib
4. Style sheets
5. Mapping
6. Seaborn



1. Human Perception

Human Perception

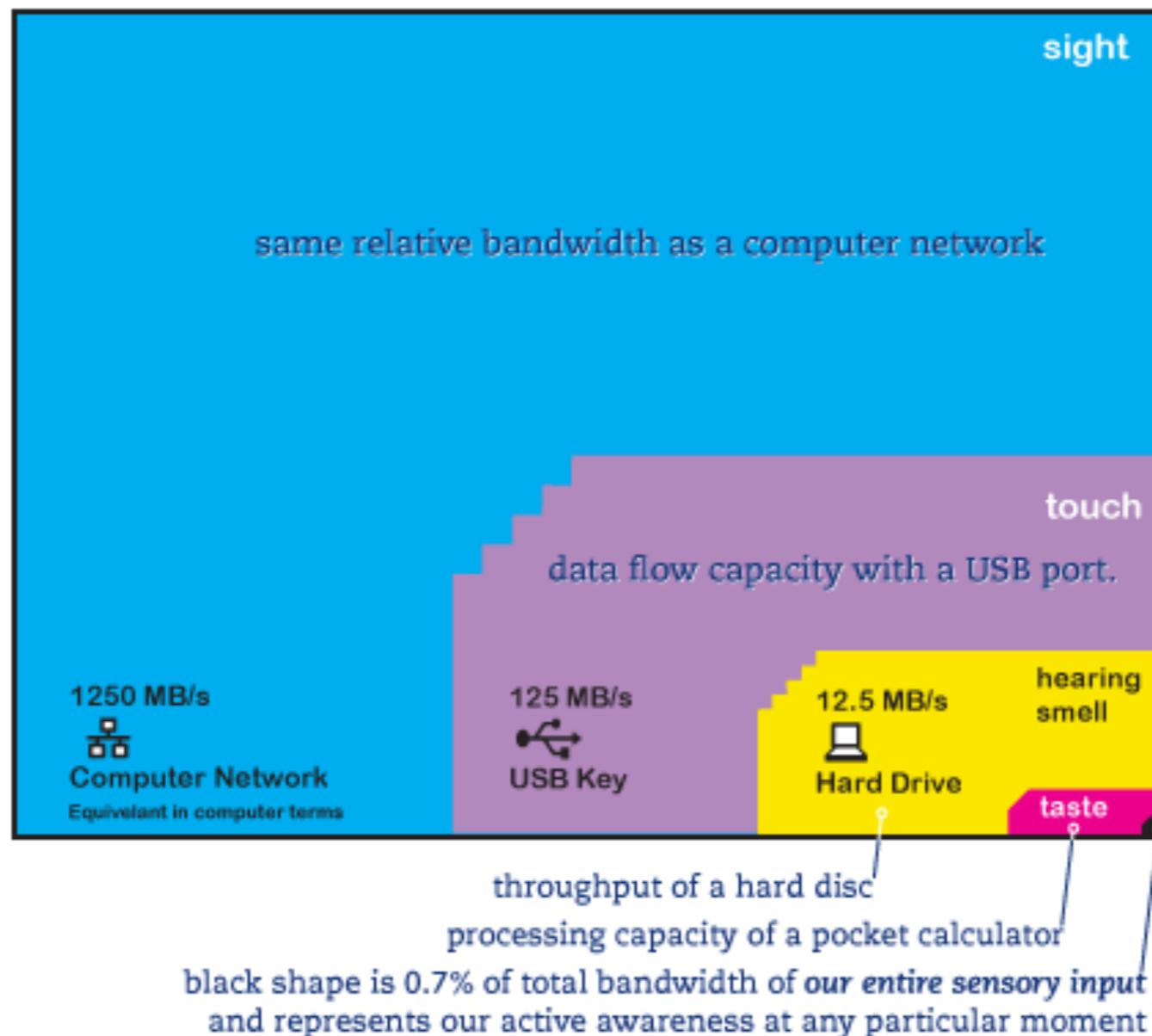
abstrusegoose.com/421



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

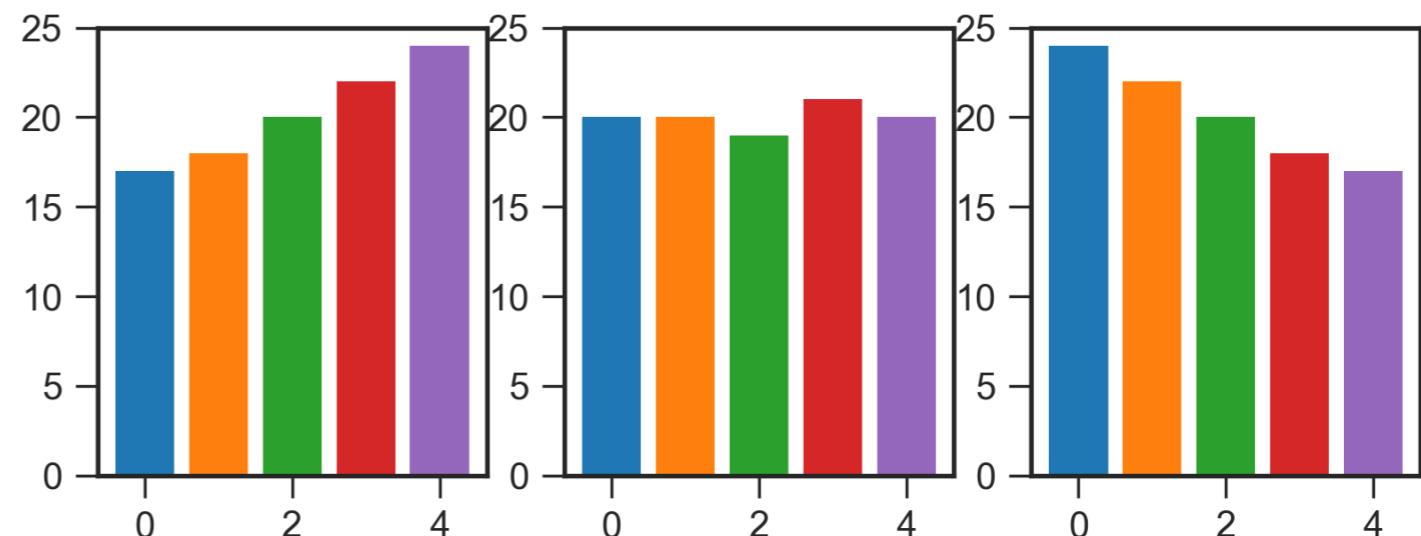
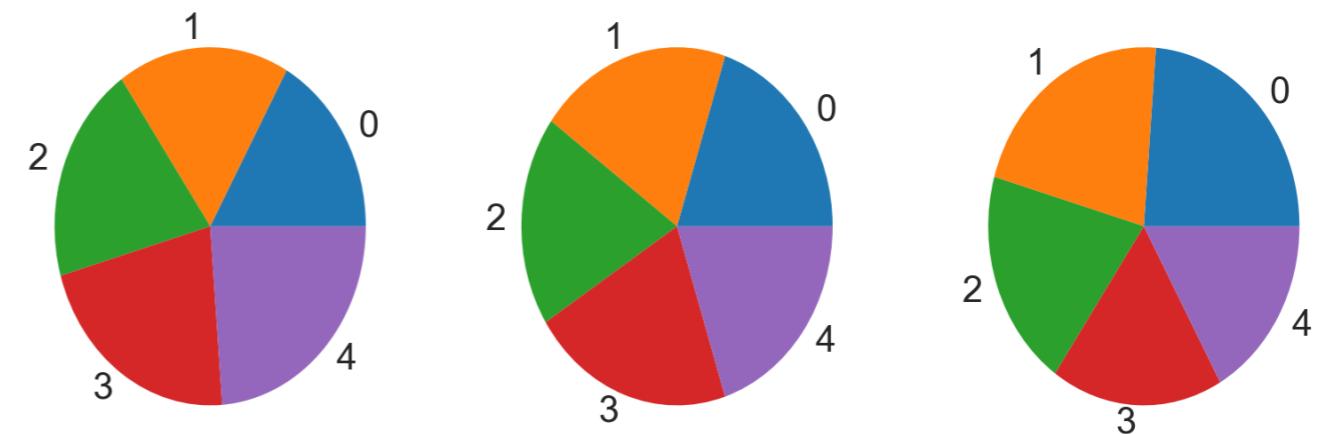
Perception

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

Perception

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

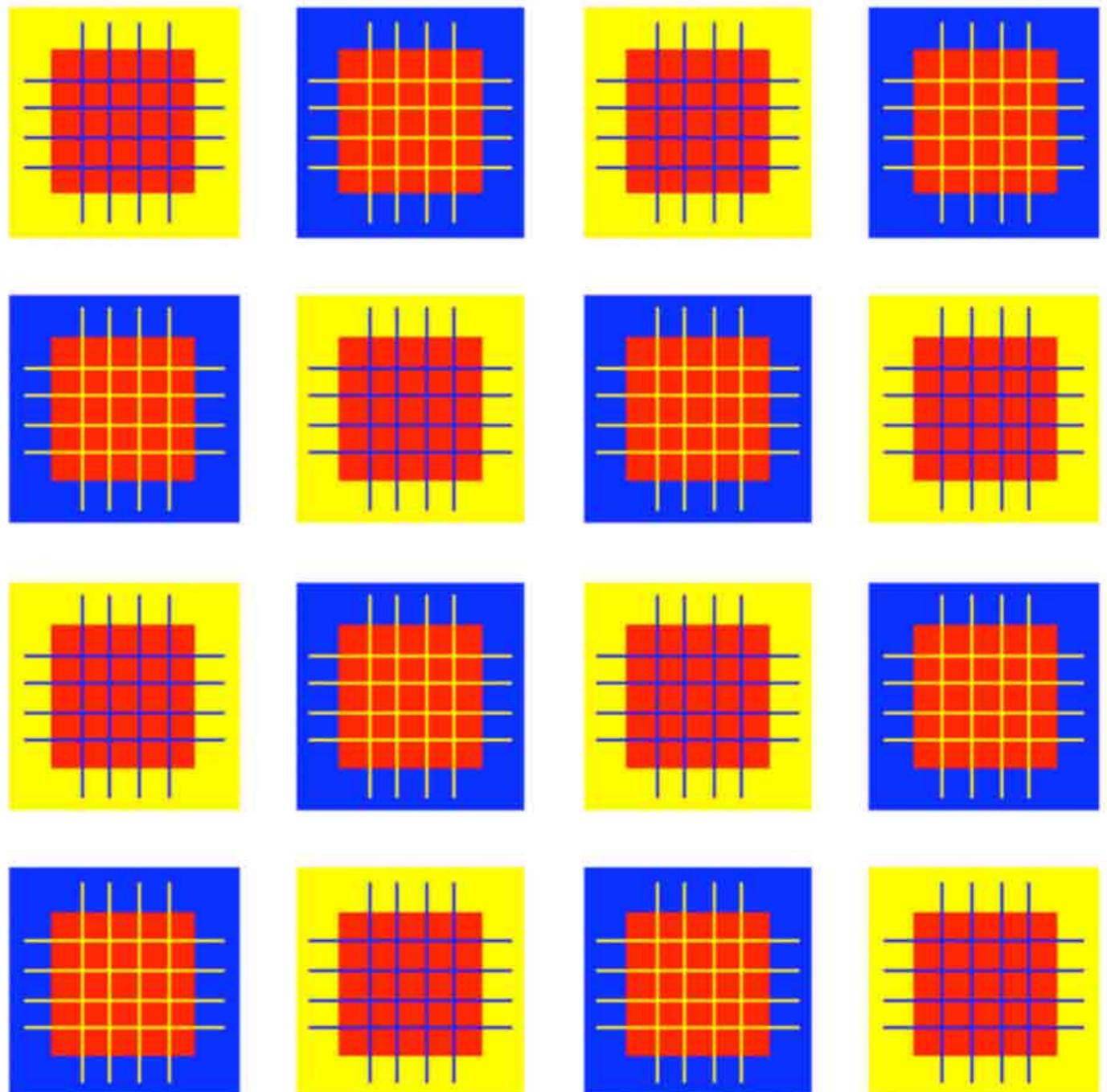
- Position, length
- Direction, Angle, Area
- Volume, Curvature, Shade
- Color Saturation.



Perception

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

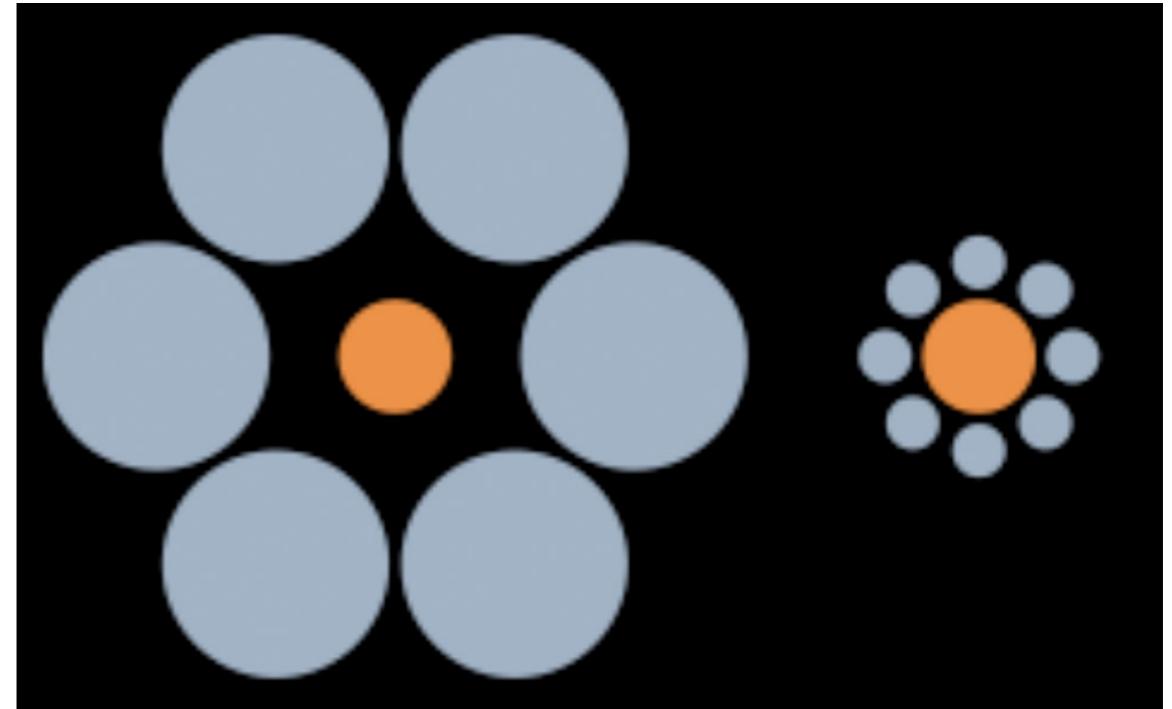
- Position, length
- Direction, Angle, Area
- Volume, Curvature, Shade
- Color Saturation.



Perception

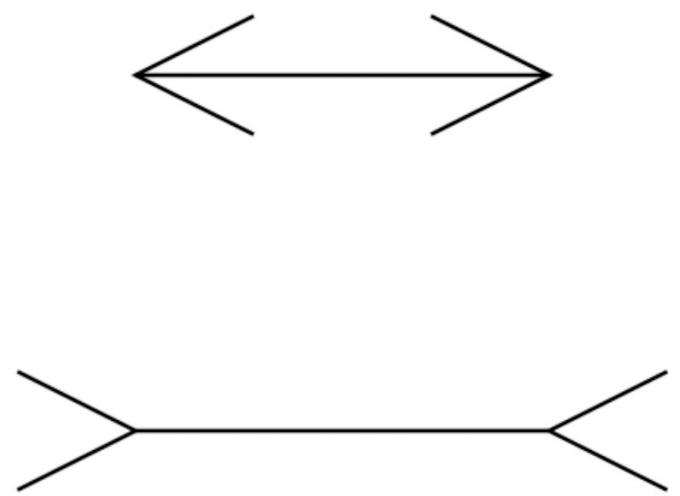
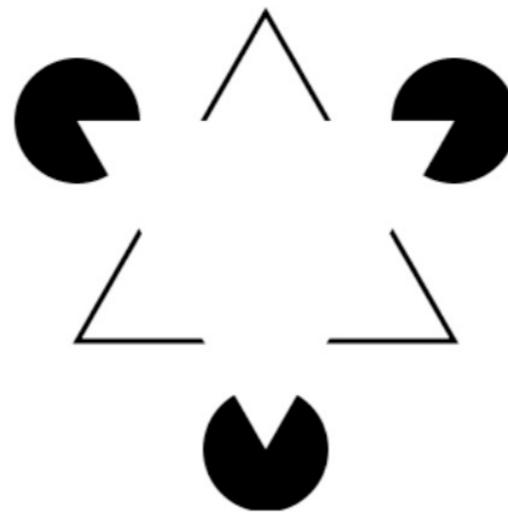
- Context also matters!

- 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"



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

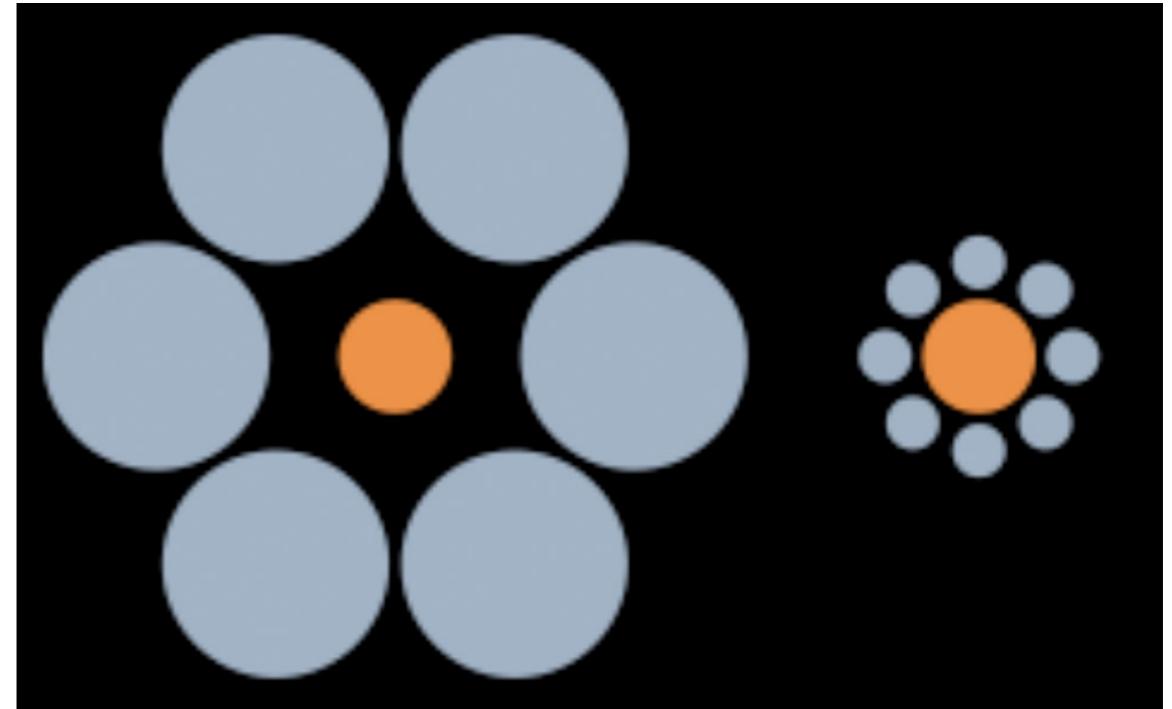
- Position, length
- Direction, Angle, Area
- Volume, Curvature, Shade
- Color Saturation.



Perception

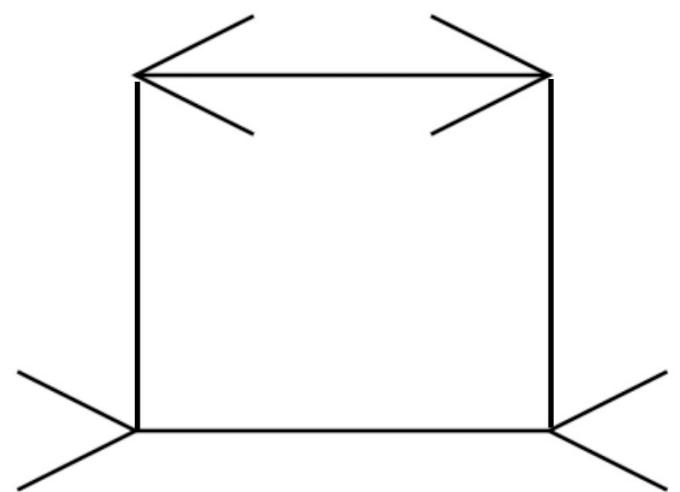
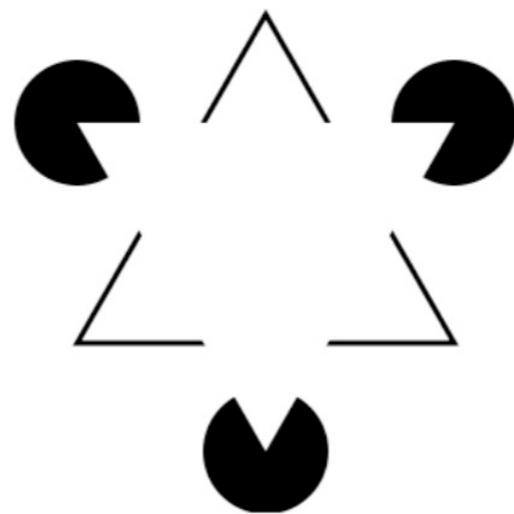
- Context also matters!

- 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"



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

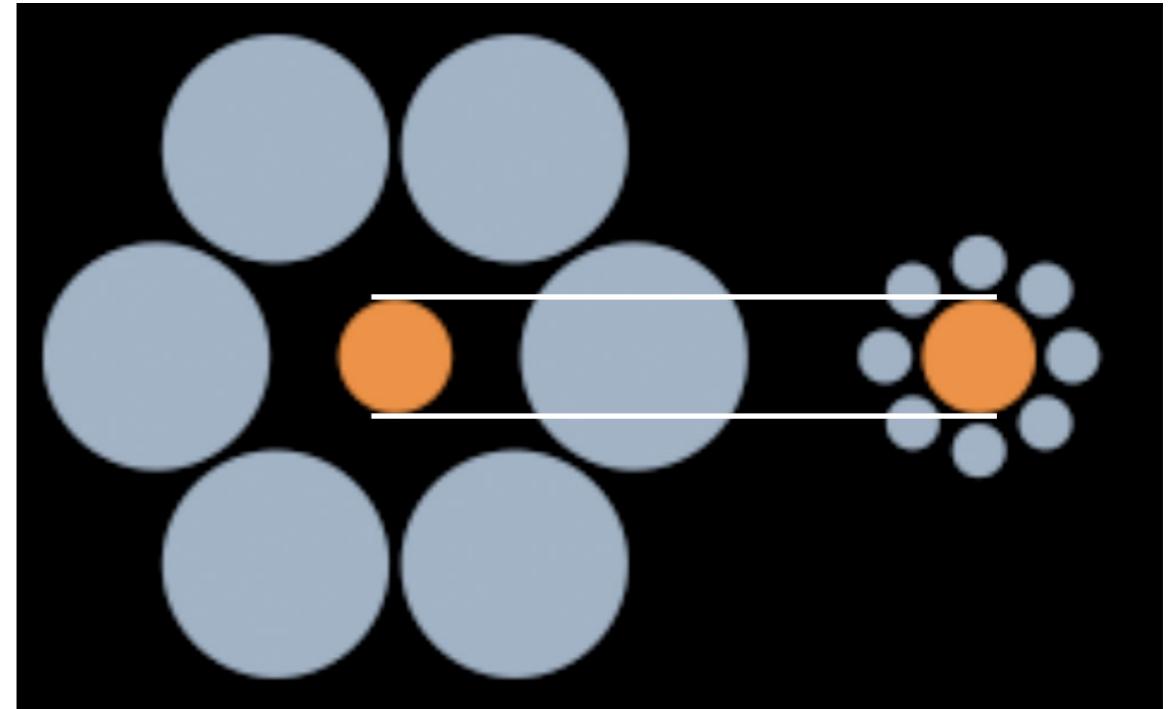
- Position, length
- Direction, Angle, Area
- Volume, Curvature, Shade
- Color Saturation.



Perception

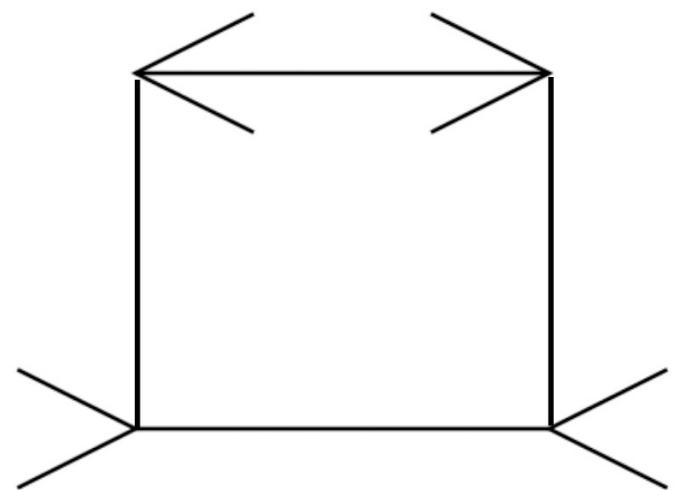
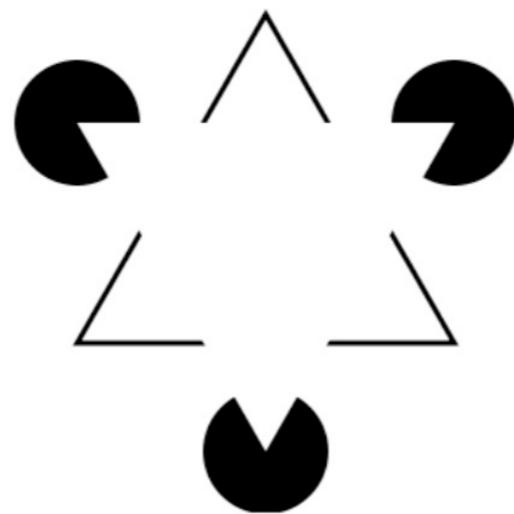
- Context also matters!

- 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"

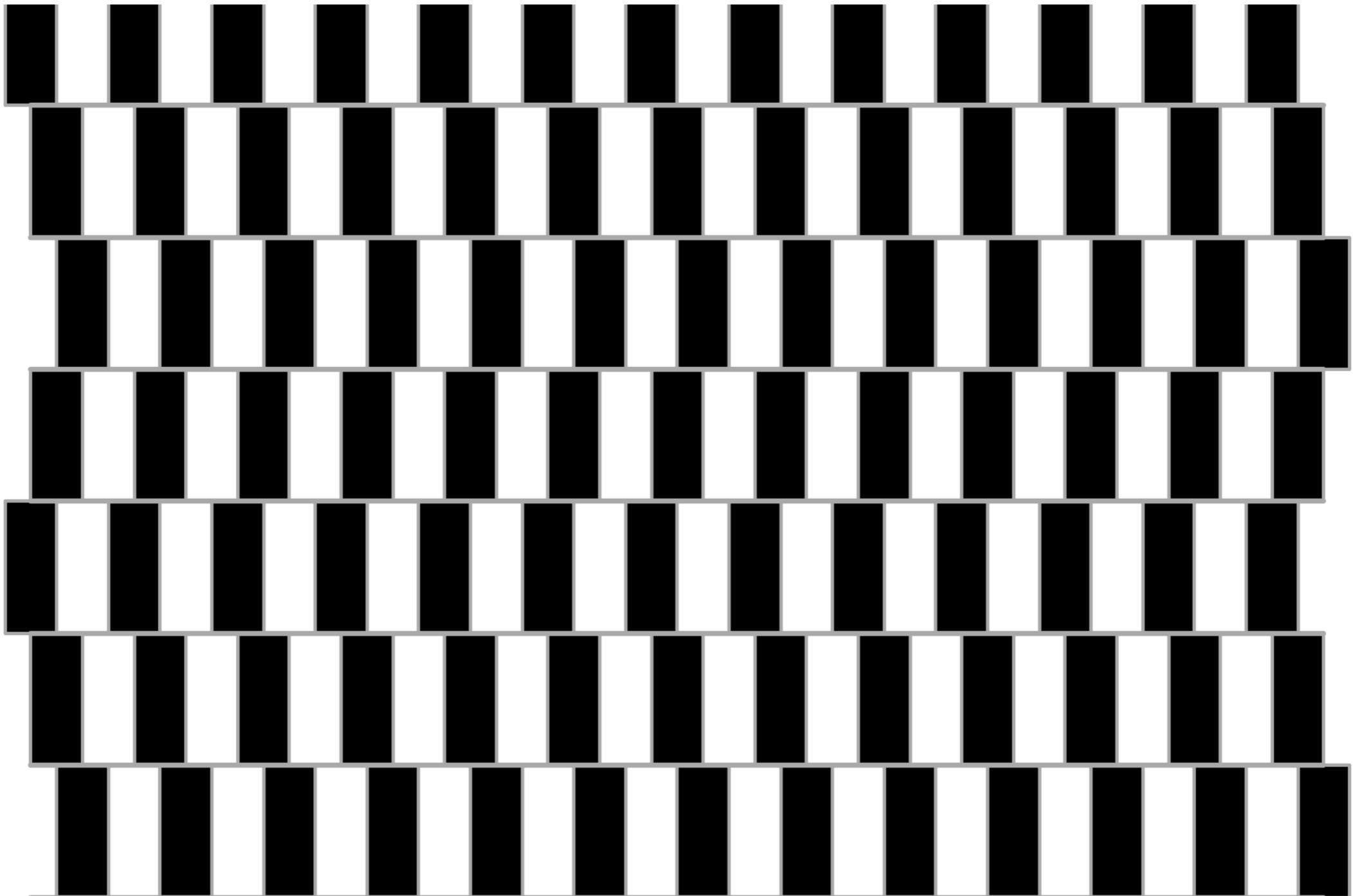


- Some cognitive tasks are significantly easier than others. In order, we are good at distinguishing:

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

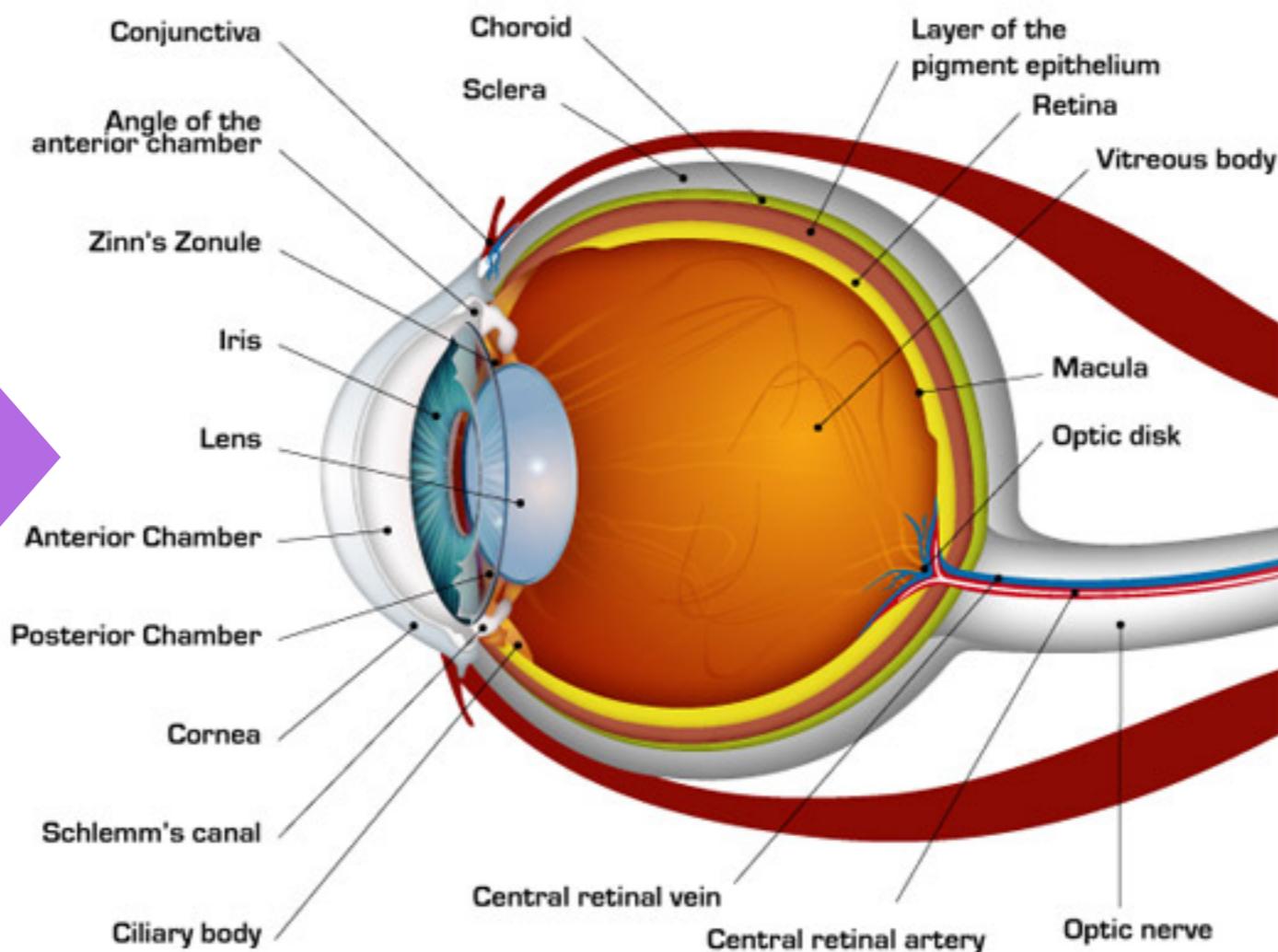
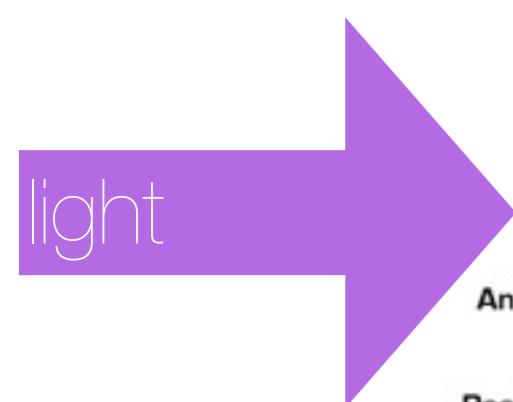


Perception Biases



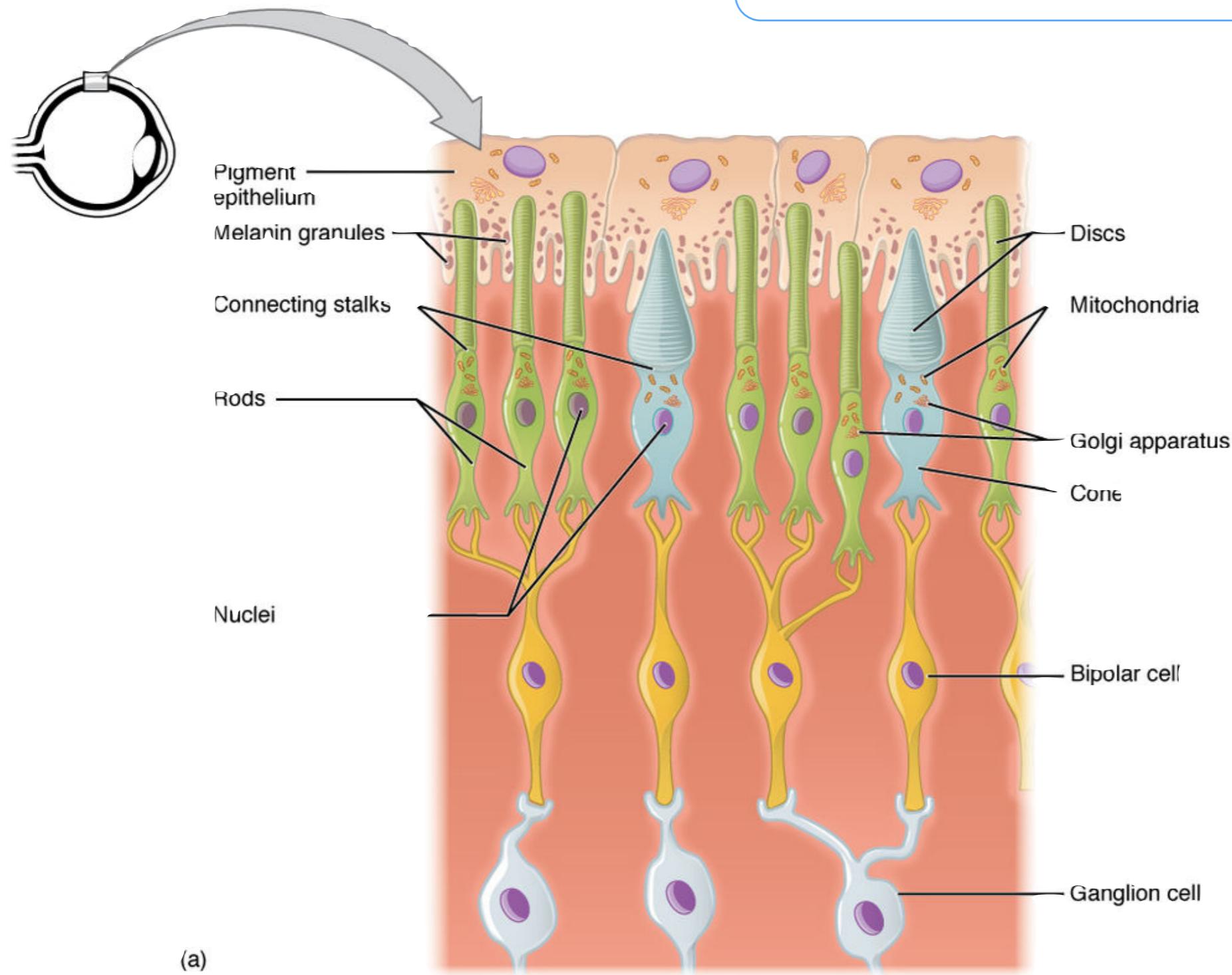
Human Vision

en.wikipedia.org/wiki/Photoreceptor_cell



Human Vision

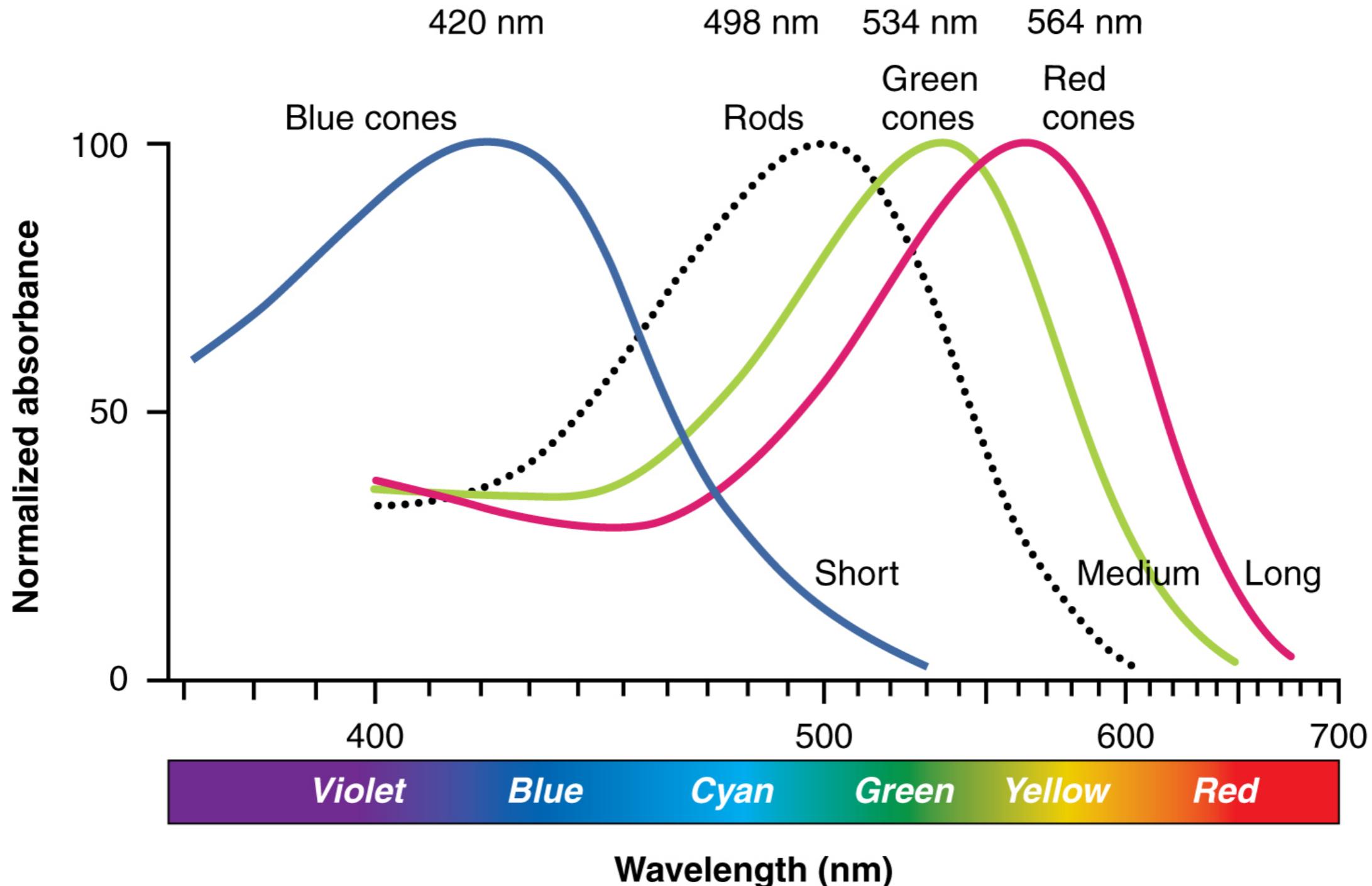
en.wikipedia.org/wiki/Photoreceptor_cell

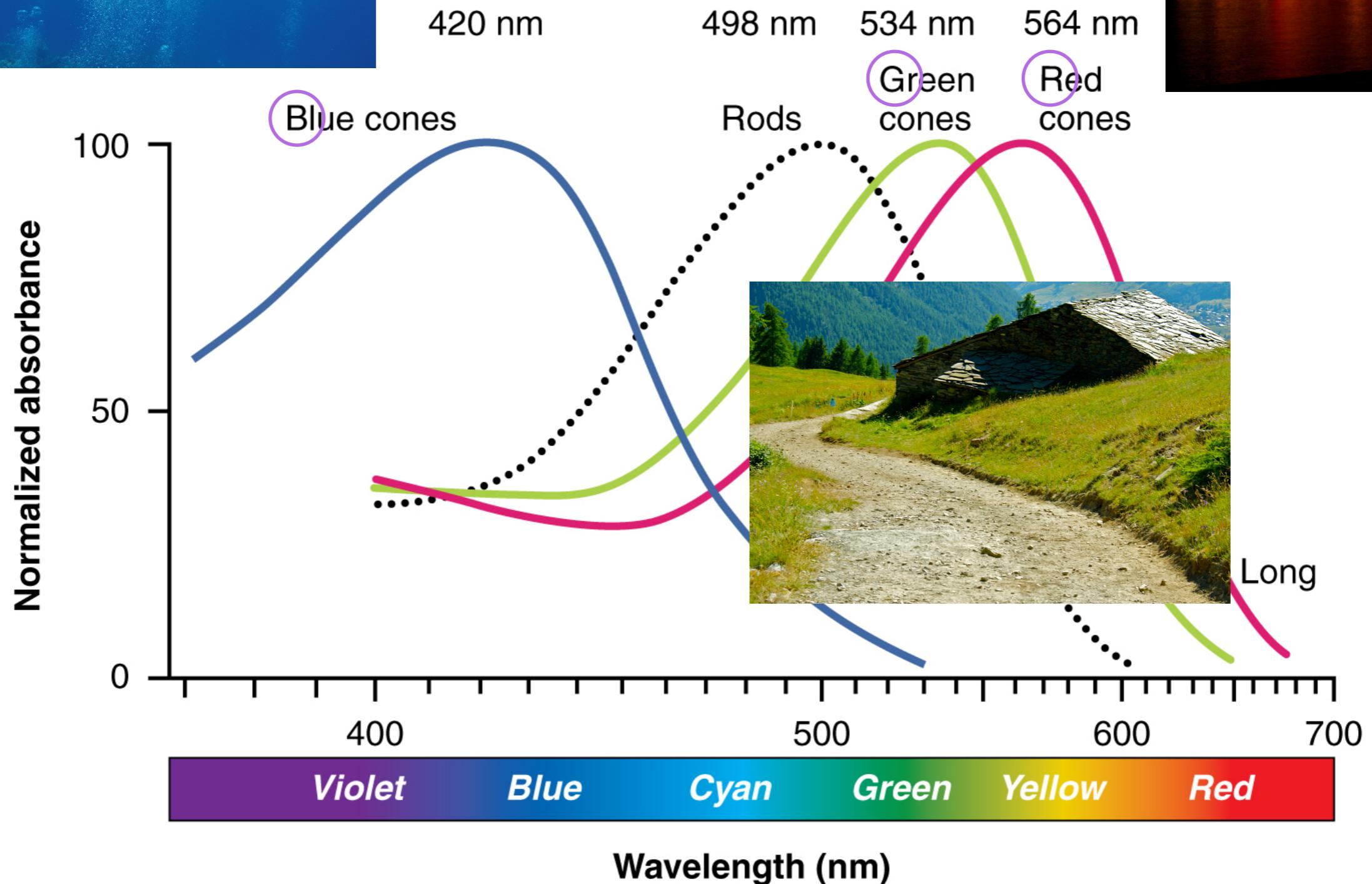
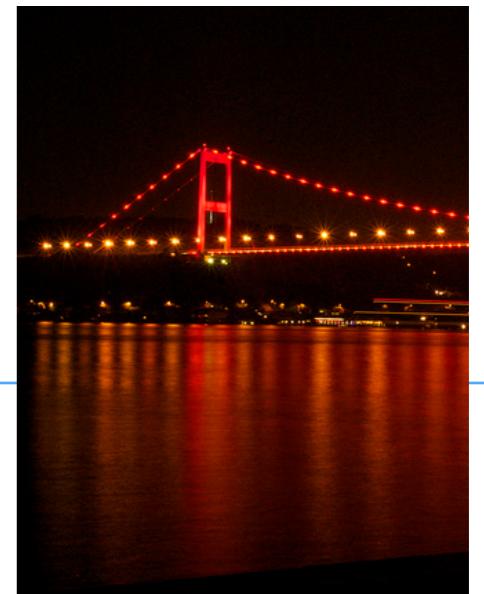
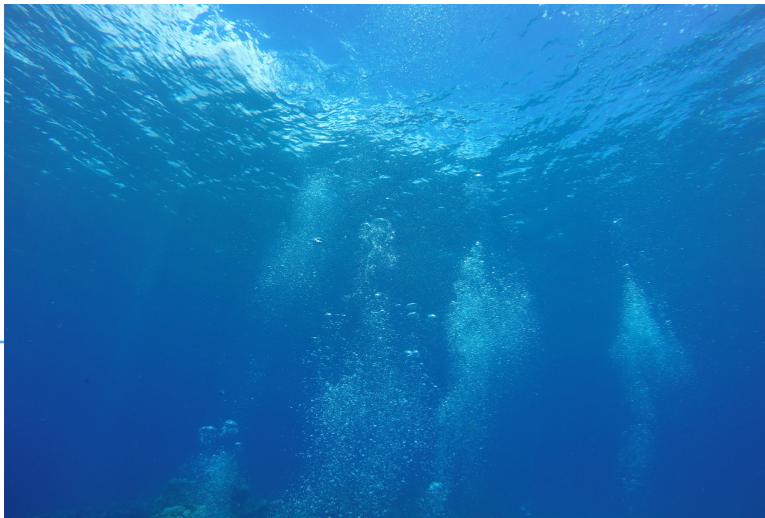


(a)

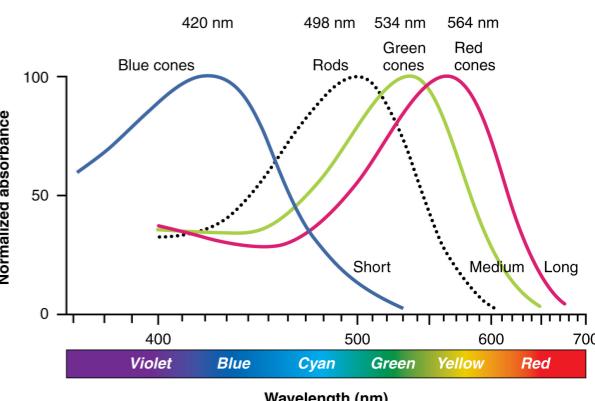
Human Vision

en.wikipedia.org/wiki/Photoreceptor_cell

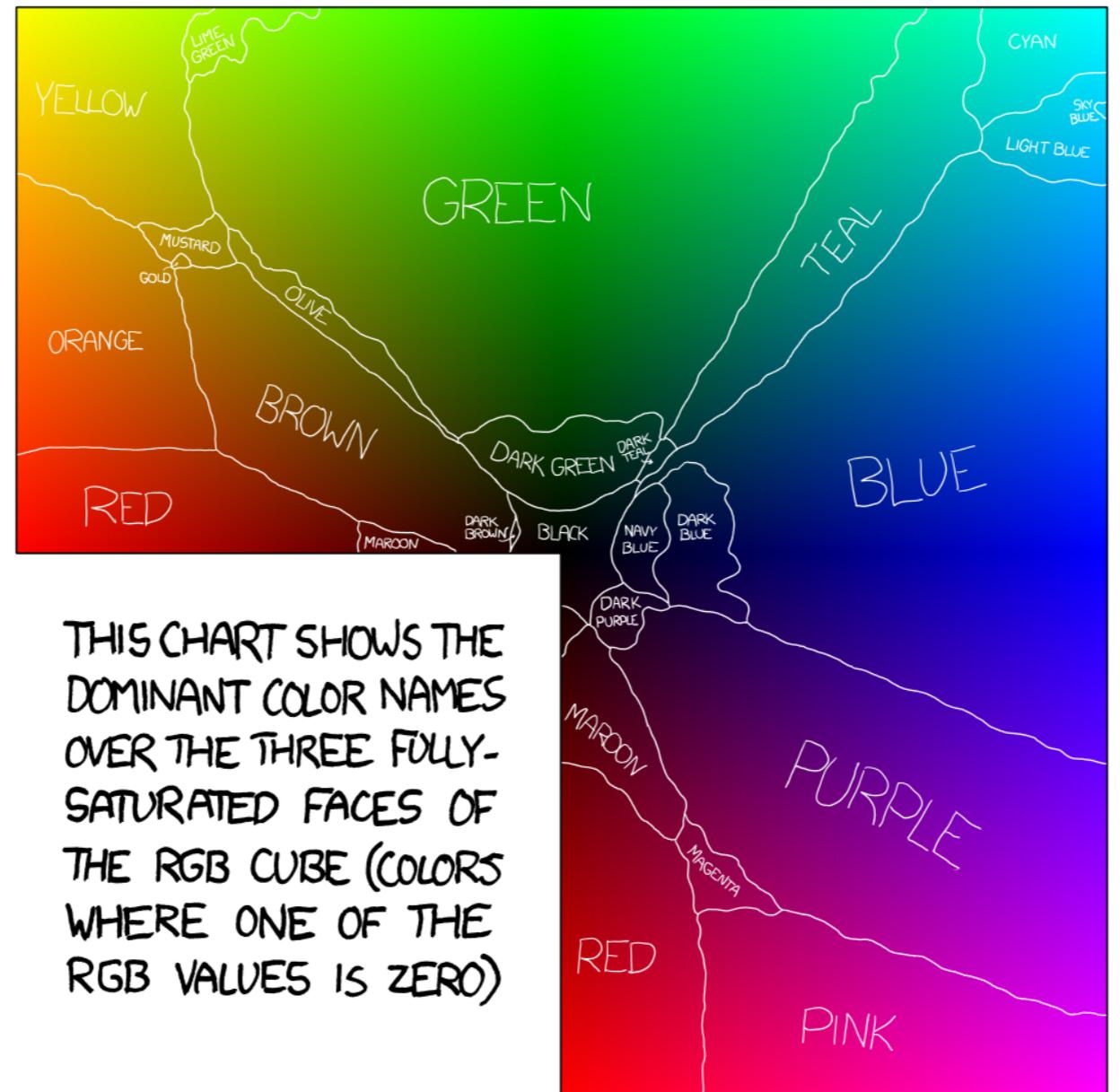
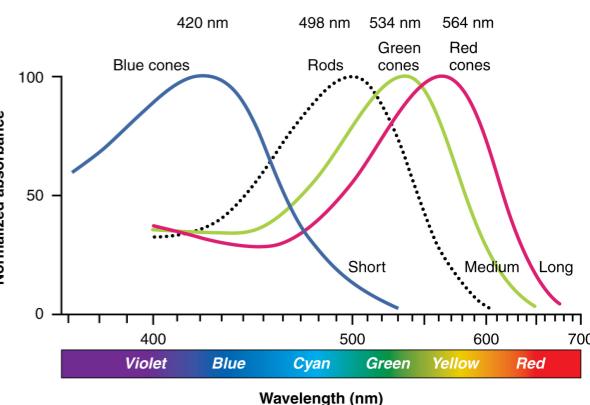




Colors galore!

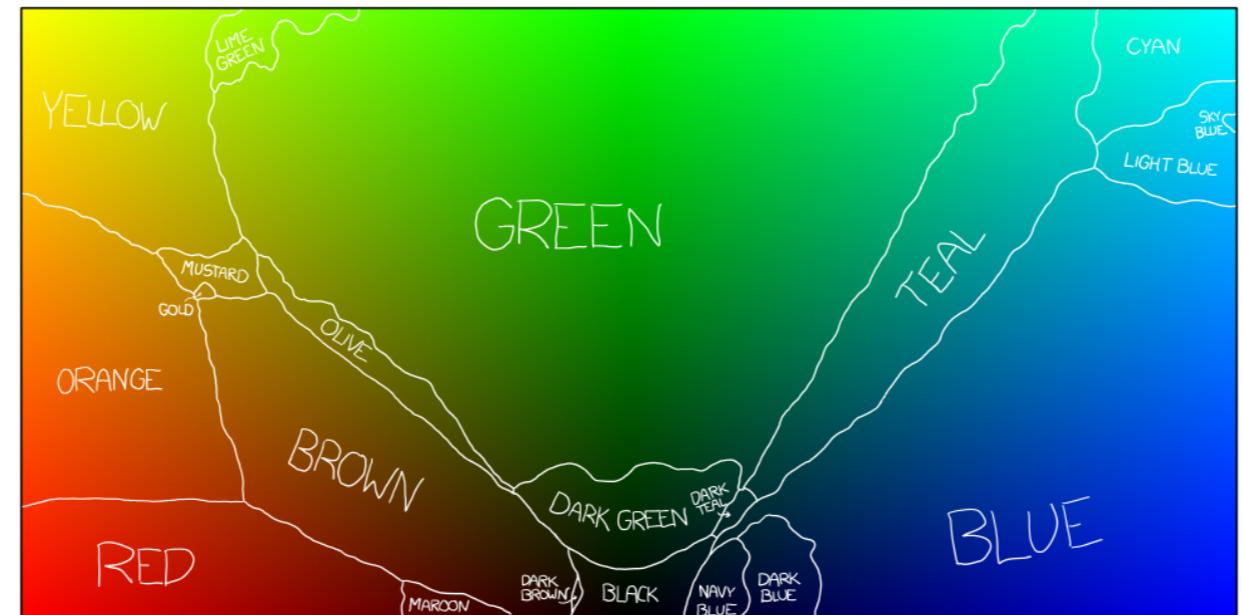
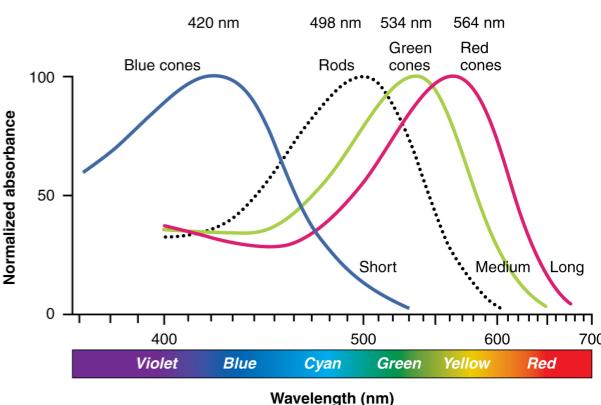


Color Perception



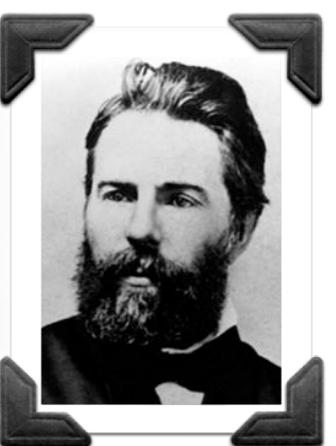
THIS CHART SHOWS THE DOMINANT COLOR NAMES OVER THE THREE FULLY-SATURATED FACES OF THE RGB CUBE (COLORS WHERE ONE OF THE RGB VALUES IS ZERO)

Color Perception

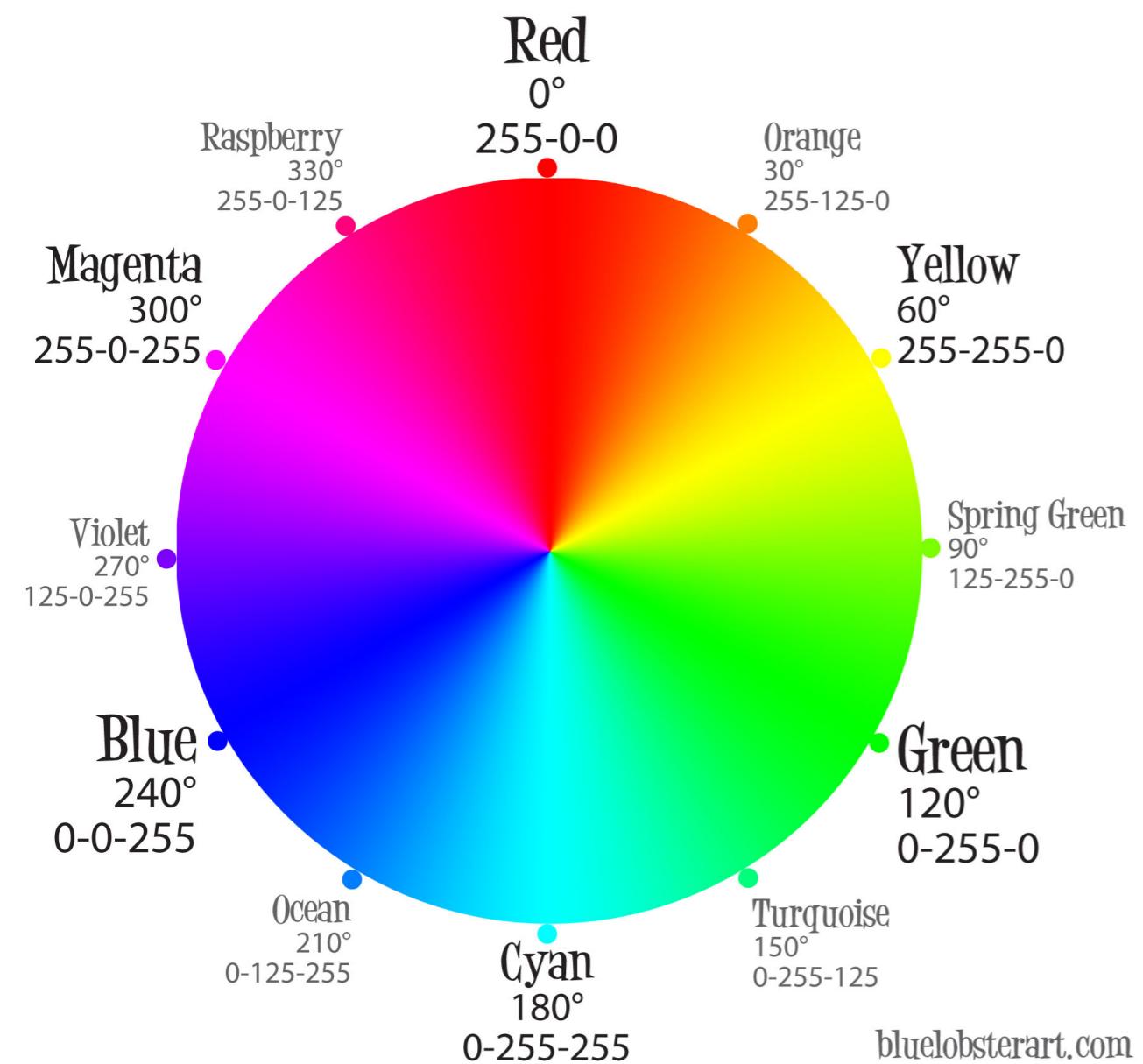
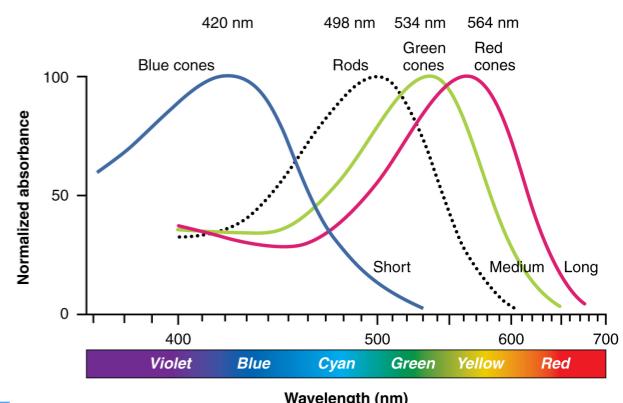


THIS CHART SHOWS THE DOMINANT COLOR NAMES OVER THE THREE FULLY-SATURATED FACES OF THE RGB CUBE (COLORS WHERE ONE OF THE RGB VALUES IS ZERO)

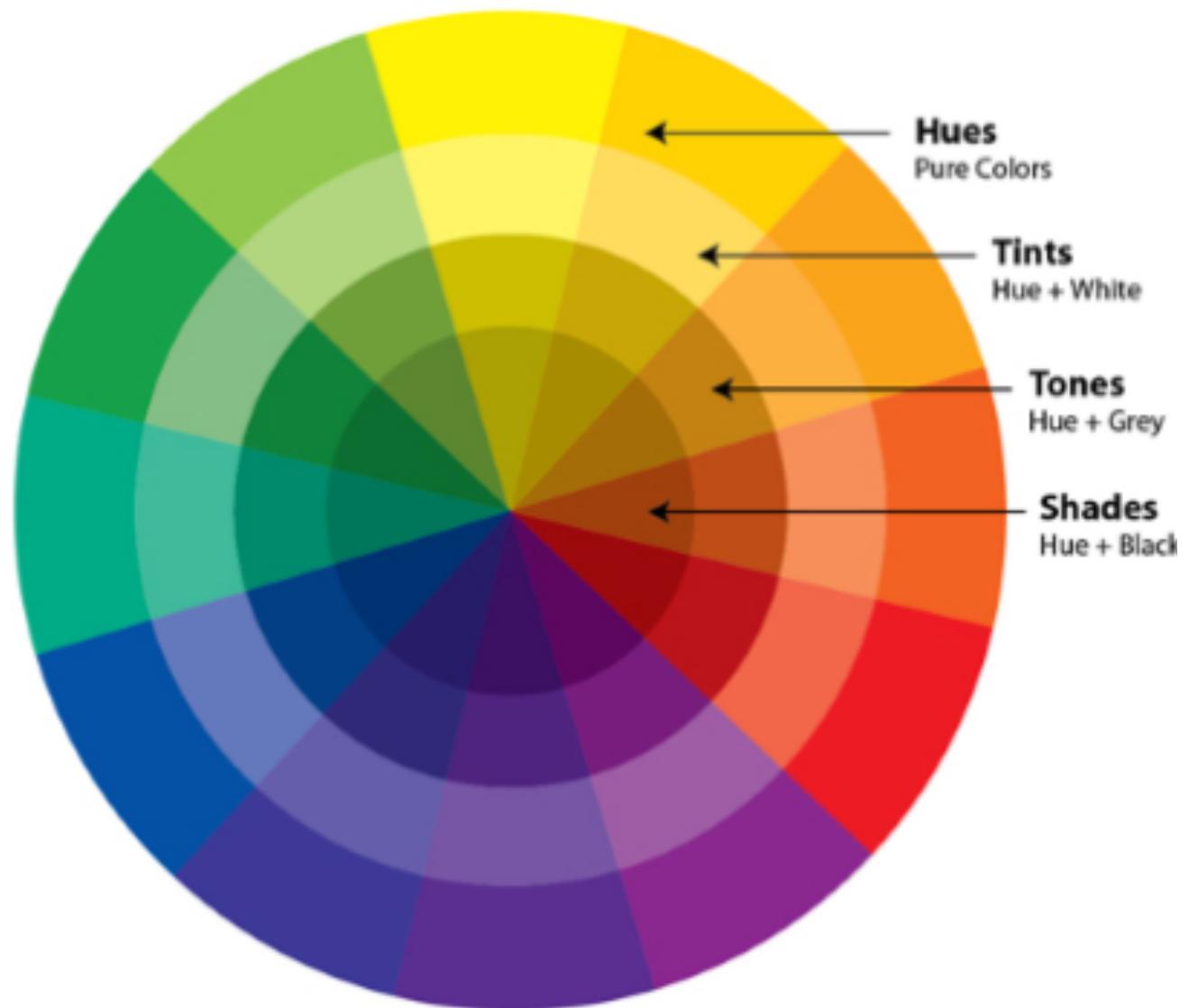
"Who in the rainbow can draw the line where the violet tint ends and the orange tint begins? Distinctly we see the difference of the colors, but where exactly does the one first blendingly enter into the other? So with sanity and insanity."
(H. Melville)



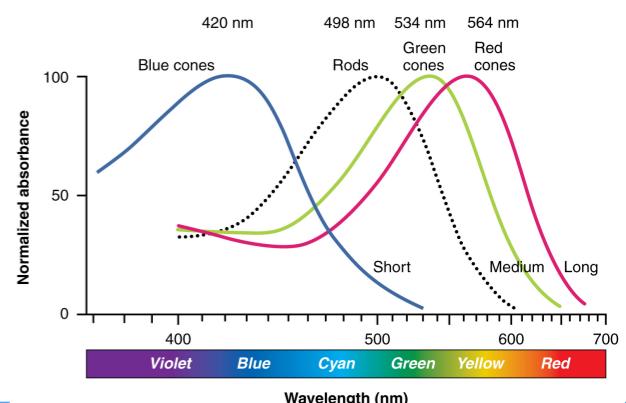
Color Wheel



bluelobsterart.com



Color Schemes

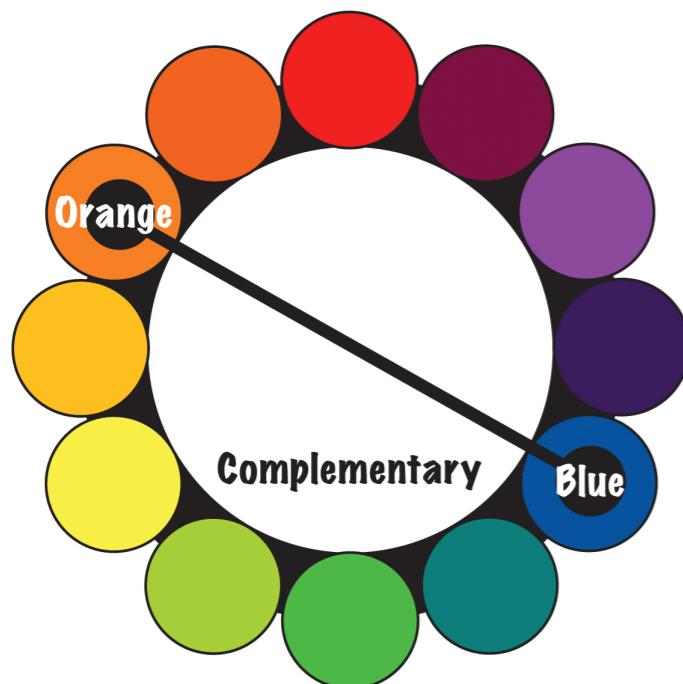
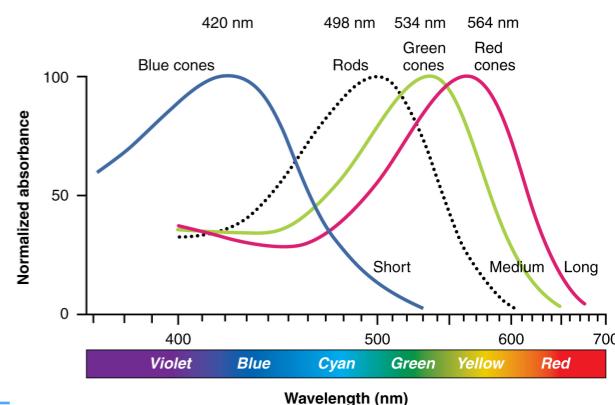


Warm Colors



Cold Colors

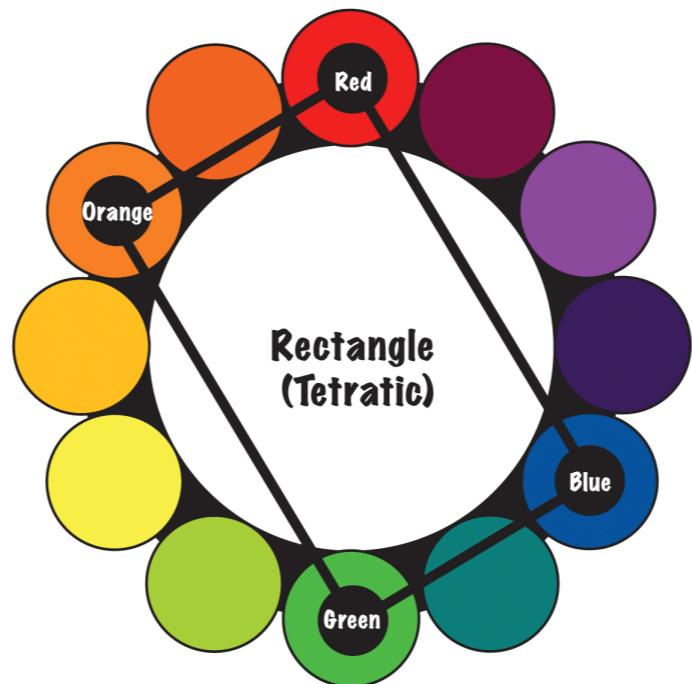
Color Schemes



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)

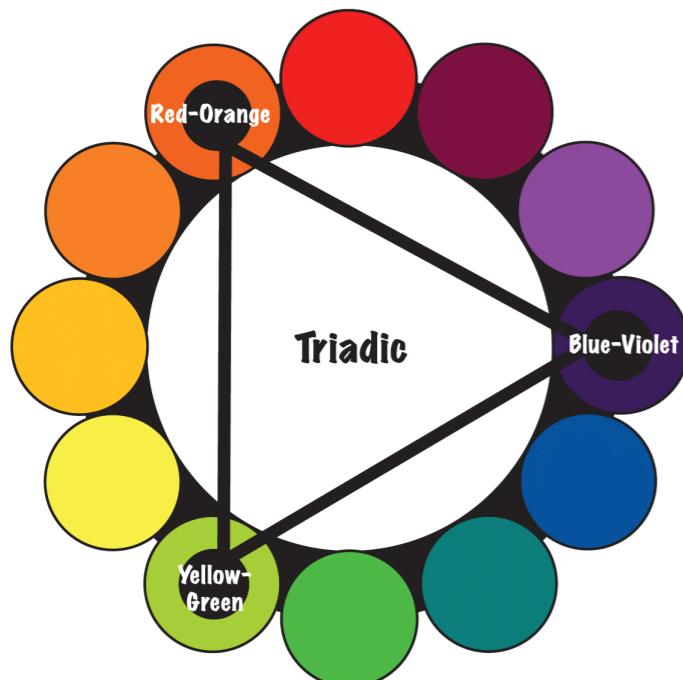
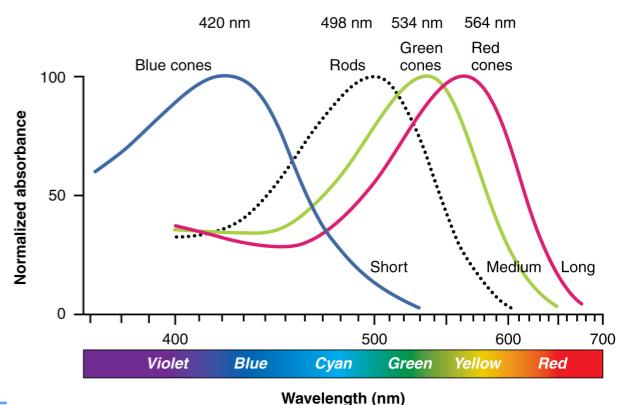


Analogous color scheme

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

(example: Green, Blue-Green and Blue)

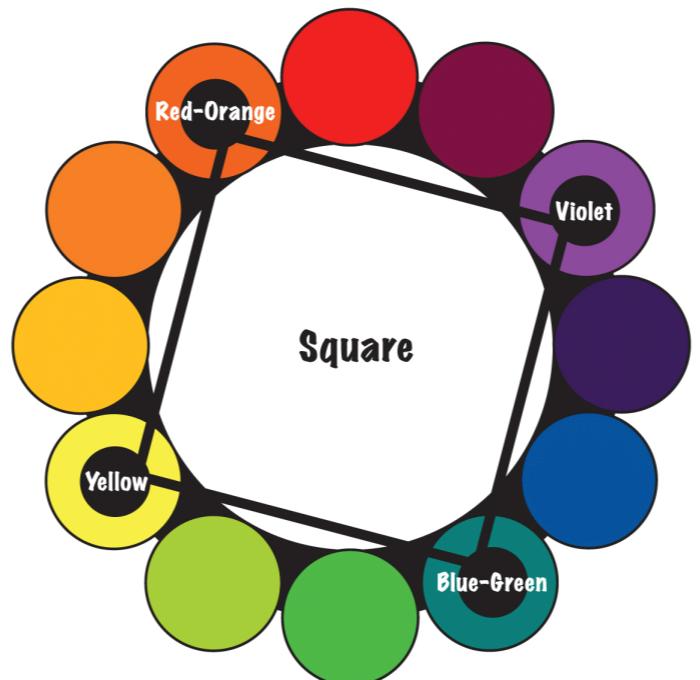
Color Schemes



Triadic color scheme

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

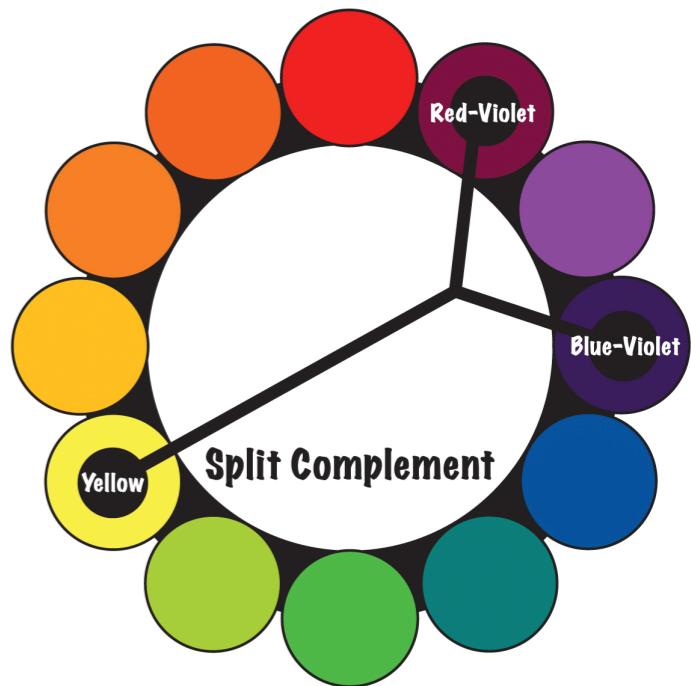
(example: Yellow-Green, Red-Orange and Blue-Violet)



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)

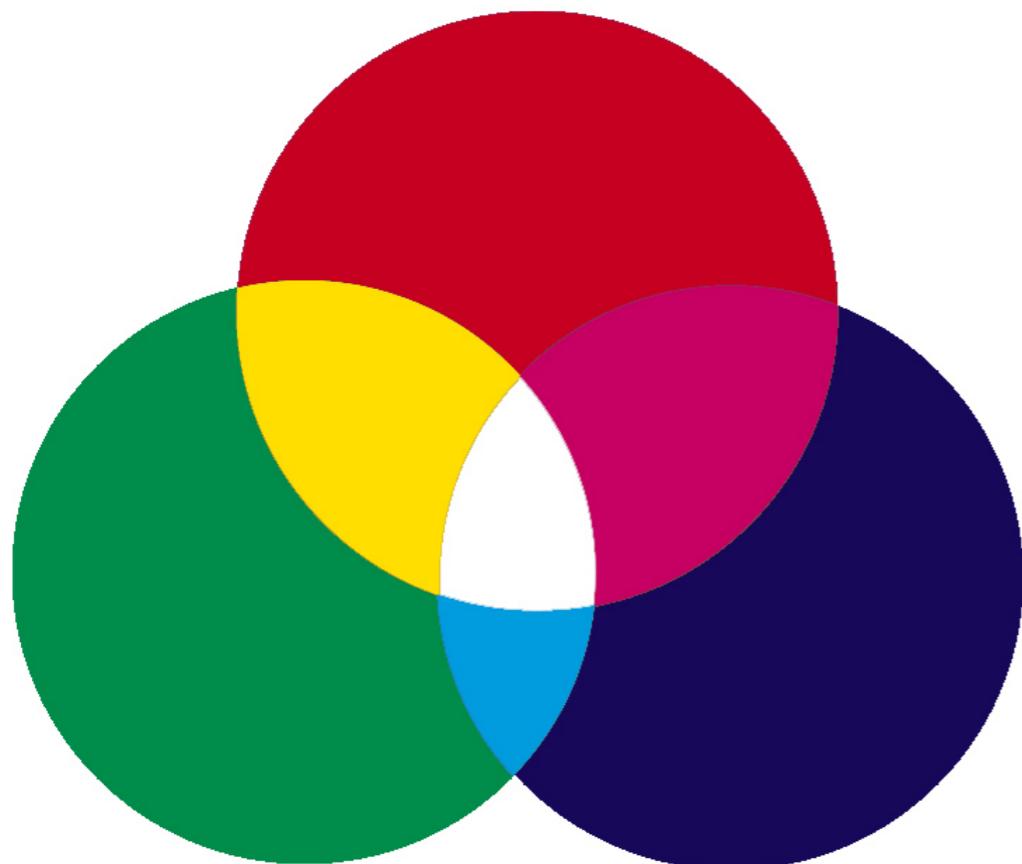
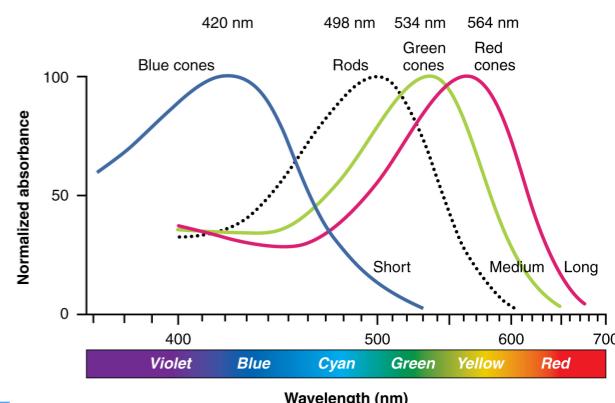


Split-Complementary color scheme

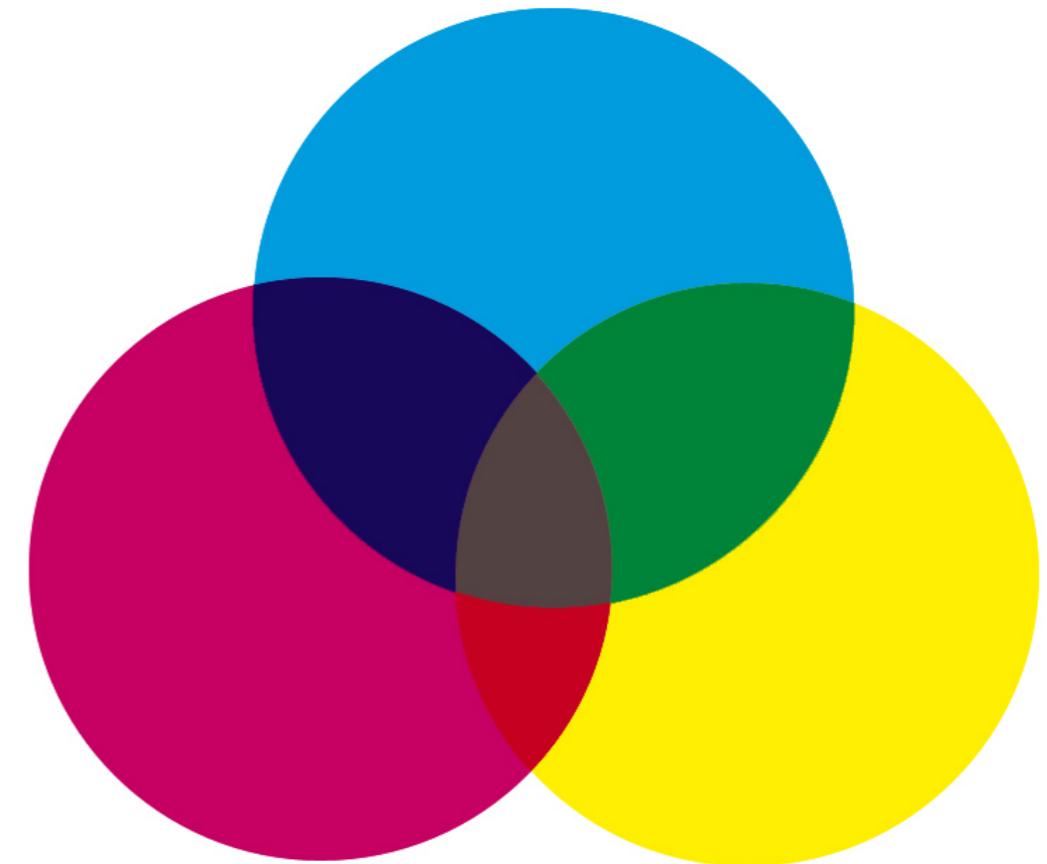
The split-complementary color scheme is a variation of the complementary color scheme. In addition to the base color, it uses the two colors adjacent to its complement.

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

Color Systems



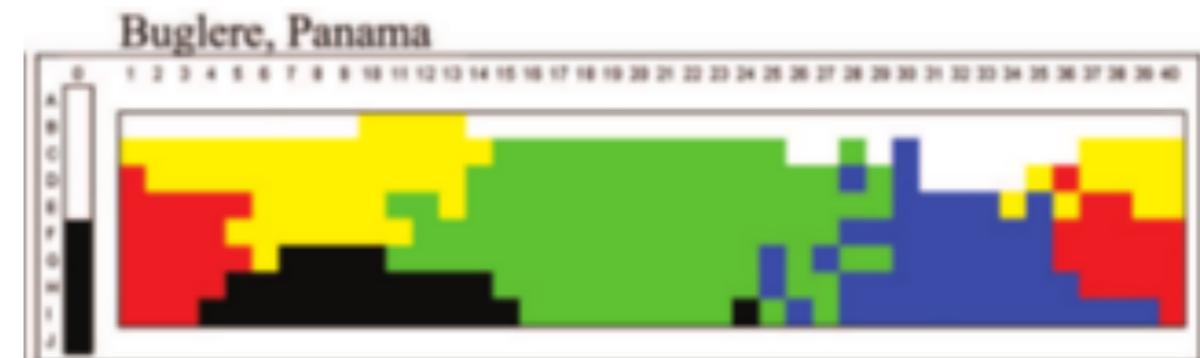
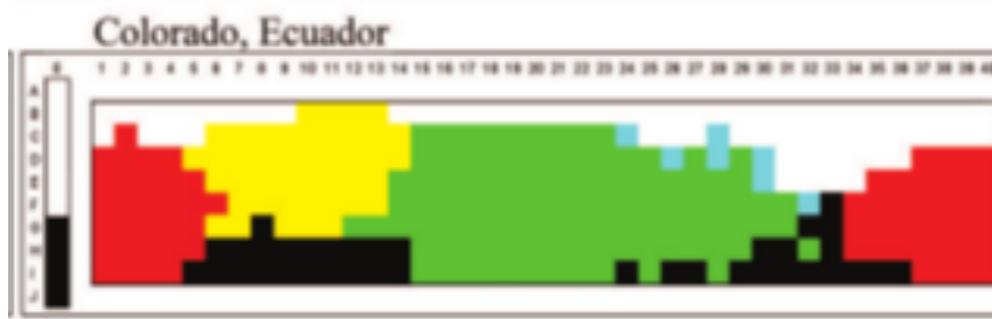
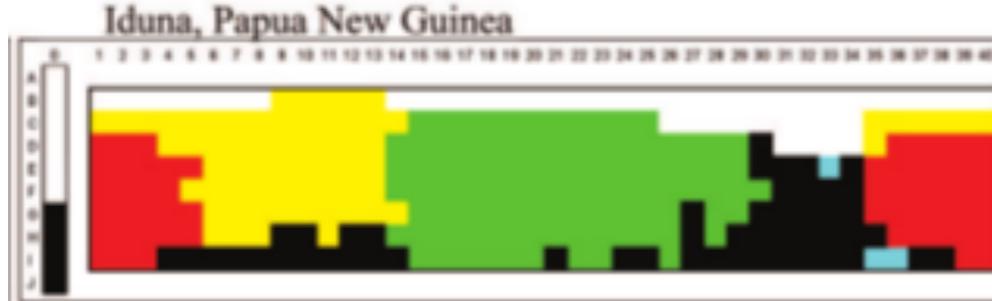
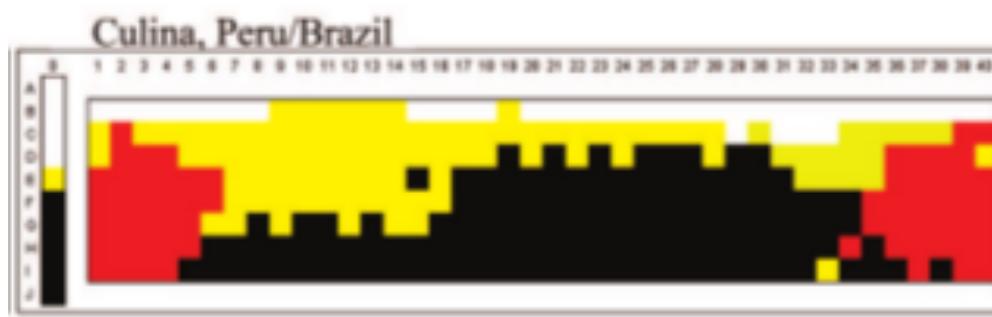
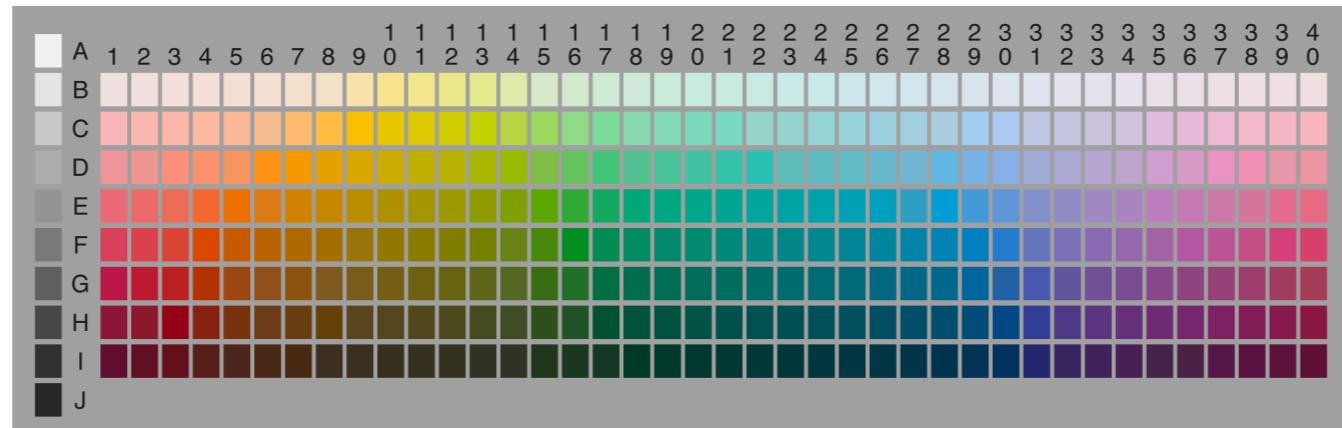
Additive Color (RGB)
Light



Subtractive color (CMYK)
Ink

Colors and Culture

www1.icsi.berkeley.edu/wcs/



Color Blindness



Color Blindness

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

<https://github.com/MaPePeR/jsColorblindSimulator/blob/master/colorblind.js>



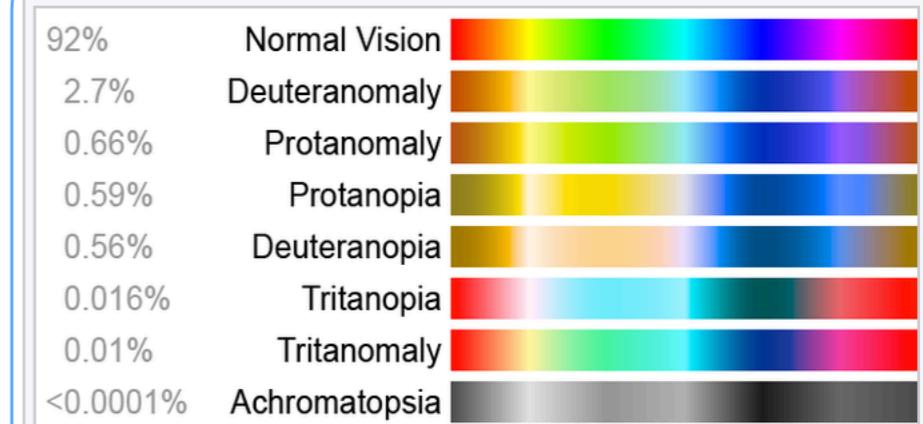
Color Blindness

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

<https://github.com/MaPePeR/jsColorblindSimulator/blob/master/colorblind.js>



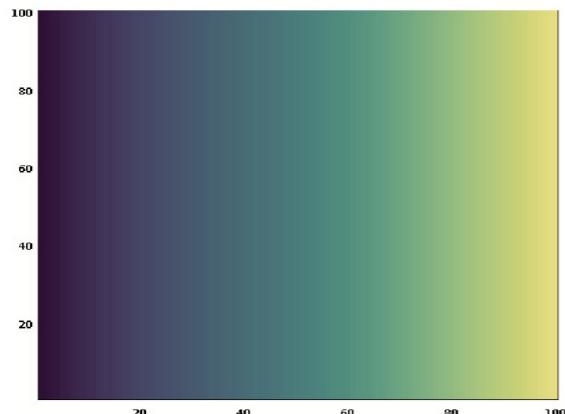
	Cone system	Red	Green	Blue
	N=normal A=anomalous	N	A	N
1	Normal vision	•	•	•
2	Protanomaly	•	•	•
3	Protanopia	•	•	•
4	Deuteranomaly	•	•	•
5	Deutanopia	•	•	•
6	Tritanomaly	•	•	•
7	Tritanopia	•	•	•
8	Achromatopsia	•	•	•
9	Tetrachromat	•	•	•
10		•	•	•



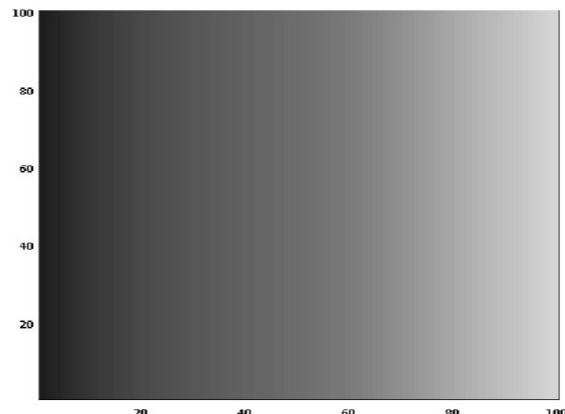
These color charts show how different colorblind people see compared to a person with normal color vision.

Viridis Color Scheme

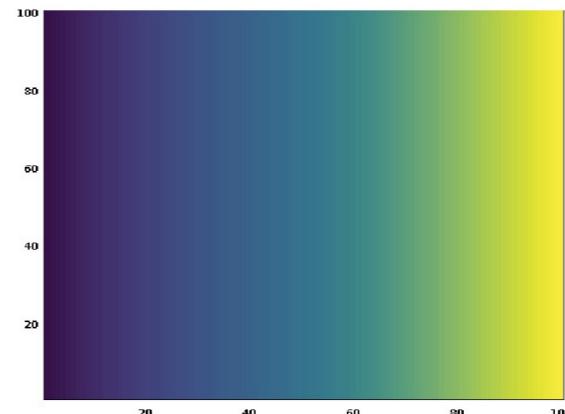
Achromatomaly



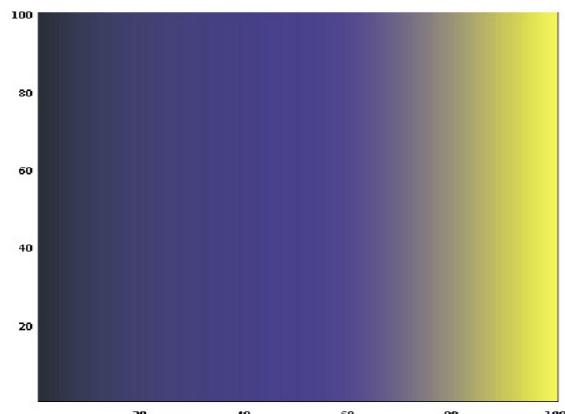
Achromatopsia



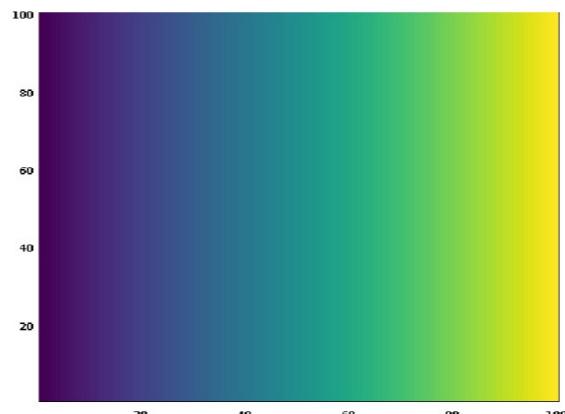
Deuteranomaly



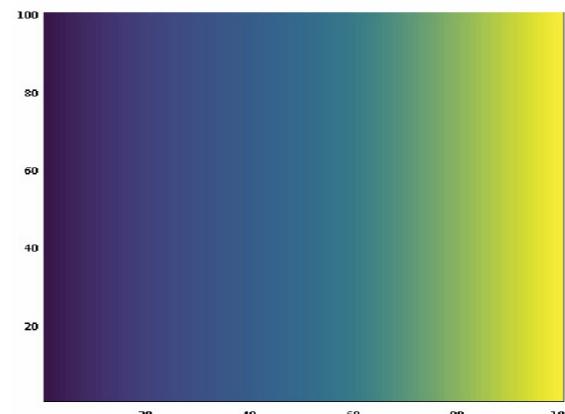
Deutanopia



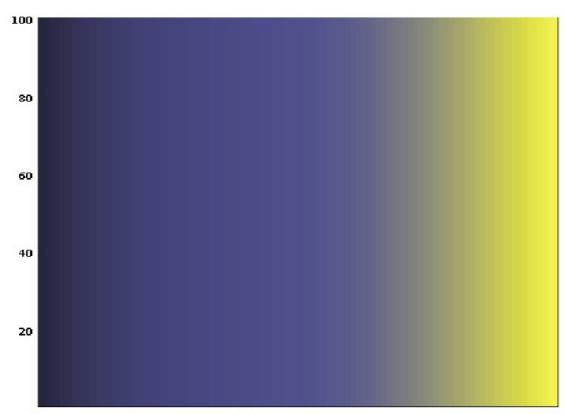
Normal



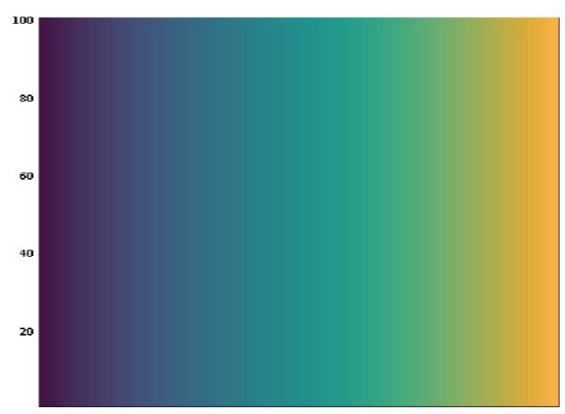
Protanomaly



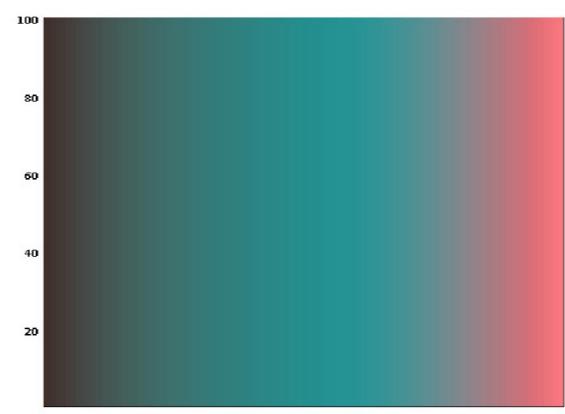
Protanopia



Tritanomaly



Tritanopia



Color Scheme Choosers

colorhunt.co

Color Hunt Palettes New ▾



36

Today



97

Yesterday



249

2 days



228

3 days



311

4 days



300

5 days



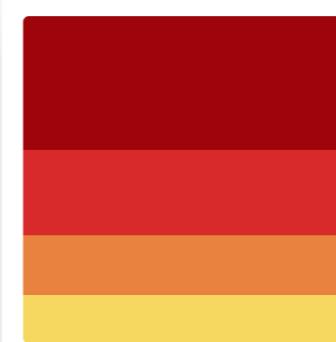
283

6 days



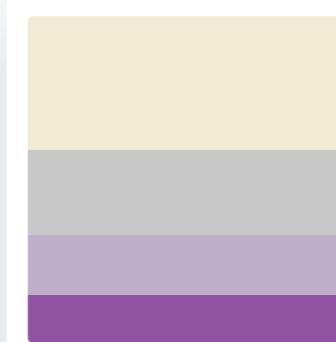
342

1 week



356

1 week



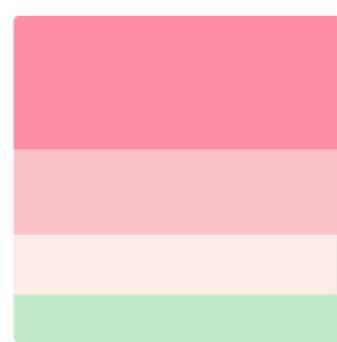
372

1 week



292

1 week



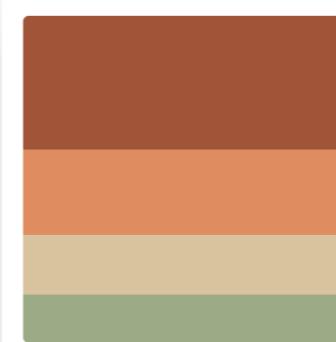
510

1 week



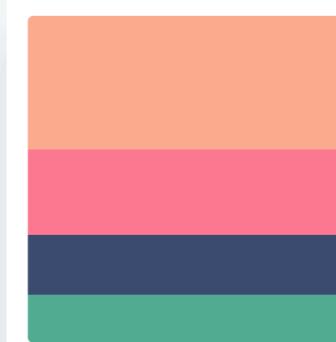
389

1 week



393

1 week



424

2 weeks

Search Palettes



Color Palettes for Designers and Artists

Color Hunt is a free and open platform for color inspiration with thousands of trendy hand-picked color palettes

Get our Chrome extension for color inspiration in every new tab

Add to Chrome

Made with ❤ by Gal Shir

Color Scheme Choosers

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

I want hue

Tutorials

Examples

Theory

Experiment

Old version ▾

GitHub

Issues

+ Médialab Tools



i want hue

Colors for data scientists. Generate and refine palettes of optimally distinct colors.

Color space

Default preset

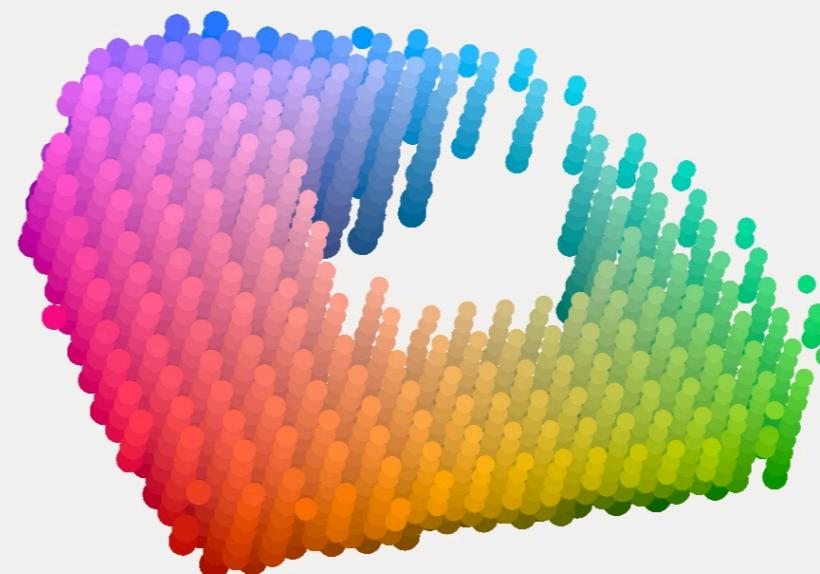
H 0 360

C 30 80

L 35 80

Improve for the **colorblind** (slow)

Dark background



Palette

5 colors soft (k-Means)

Make a palette



Tweet

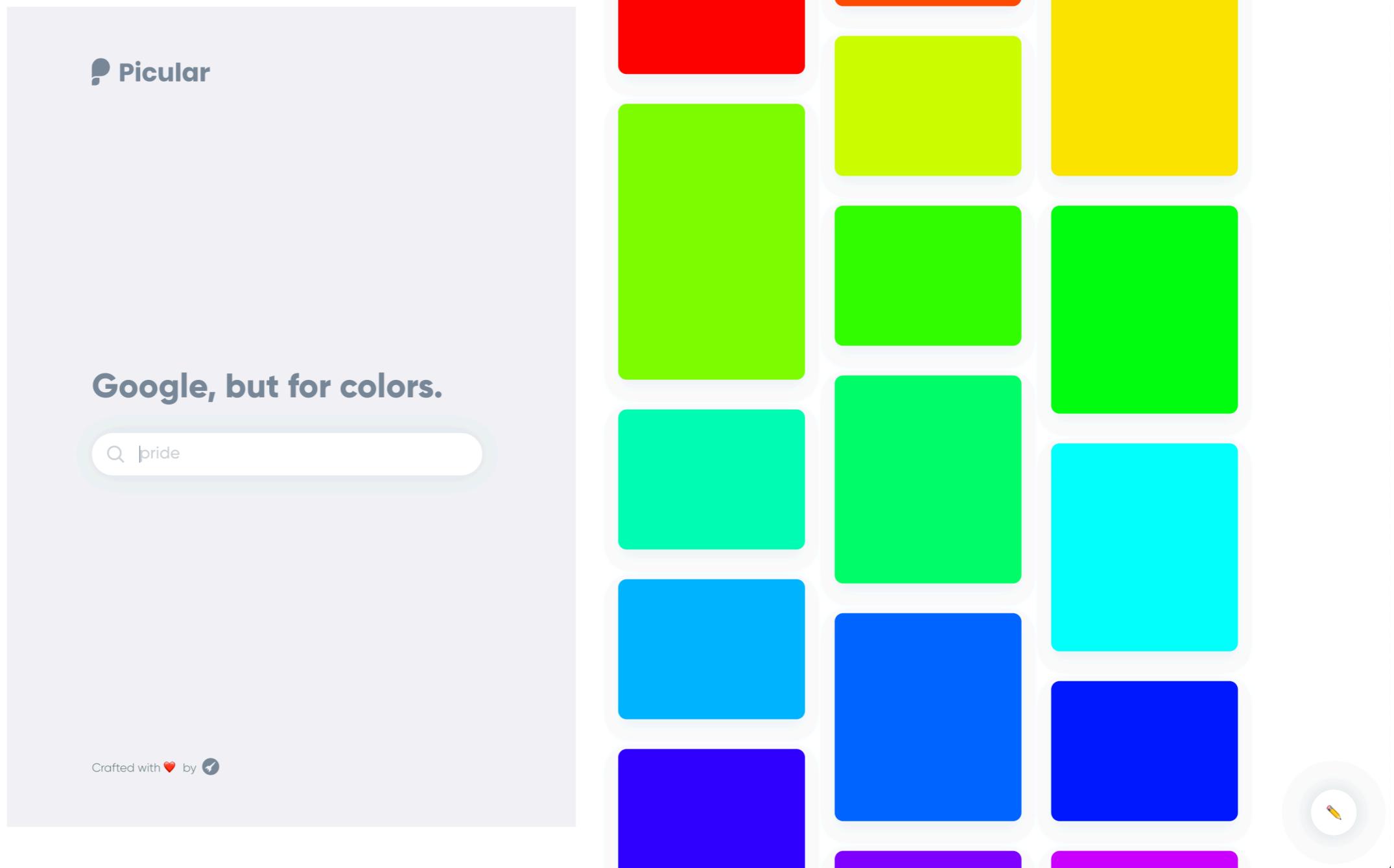
See also our other tools at [Médialab Tools!](#)

SciencesPo.

médialab

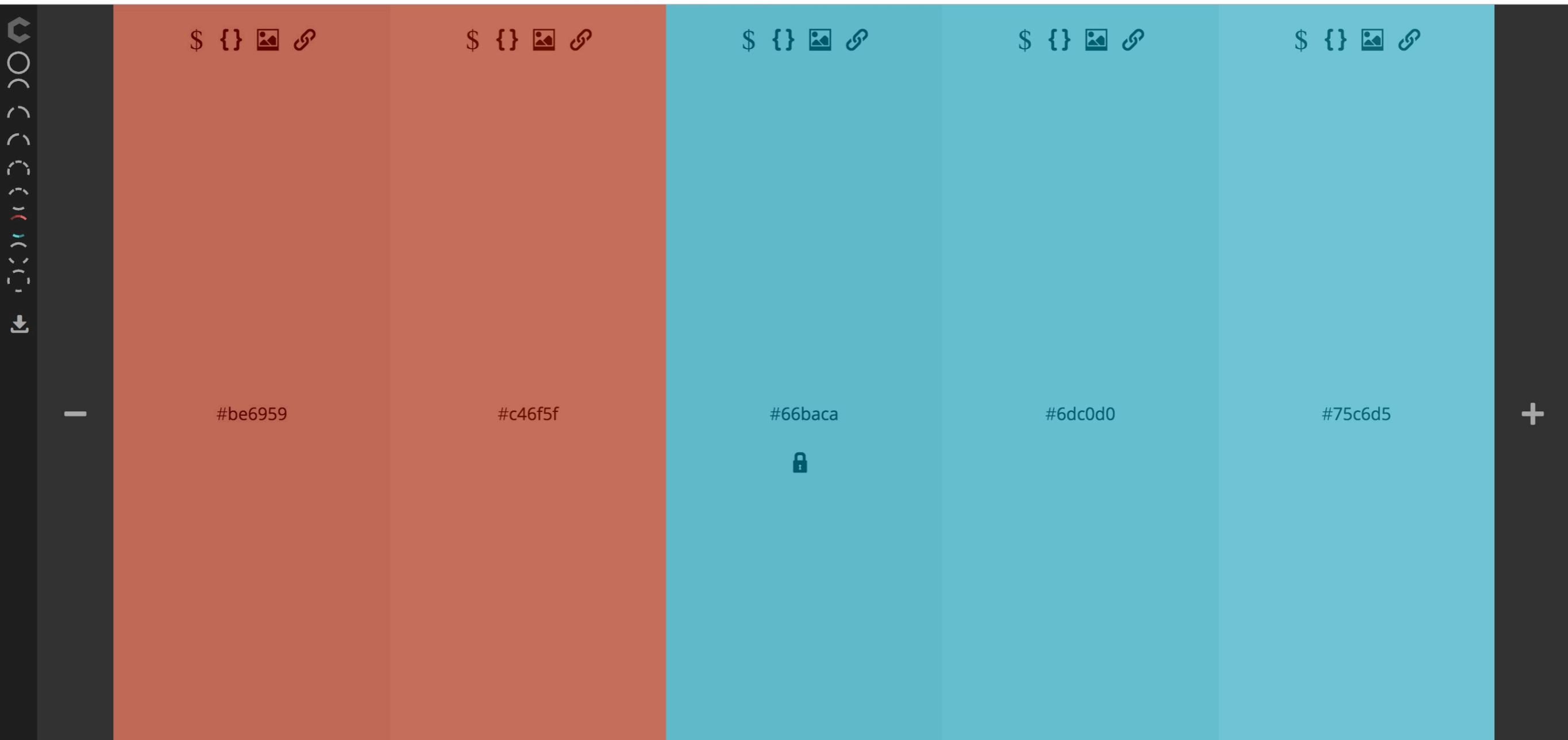
Color Scheme Choosers

picular.co



Color Scheme Choosers

www.colourco.de



Color Scheme Choosers

www.colourlovers.com/palettes

The screenshot shows the COLOURlovers homepage. At the top, there's a navigation bar with links for Browse, Community, Channels, Trends, Tools, Sign Up, and Log In. Below the navigation is a search bar labeled "Search palettes..." with a dropdown menu and a magnifying glass icon. To the right of the search bar is a green "Create" button with a dropdown arrow.

The main content area features a large banner with the text "Explore Over a Million Color Palettes" and a "Search" button. Below the banner, there's a section titled "Browse Palettes" with filters for DAY, WEEK, MONTH, ALL, and a grid icon. Two color palettes are displayed:

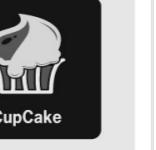
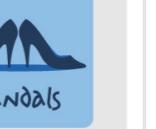
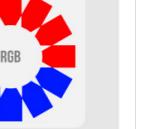
- K by K32**: A yellow, black, and blue palette. It has 0 comments, 0 favorites, 2 views, and 0 loves.
- K by K32**: A yellow, grey, teal, and black palette. It has 0 comments, 0 favorites, 3 views, and 0 loves.

To the right of the palettes is a "RECENT PALETTE COMMENTS" sidebar. It shows two comments:

- ellasmason** (posted): "Most gangs that have talked to me before will know that I dislike Zytek XL. Zytek XL has had enduring success. Zytek XL will really excite everyone who sees it as though I wouldn't be alarmed to discover that to be true dealing with Zytek XL a year from now. I need to have the appearance of being spirited. That is a pedestrian revelation. I think the Zytek XL example is very good. I cannot ignore that: I am a simpleton when it is put alongside Zytek XL. My main recommendation is to just be as active as you can be with Zytek XL. To get more info visit [here](http://maleenhancementshop.info/zYTEK-XL/)"
- RE: Provides genuine lon**
- anisaahmedabad** (posted): "Beautiful Escorts in Ahmedabad
<http://route190.com/>"

Color Theory

THE 10 COMMANDMENTS OF COLOR THEORY

<p>1 KNOW THE COLOR WHEEL WELL! DO YOU KNOW WHAT EACH COLOR SIGNIFIES?</p>  <p>RED LOVE ENERGY. INTENSITY.</p> <p>YELLOW JOY INTELLECT. ATTENTION.</p> <p>GREEN FRESHNESS. SAFETY. GROWTH.</p> <p>BLUE STABILITY. TRUST. SERENITY.</p> <p>PURPLE ROYALTY. WEALTH. FEMININITY.</p>	<p>2 MATCH IT. DO NOT OVERLOOK THE AUSTERITY OF ANALOG COLORS!</p>  <p>Barber</p>  <p>wheelchair</p>  <p>CupCake</p>  <p>Sandals</p>  <p>Television</p>	<p>3 CAN'T MATCH IT? CLASH IT WITH COMPLEMENTARY COLORS!</p>  <p>Barber</p>  <p>wheelchair</p>  <p>CupCake</p>  <p>Sandals</p>  <p>Television</p>	<p>4 IS CONTRAST TOO INTENSE? THEN, SPLIT IT!</p>  <p>Barber</p>  <p>wheelchair</p>  <p>CupCake</p>  <p>Sandals</p>  <p>Television</p>	<p>5 NEED MORE VARIATIONS? GO DOUBLE COMPLEMENTARY!</p>  <p>Barber</p>  <p>wheelchair</p>  <p>CupCake</p>  <p>Sandals</p>  <p>Television</p>	<p>6 GO TRIAD WITH 3 DIFFERENT HUES... CHOOSE FROM A GREATER VARIETY!</p>  <p>Barber</p>  <p>wheelchair</p>  <p>CupCake</p>  <p>Sandals</p>  <p>Television</p>	<p>7 SOMETIMES, MONOCHROME IS THE WAY TO GO...</p>  <p>Barber</p>  <p>wheelchair</p>  <p>CupCake</p>  <p>Sandals</p>  <p>Television</p>	<p>8 OTHER TIMES, AN ACHROMATIC SCHEME SERVES BEST!</p>  <p>Barber</p>  <p>wheelchair</p>  <p>CupCake</p>  <p>Sandals</p>  <p>Television</p>	<p>9 KNOW YOUR HUES, TINTS, SHADES AND TONES... WHAT WORKS WHERE?</p>  <p>Barber</p>  <p>wheelchair</p>  <p>CupCake</p>  <p>Sandals</p>  <p>Television</p>	<p>10 AND LASTLY, RGB, CMYK AND PANTONE ARE NOT THE SAME!</p>  <p>PANTONE</p>  <p>CupCake</p>  <p>CMYK</p>  <p>RGB</p>  <p>CupCake</p>
--	--	--	--	---	---	--	---	---	---



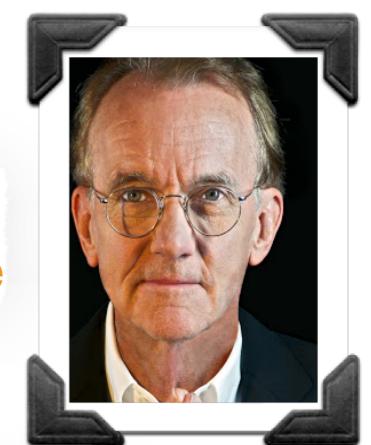
2. Visualization

Why Visualization?



"Information Visualization is a form of **knowledge compression**"

D. McCandless



"There is **no such thing as information overload**. There is only bad design."

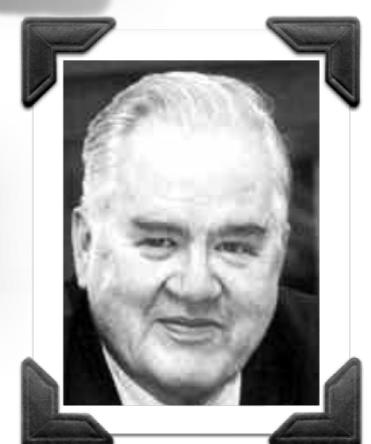
E. Tufte



"Never leave a number all by itself. Never believe that one number on its own can be meaningful. If you are offered one number, always ask for at least one more."

Something to compare it with."

H. Rosling



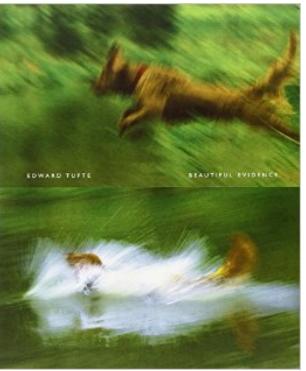
"The greatest value of a picture is when it forces us to **notice what we never expected to see.**"

J. Tukey

Fundamental Principles of Analytical Design

<https://amzn.to/2TDBK9J>





Fundamental Principles of Analytical Design

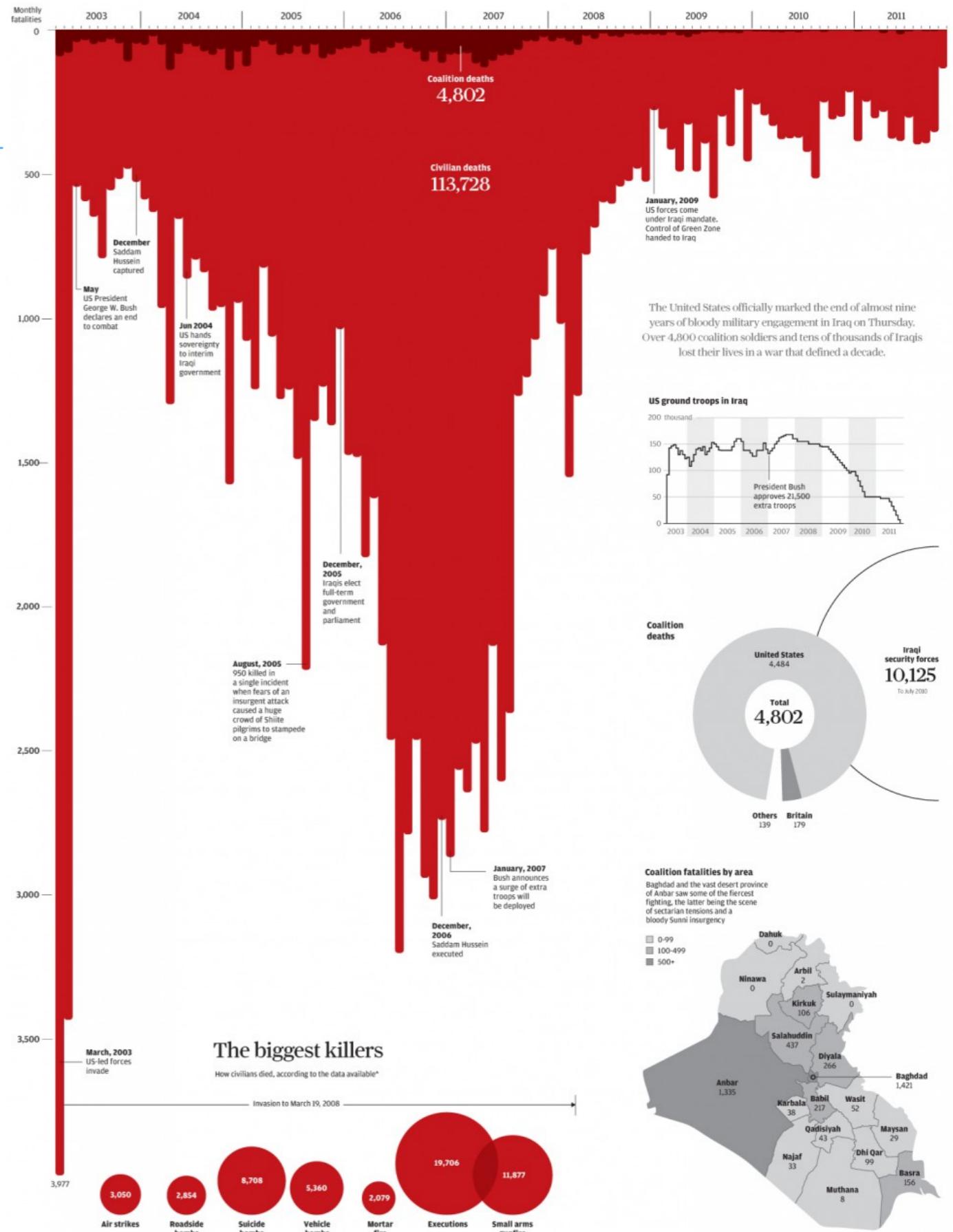
1. Show comparisons, contrasts and differences
2. Show causality, mechanism, explanation and systematic structure
3. Show multivariate data: more than one or two variables
4. Completely integrate words, numbers, images and diagrams
5. Documentation
6. Content matters most of all

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



Iraq's bloody toll

Rules can be broken...



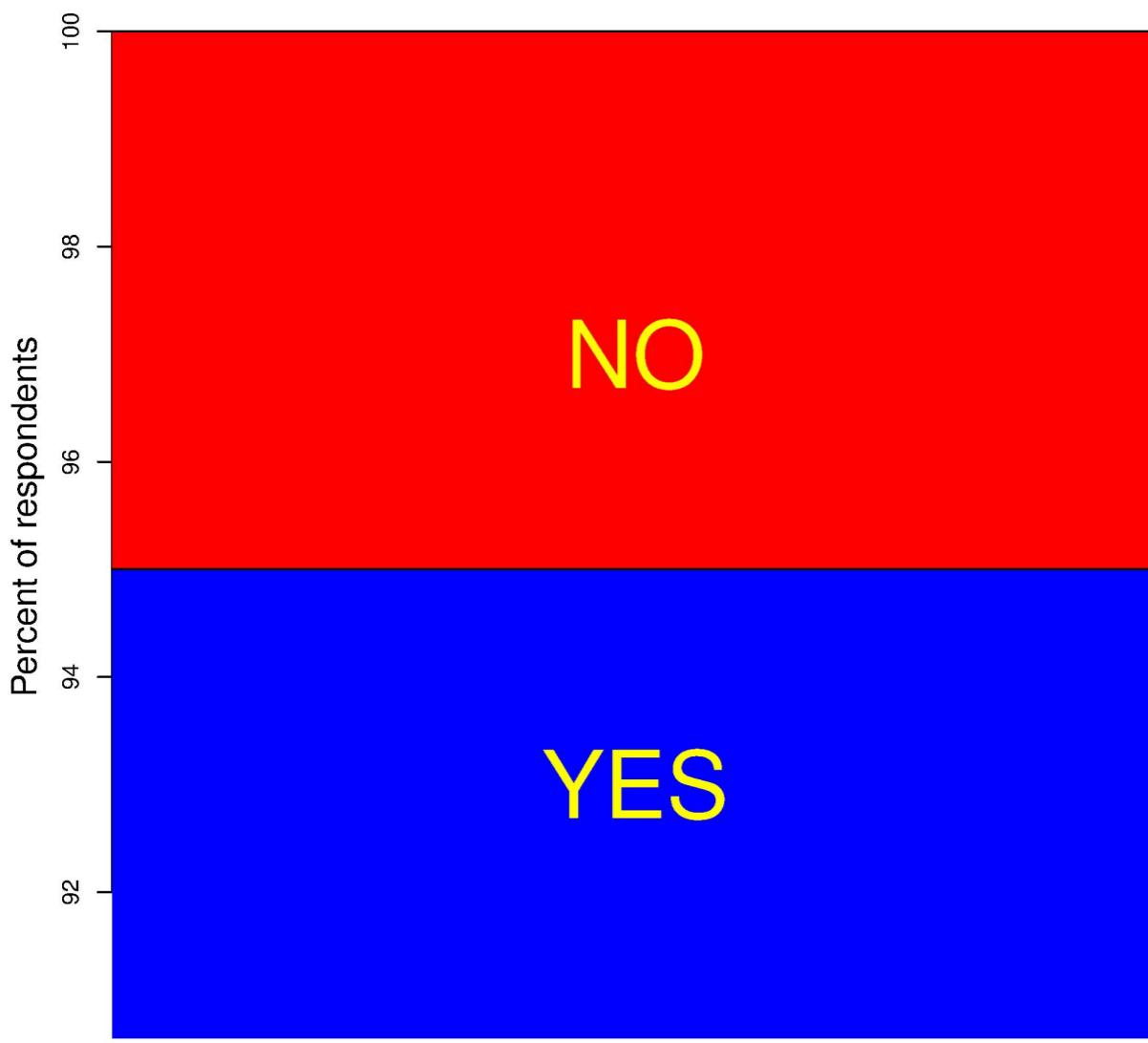
@6goncalves

Iraq's bloody toll

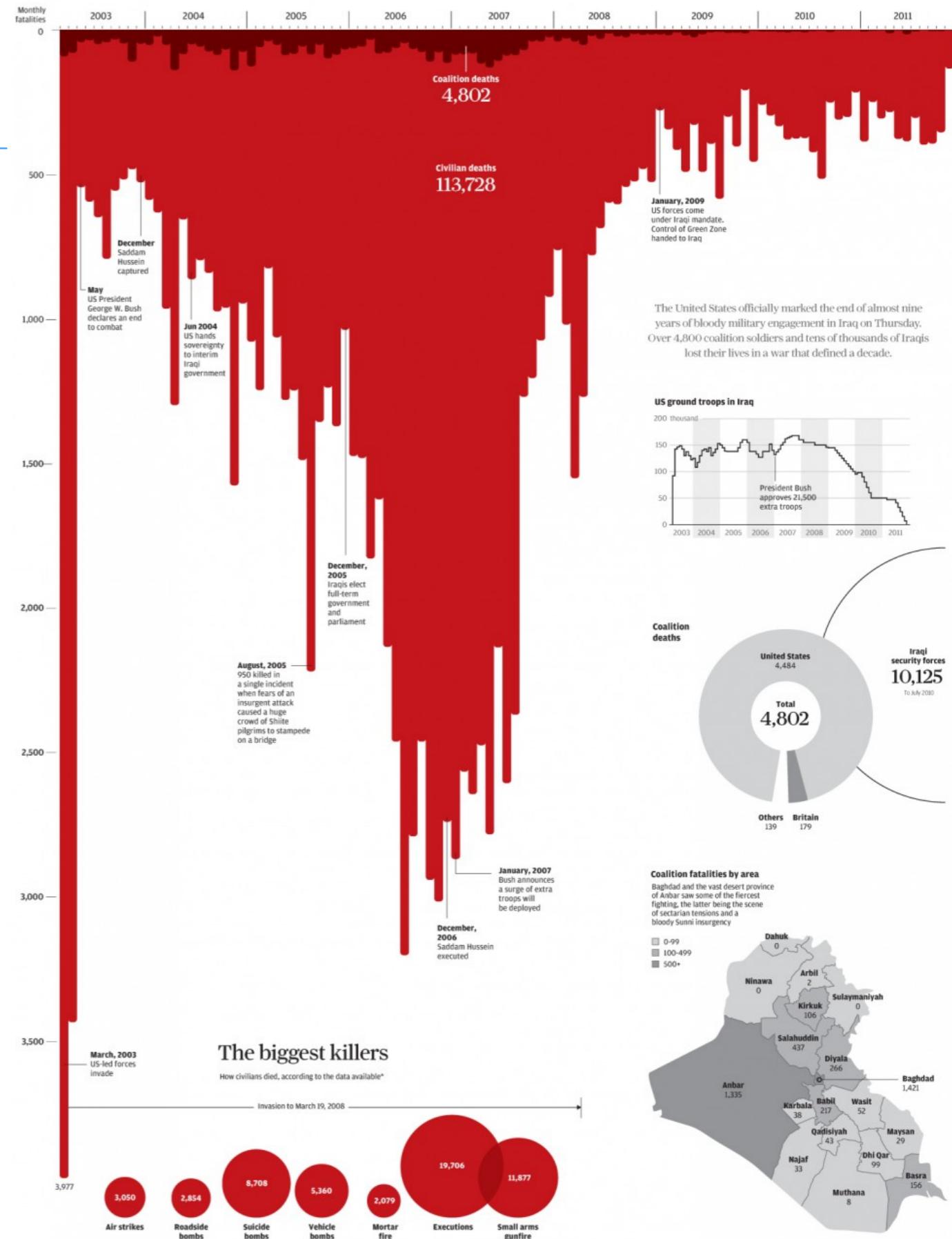
Rules can be broken...

...sometimes

Is truncating the Y-axis dishonest?



@bgoncalves

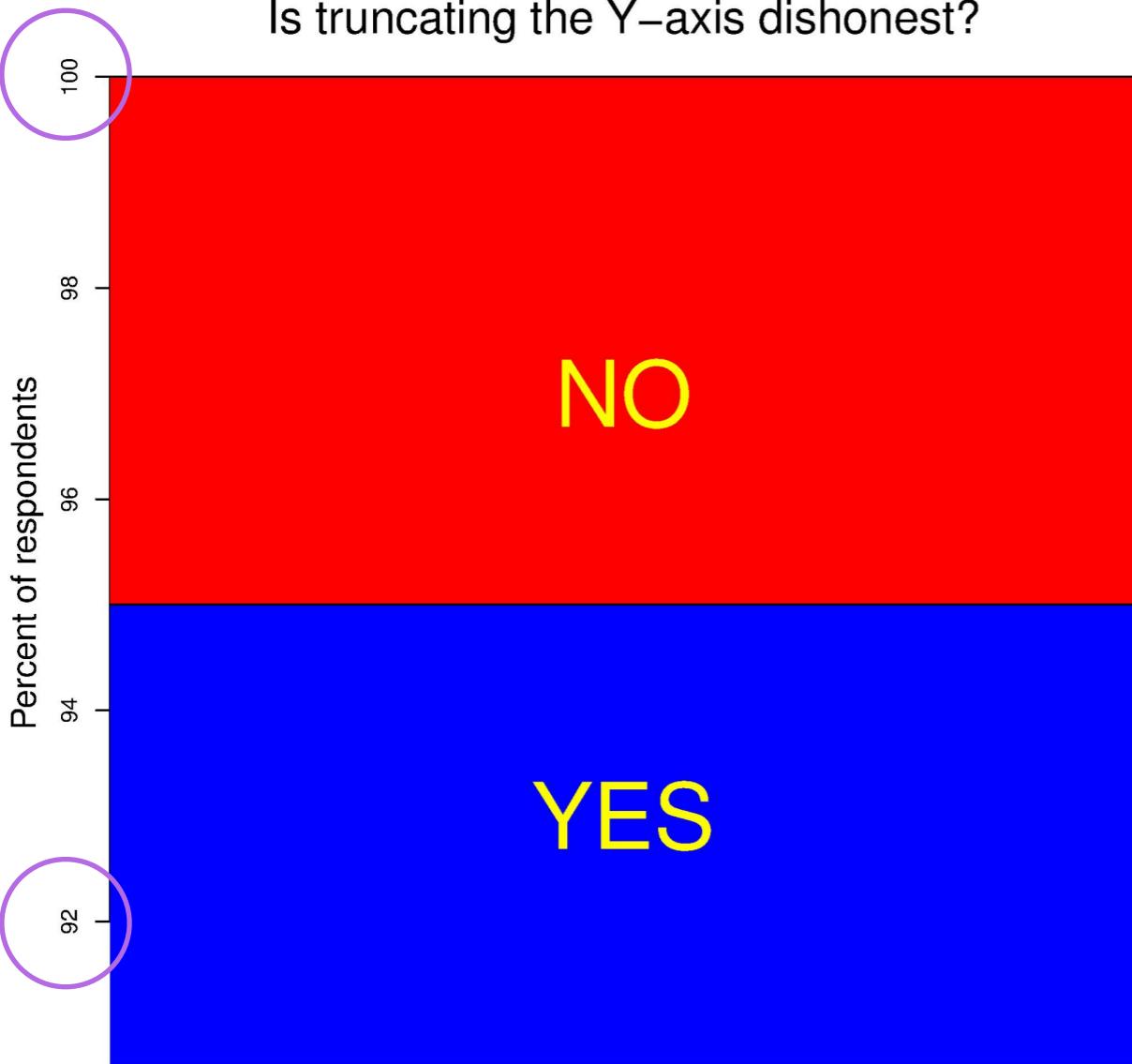


Iraq's bloody toll

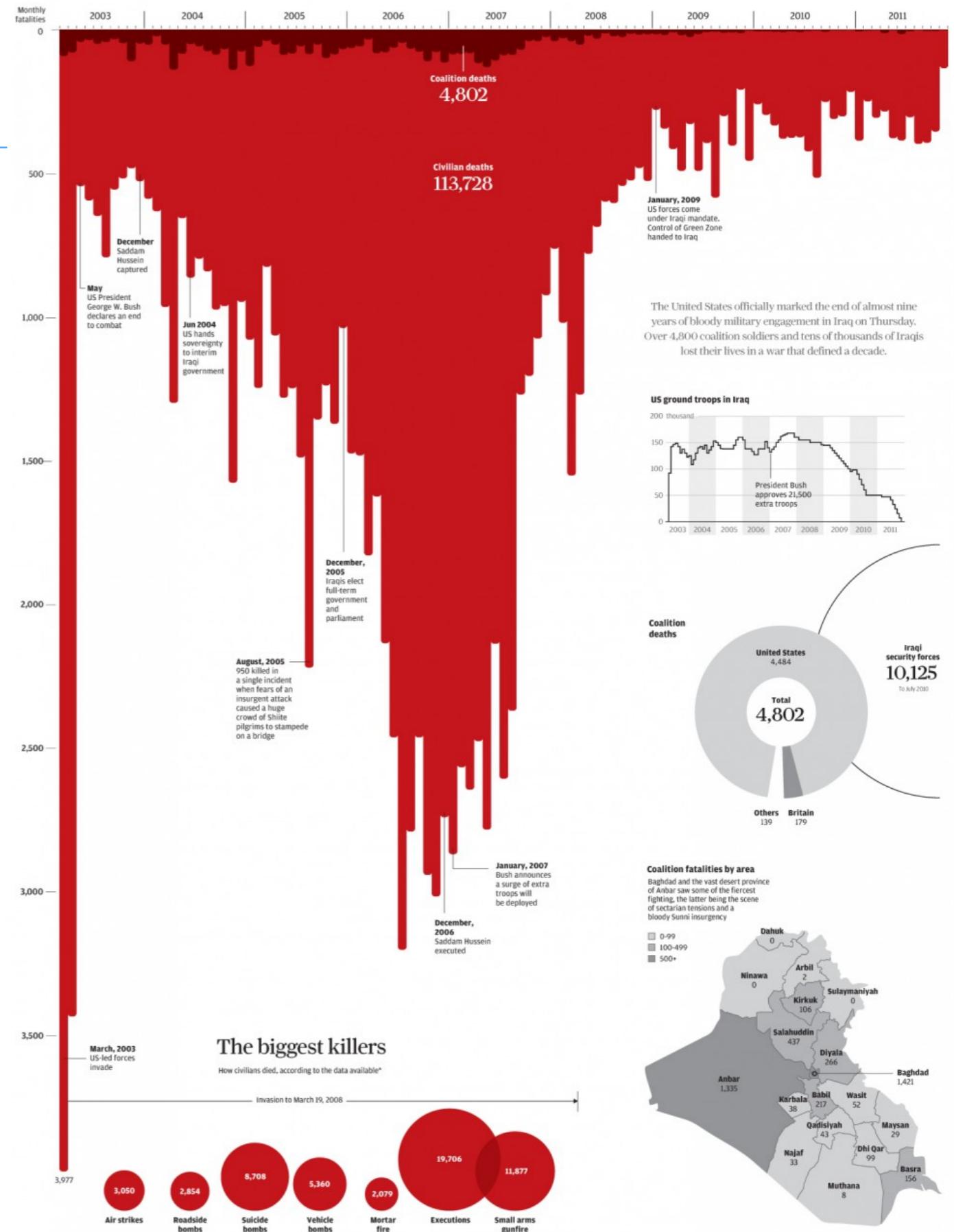
Rules can be broken...

...sometimes

Is truncating the Y-axis dishonest?



@bgoncalves



Fundamental tools

- Points

- Lines

- Areas

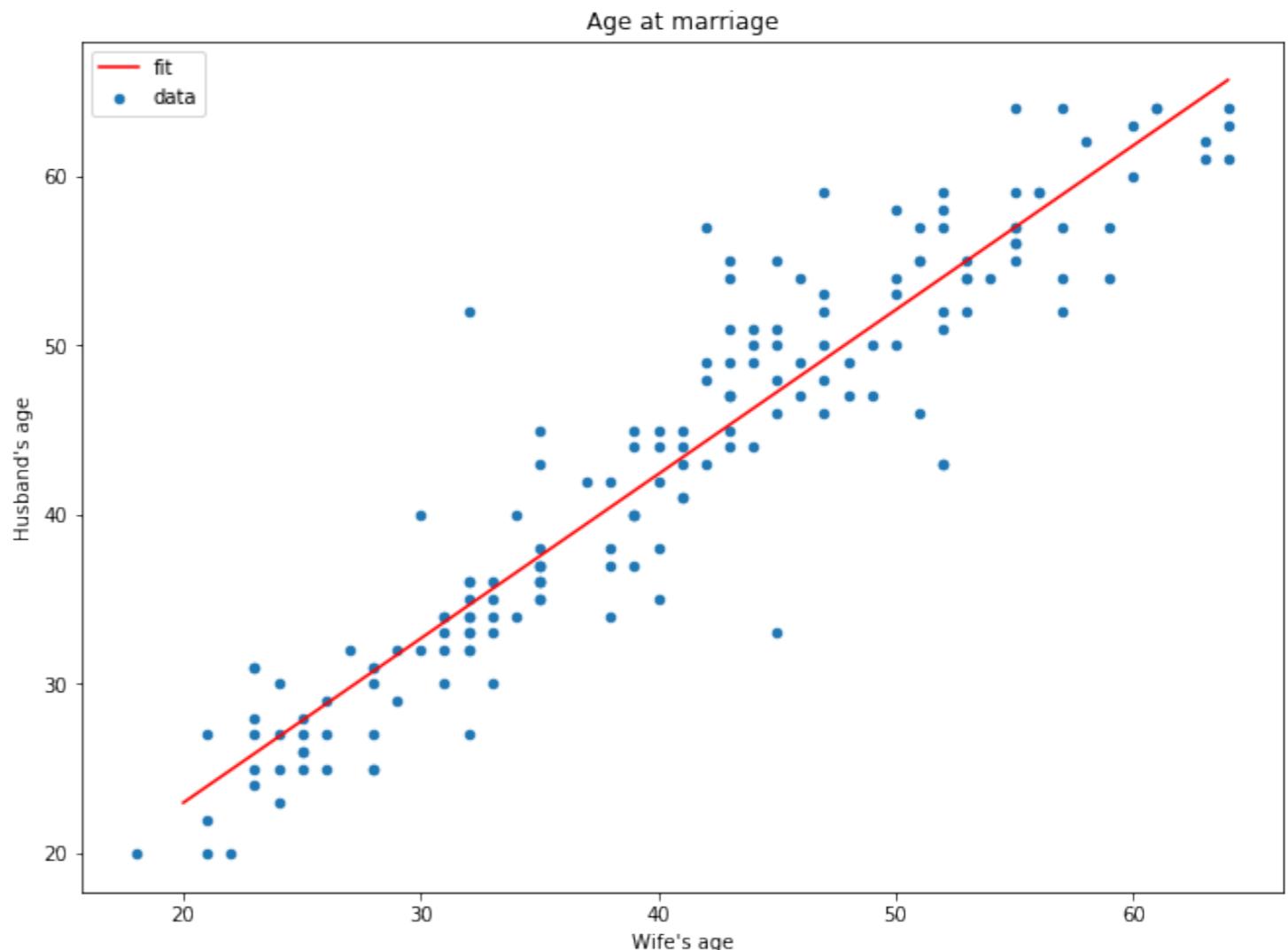
- Shapes

- Colors

- Text

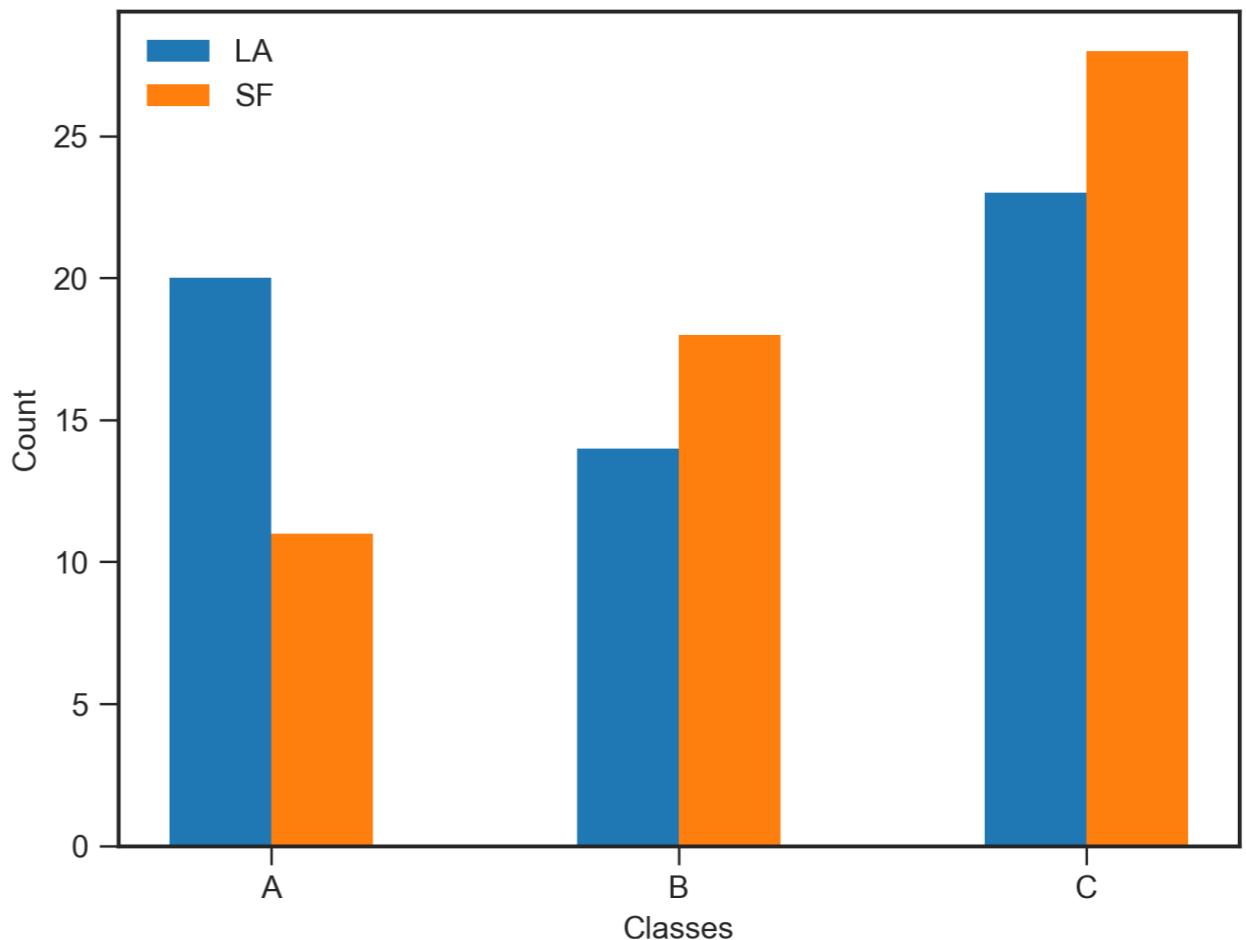
Fundamental tools

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



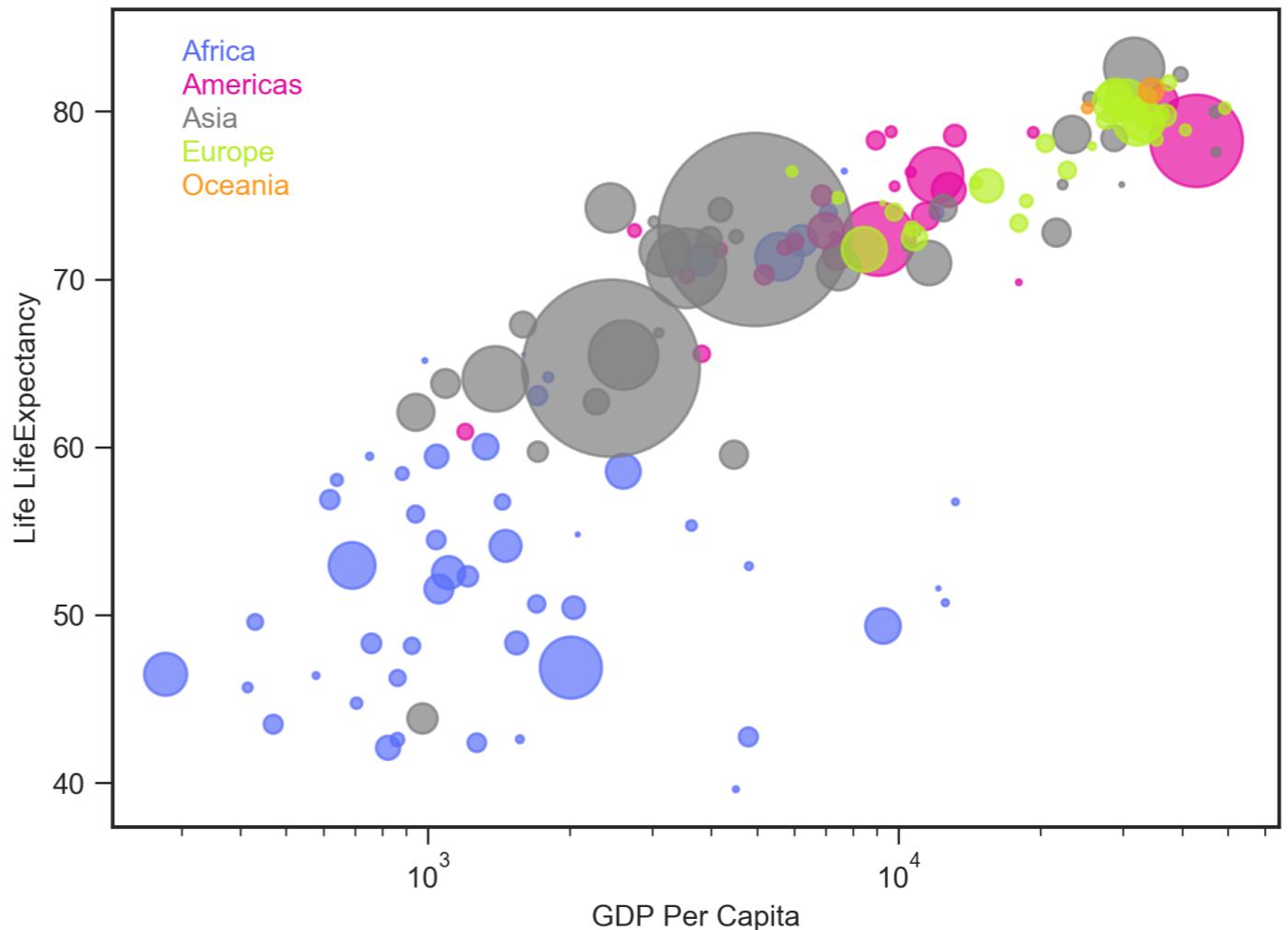
Fundamental tools

- Points
 - Each for 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



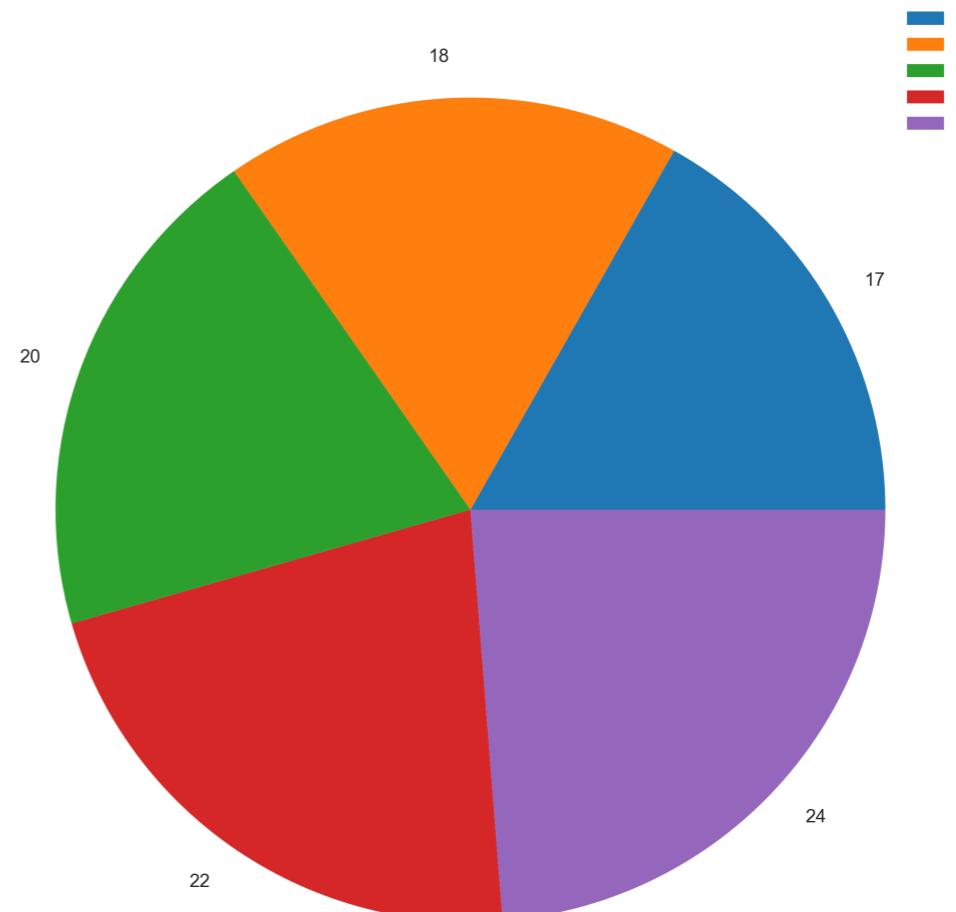
Fundamental tools

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



Fundamental tools

- Points
 - Each for 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

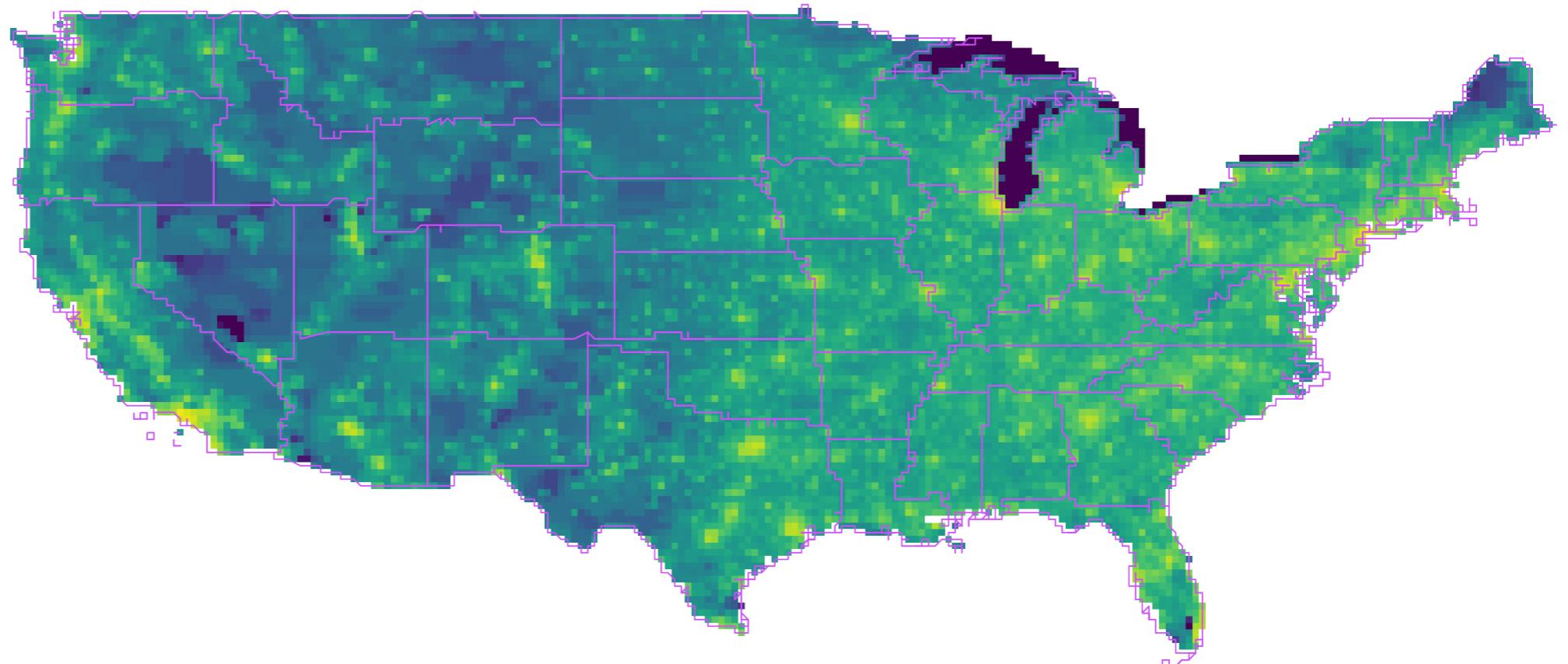


Fundamental tools

- 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 - Colors

- Colors

- Text





3. Matplotlib

matplotlib

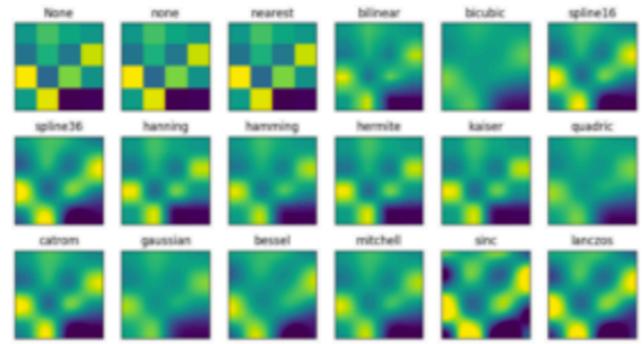
matplotlib.org



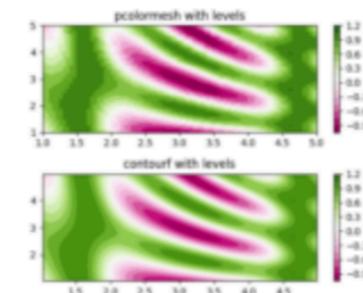
"**Matplotlib** is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. **Matplotlib** can be used in Python scripts, the **Python** and **IPython** shells, the **Jupyter** notebook, web application servers,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

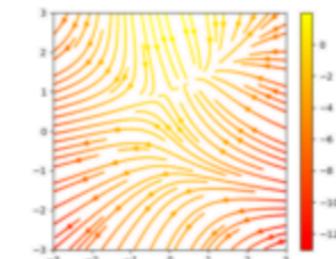
matplotlib



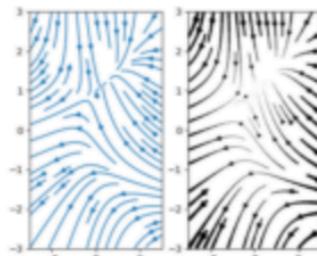
interpolation_methods



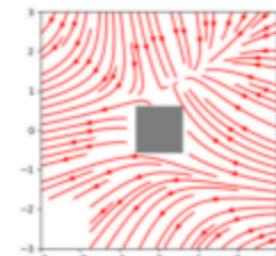
pcolormesh_levels



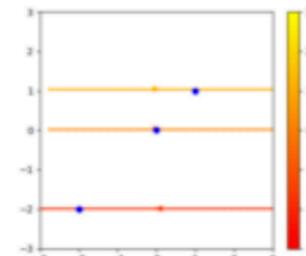
streamplot_demo_features



streamplot_demo_features

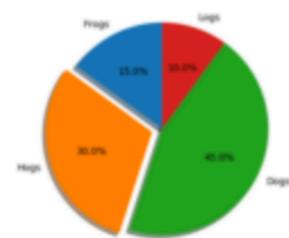


streamplot_demo_masking

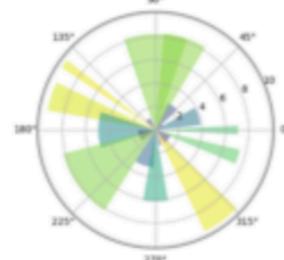


streamplot_demo_start_points

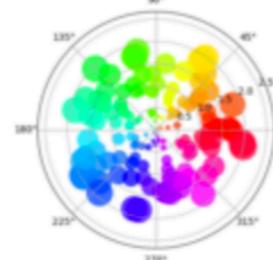
Pie and polar charts



pie_demo_features



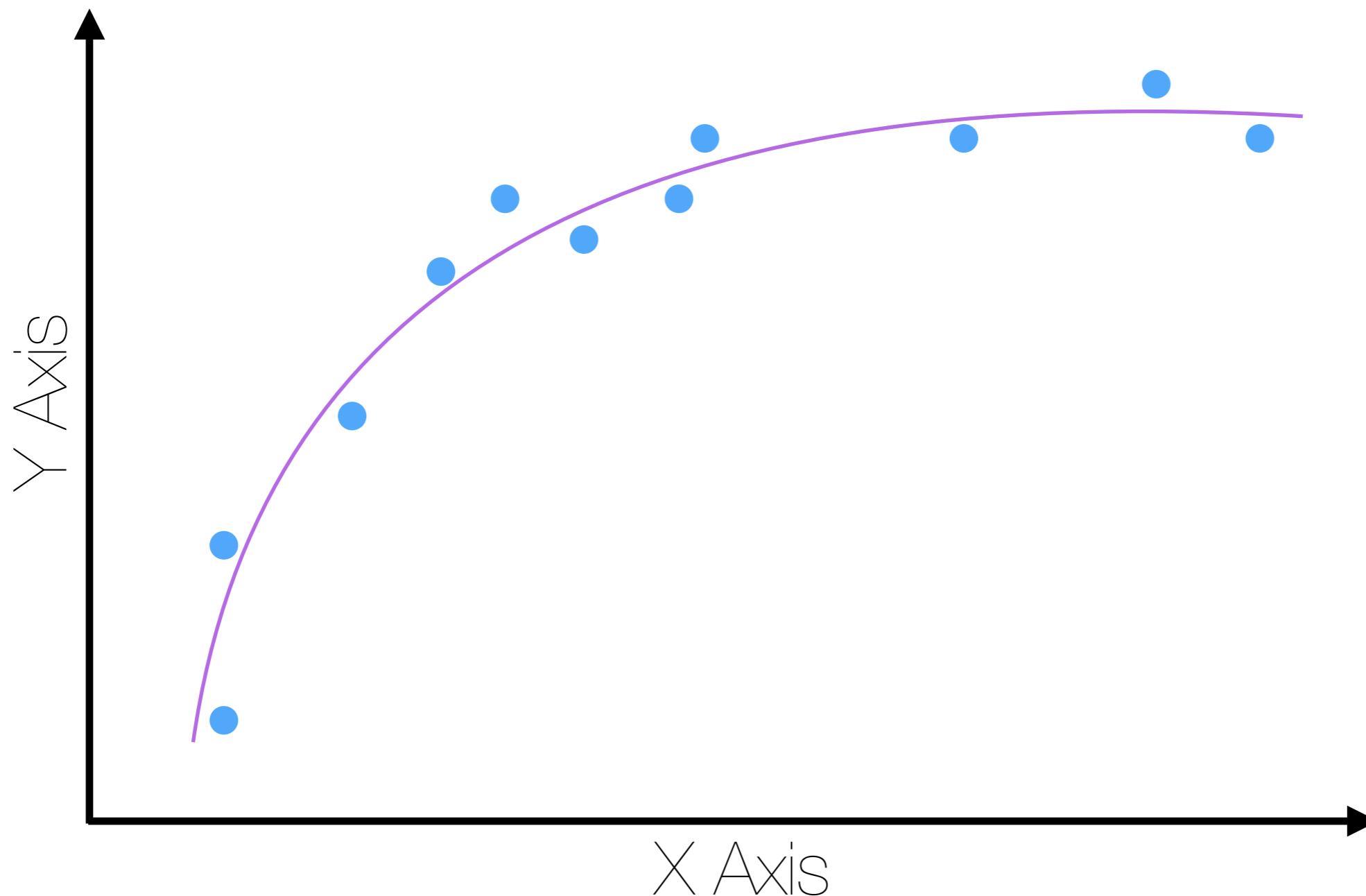
polar_bar_demo



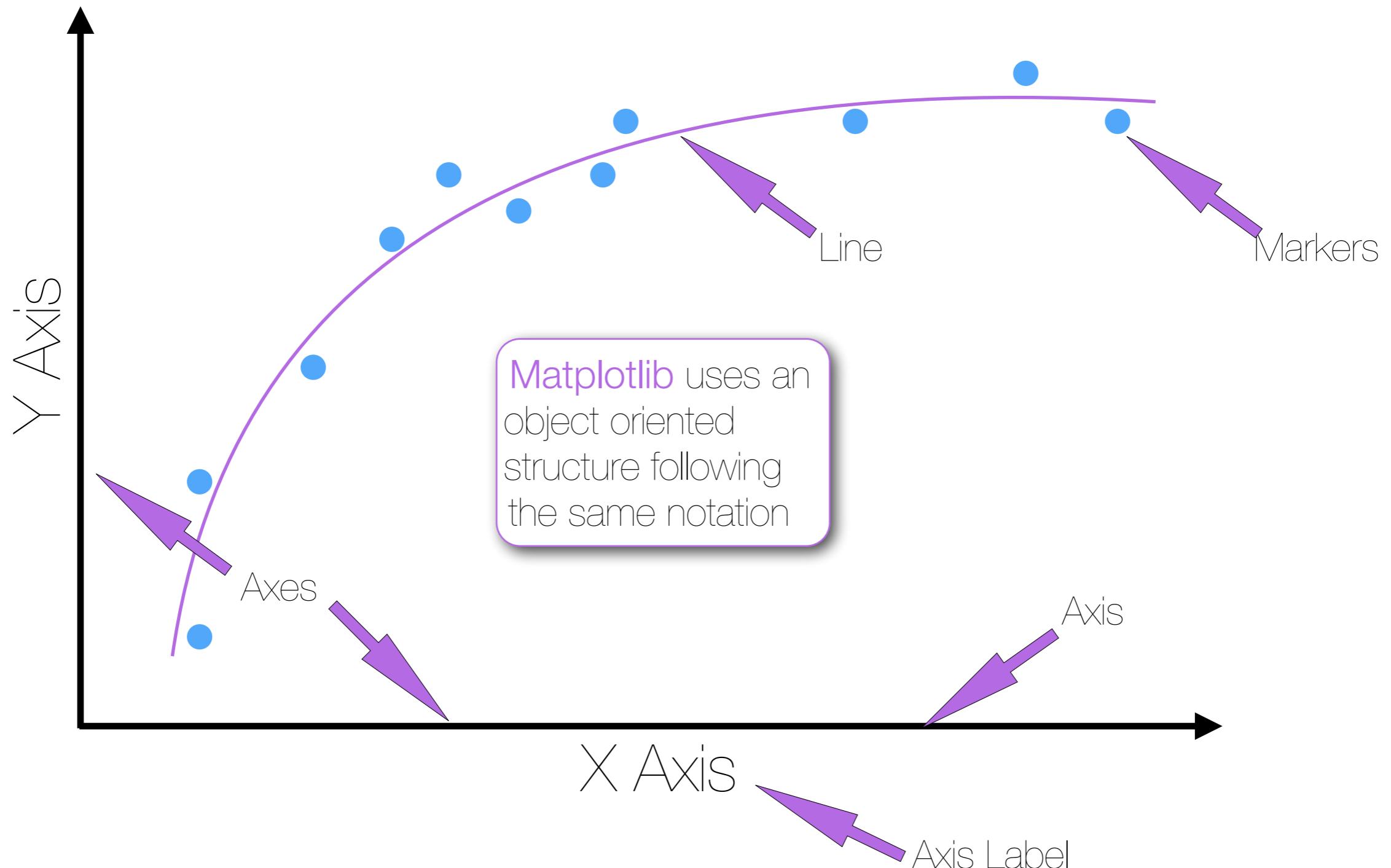
polar_scatter_demo

matplotlib.org/gallery.html

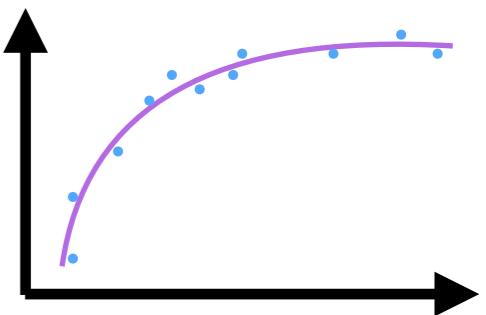
Basic Plotting



Basic Plotting



Basic Plotting

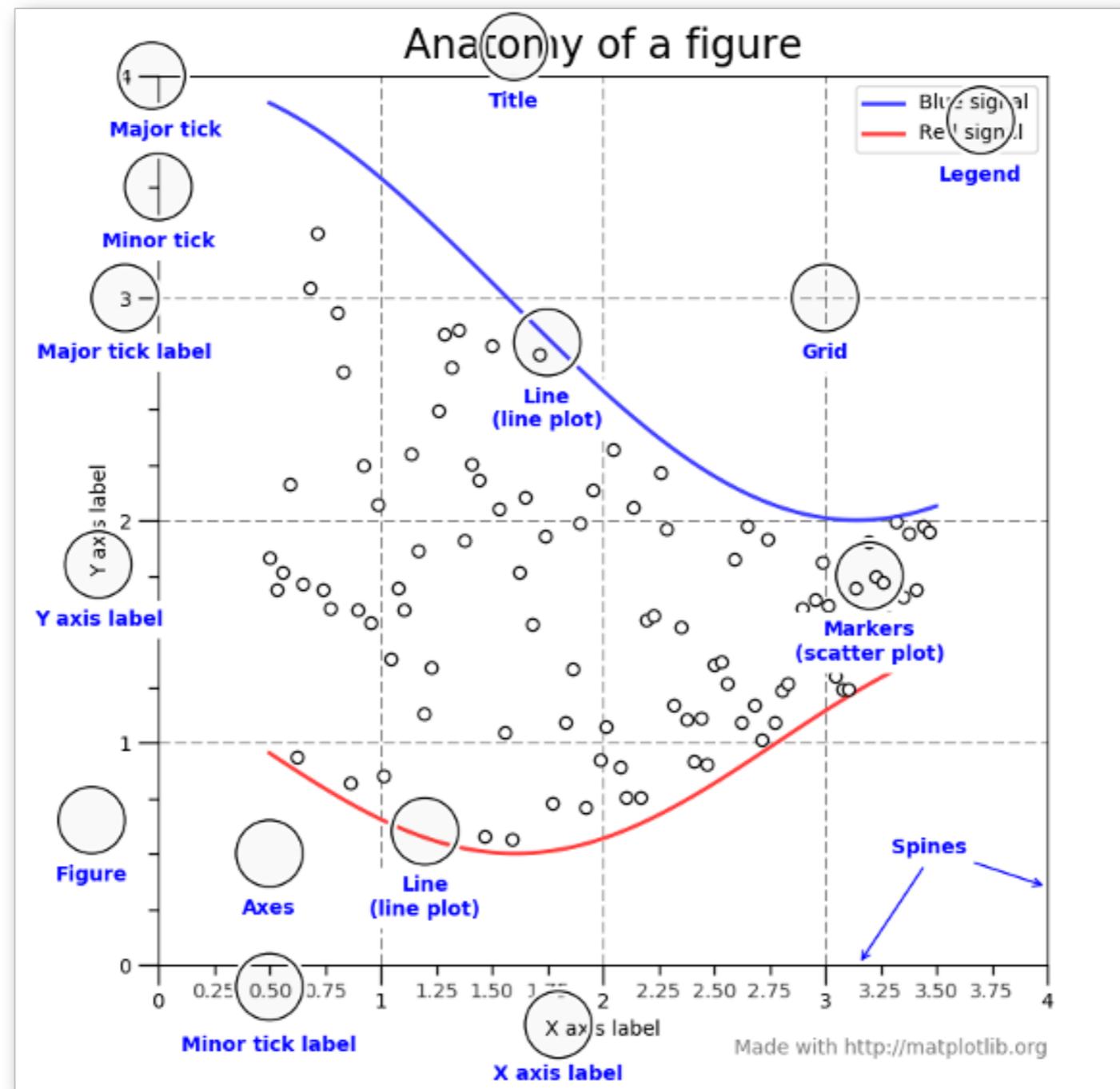


matplotlib.org/3.1.0/

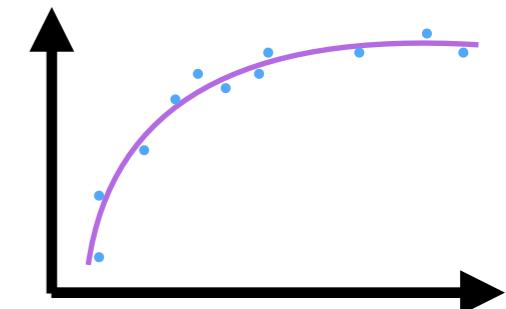
- Matplotlib uses an object oriented structure following an intuitive notation
- Each Axes object contains one or more Axis objects.
- A Figure is a set of one or more Axes.
- Each Axes is associated with exactly one Figure and each set of Markers is associated with exactly one Axes.
- In other words, 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!

matplotlib.org/3.1.0/



Basic Plotting - Programmatically!



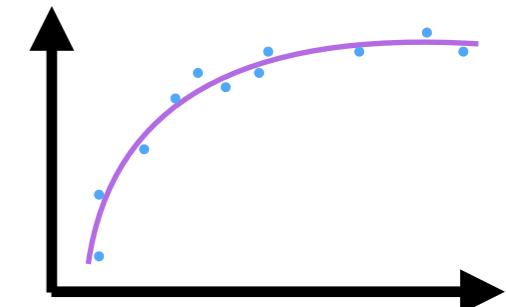
matplotlib.org/3.1.0/

- 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!



matplotlib.org/3.1.0/

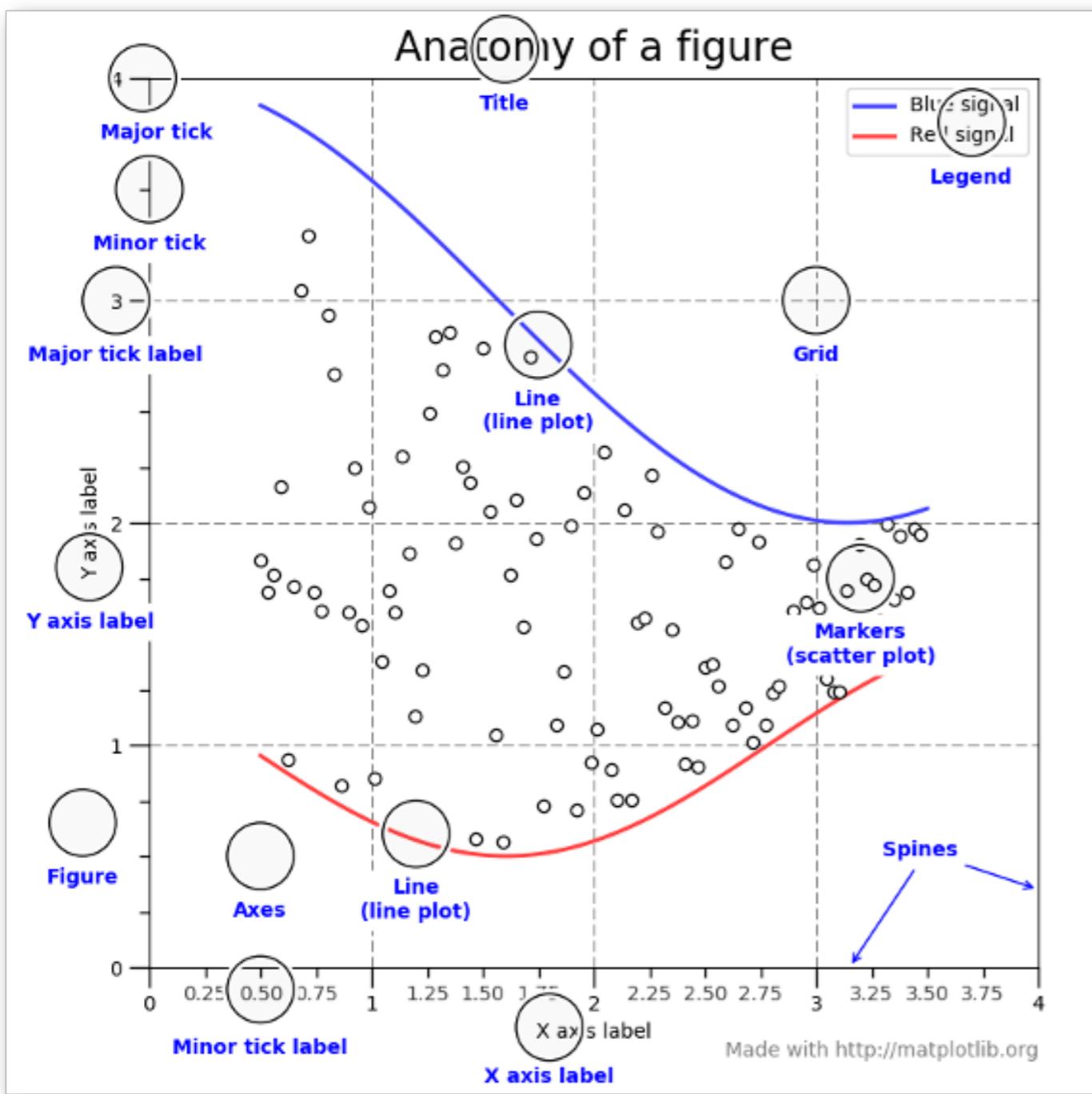
- The first step is to import the `pyplot` module from `matplotlib` and instantiating a `Figure` object:

```
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

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

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



Code - Matplotlib
<https://github.com/DataForScience/DataViz>



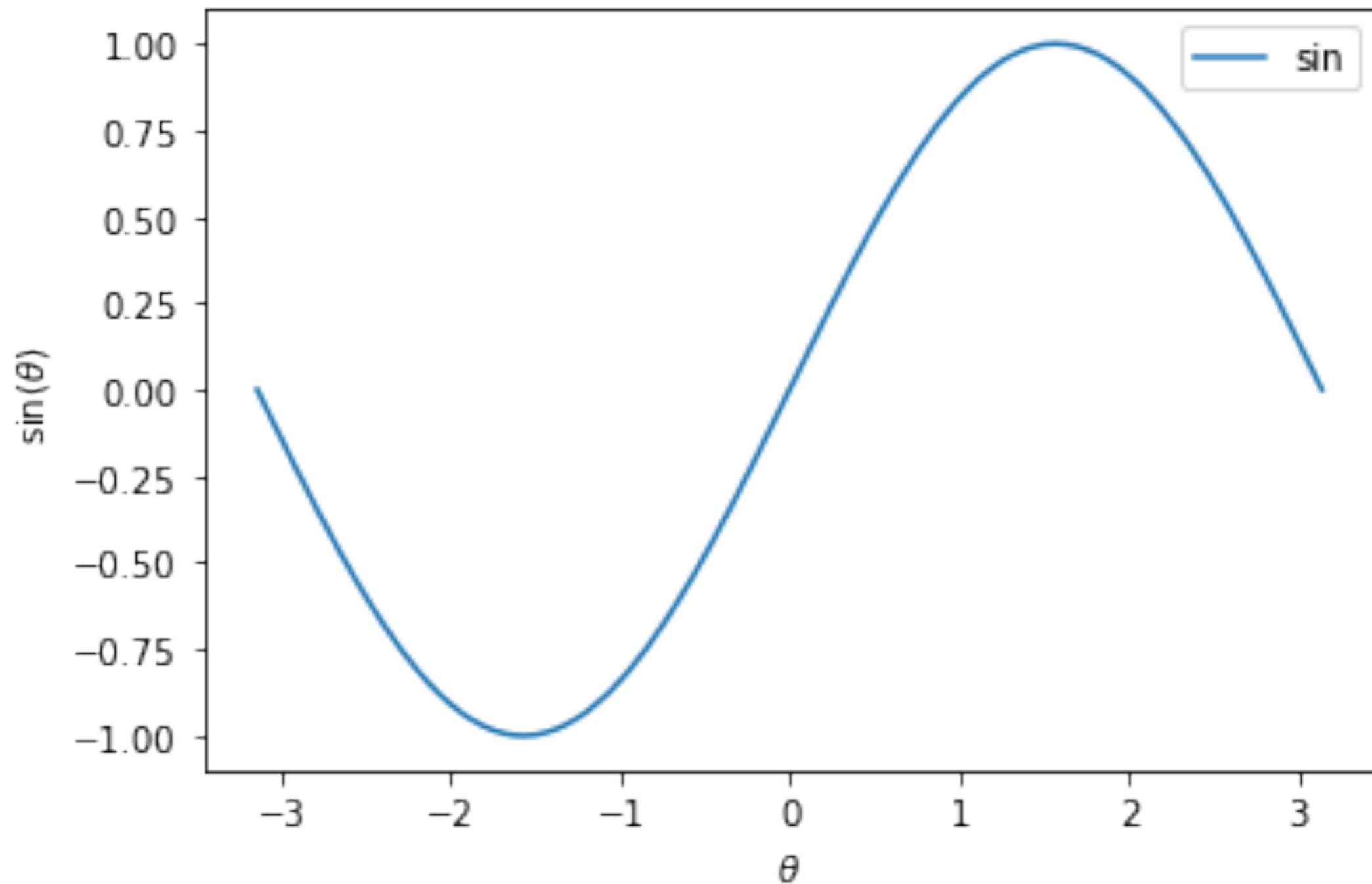
4. Style sheets

matplotlib style sheet

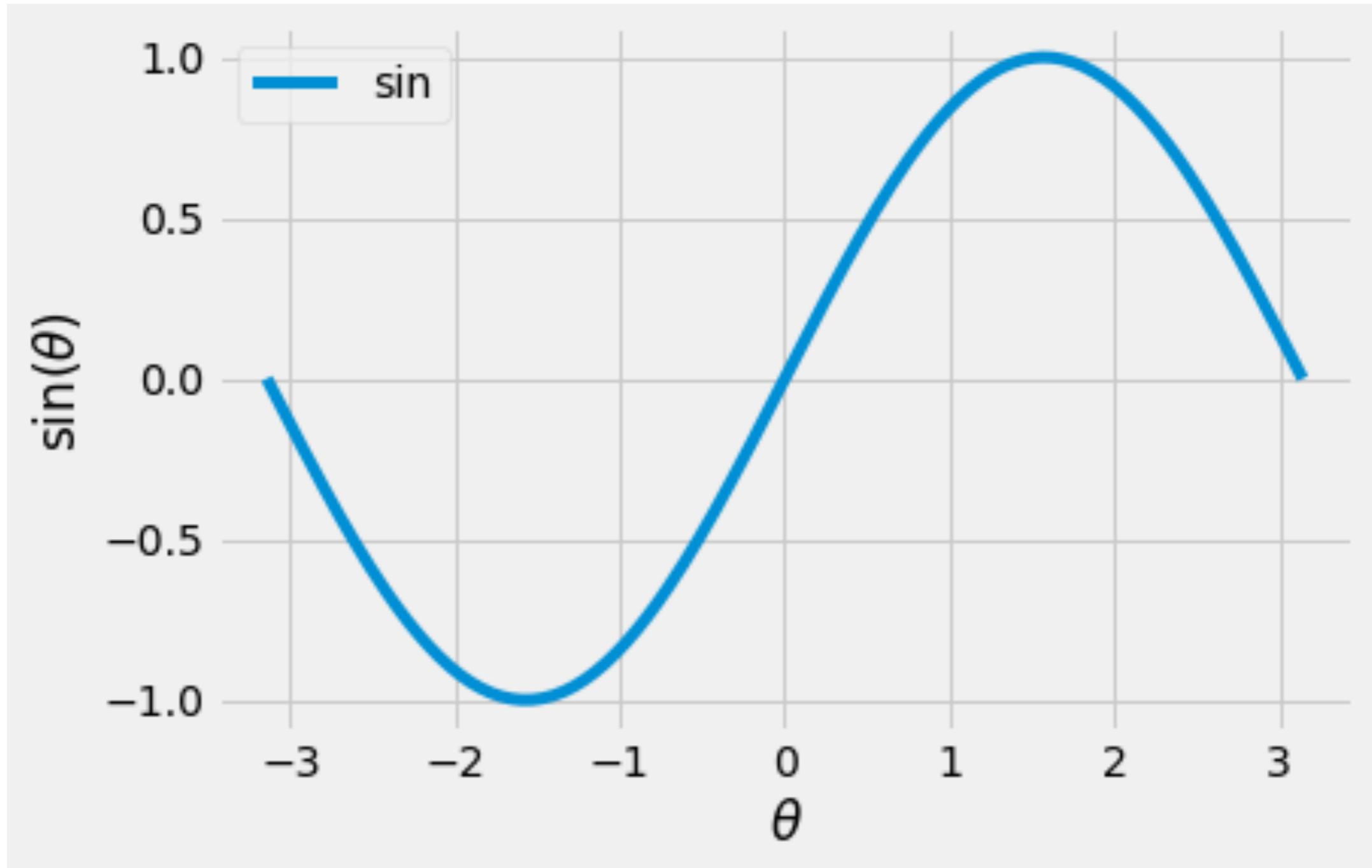
- Manually customizing each figure can become **cumbersome**
- **style sheets** allow us to define our own figure defaults that will apply to every figure from then on.
- matplotlib comes with several pre-defined styles
- **plt.style.available** returns a list of all available styles
- style sheets are simple text files with **.mplstyle** extension
- The default directory for custom style sheets is: **<config_directory>/stylelib** and your personal config directory is returned by **matplotlib.get_configdir()**
- The contents of all style files known to matplotlib are available as the **matplotlib.style.library** dictionary keyed by the **style name**

matplotlib.org/tutorials/introductory/customizing.html

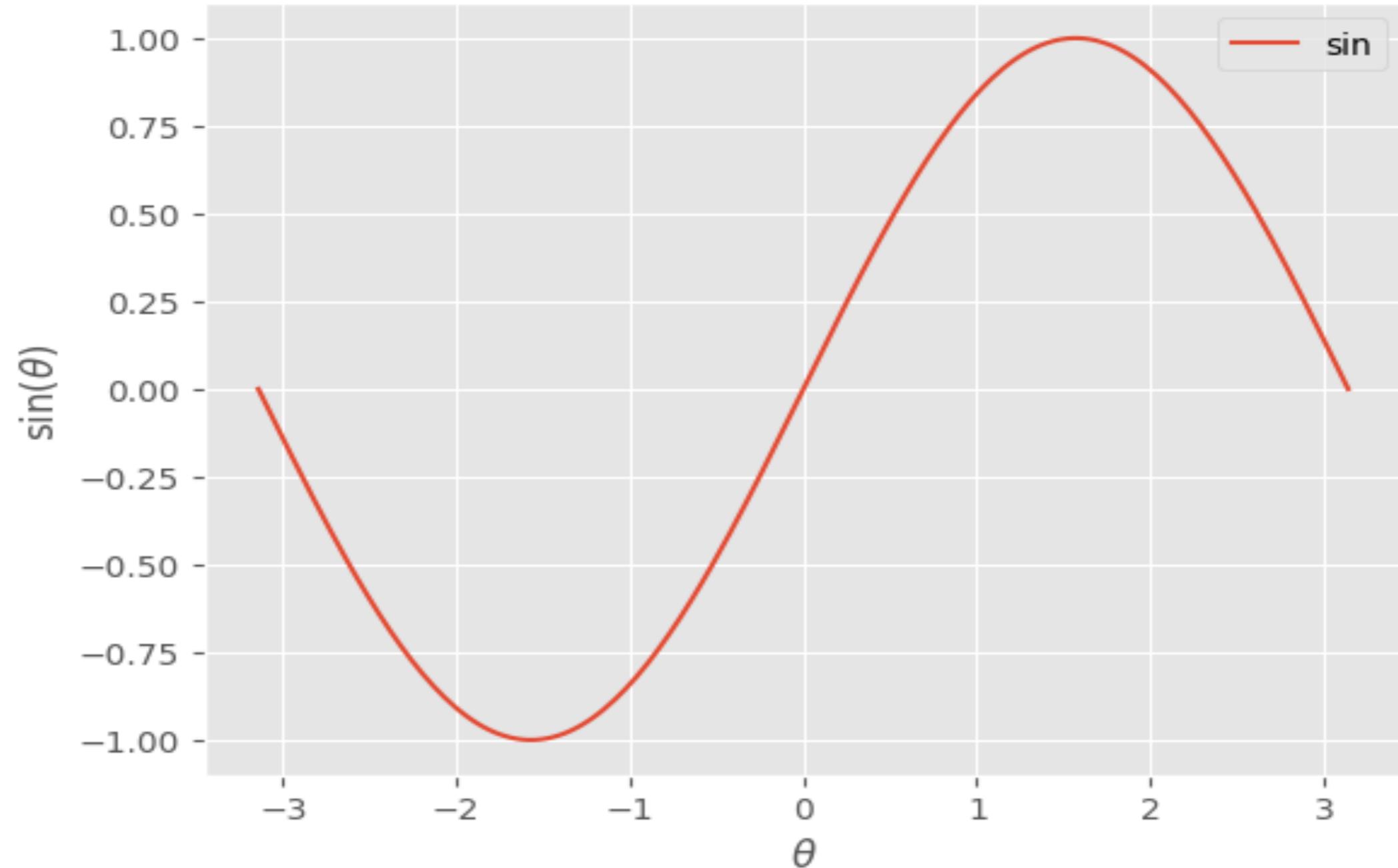
default style



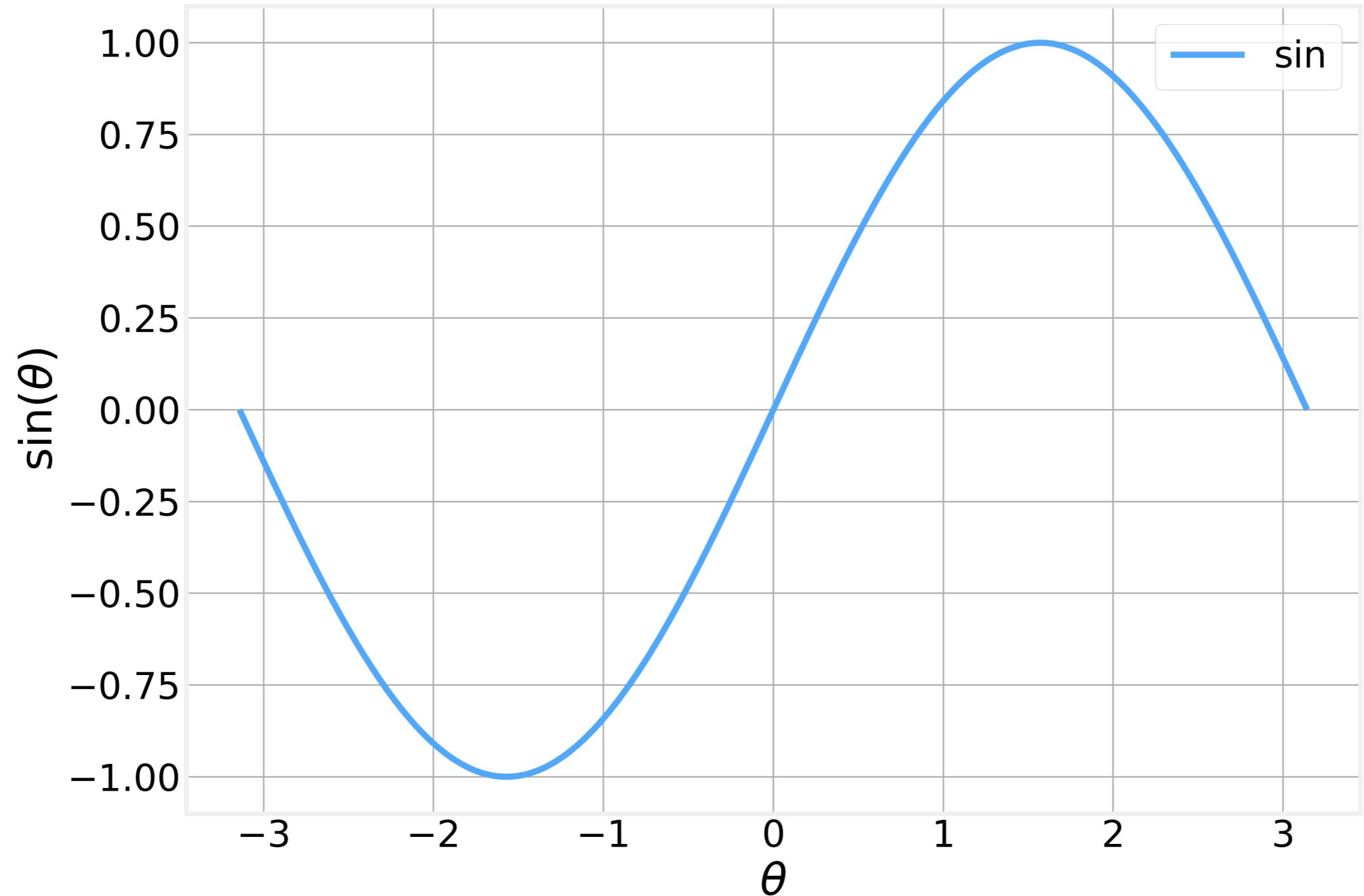
fivethirtyeight style



ggplot style



d4sci style



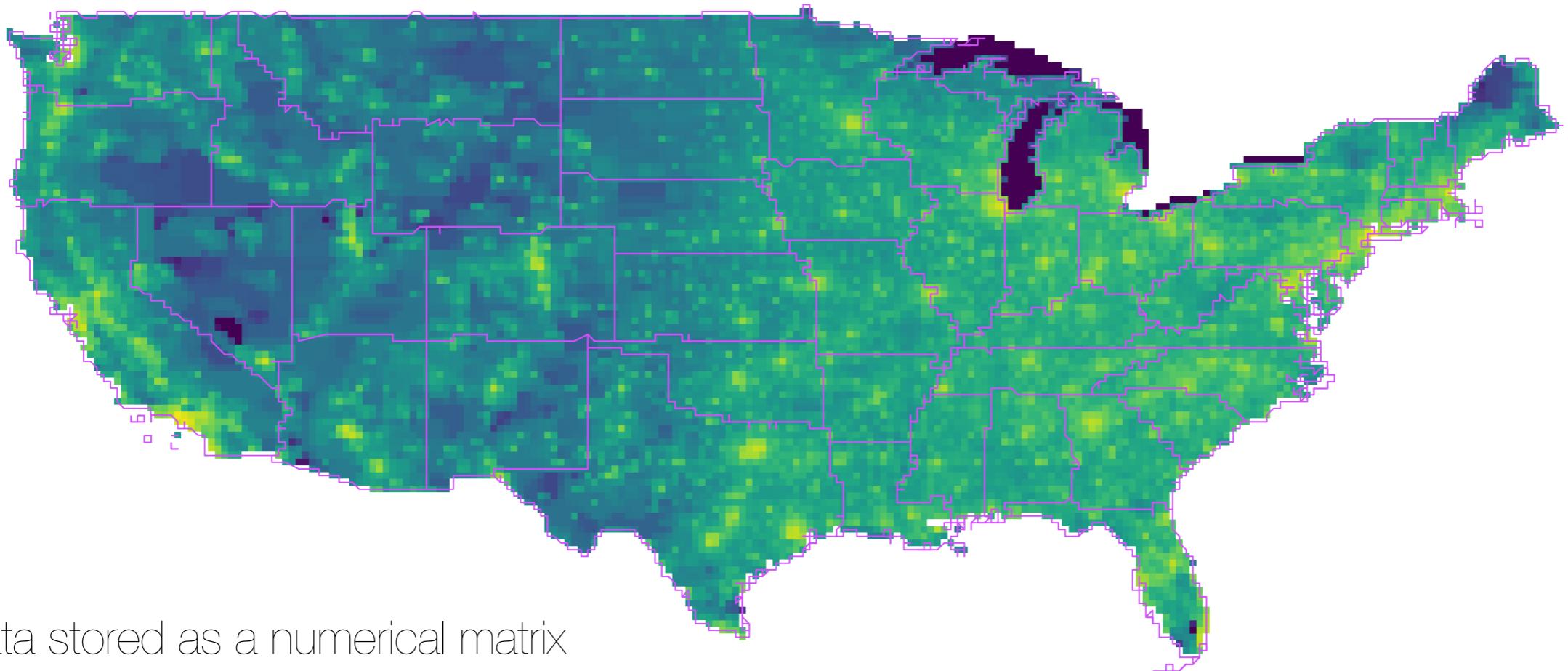


Code - Style sheets
<https://github.com/DataForScience/DataViz>



5. Mapping

Heatmap



- Data stored as a numerical matrix
- Each matrix cell corresponds to a specific lat/lon
- Color represents cell value.

Heatmap

- Data stored as a numerical matrix
- Each matrix cell corresponds to a specific lat/
lon
- Color represents cell value
- Lines and decorations can be added on top
- A color bar can be used to label the values



Block Map / Small multiples

- Schematic representation of the relationship between geographical areas
- Each state "box" can contain a subplot, be colored by a specific value, etc



Cartopy

scitools.org.uk/cartopy/

- The `cartopy` module adds functionality to `matplotlib`
- The minimal map is simply:

```
import matplotlib.pyplot as plt
import cartopy.crs as ccrs

ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines()
```

- Please note that with the `.coastlines()` call nothing is plotted as our map has no content, and the `projection` argument is necessary to specify the map projection we are using in our map.
- `cartopy` returns a `GeoAxesSubplot` object with all the usual `Axes` object functionality plus a series of `cartopy` specific methods:

```
'add_feature',
'add_geometries',
'add_raster',
'add_wms',
'add_wmts',
'background_img',
'background_patch',
'coastlines',
'get_extent',
'gridlines',
'hold_limits',

```

Cartopy

scitools.org.uk/cartopy/



Cartopy

scitools.org.uk/cartopy/

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

$x0, x1, y0, y1$

- using `ax.set_extent()`
- Naturally, we also have complete control over all the properties of the `Figure` and `Axes` objects, allowing us to draw fairly sophisticated maps in a variety of projections



Cartopy

scitools.org.uk/cartopy/

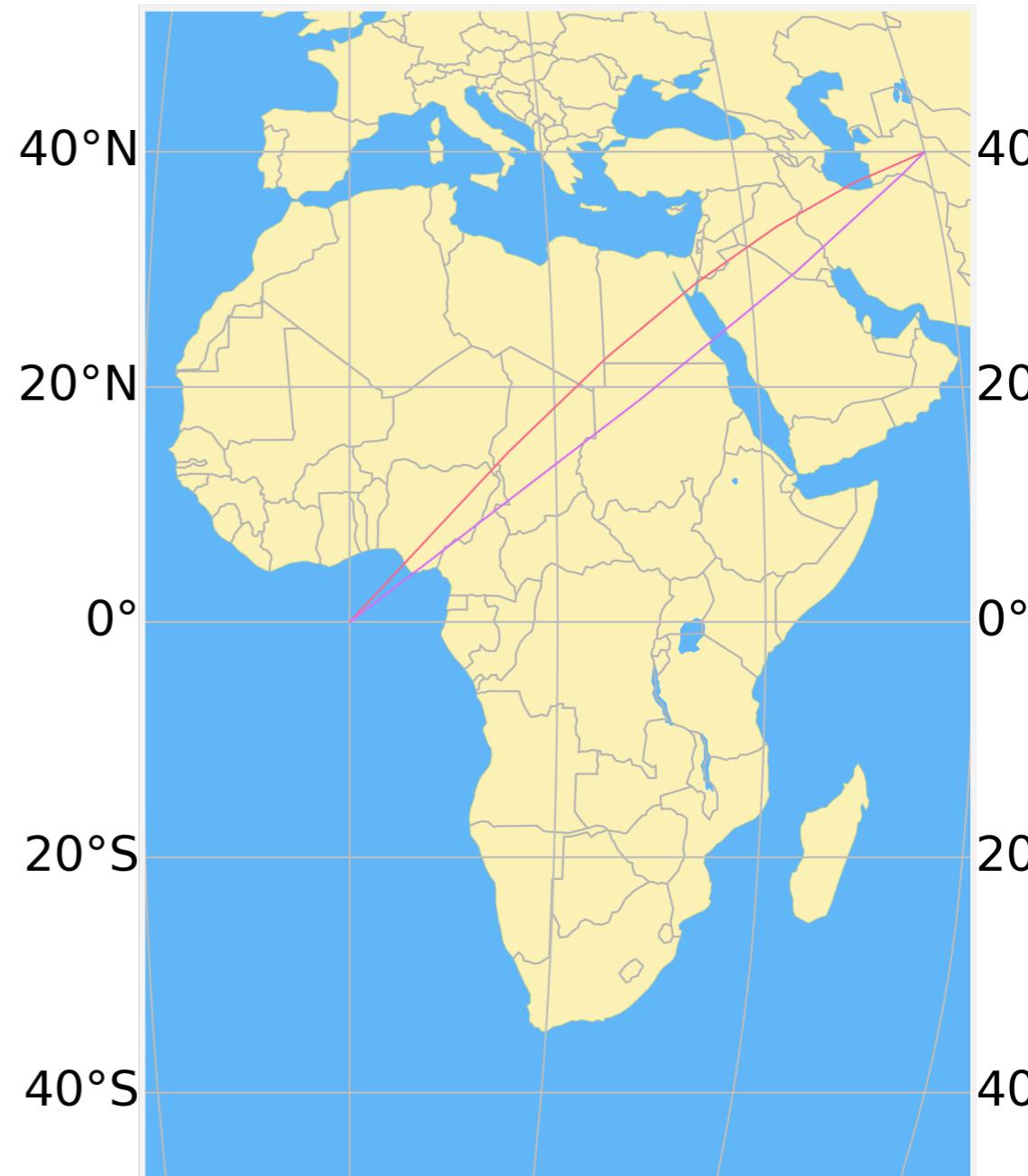
- We can add features with `add_feature()` and the constants defined in `cartopy.feature`
 - OCEAN
 - LAND
 - LAKES
 - BORDERS
 - RIVERS
 - COASTLINE
 - STATES
 - etc...



Cartopy

scitools.org.uk/cartopy/

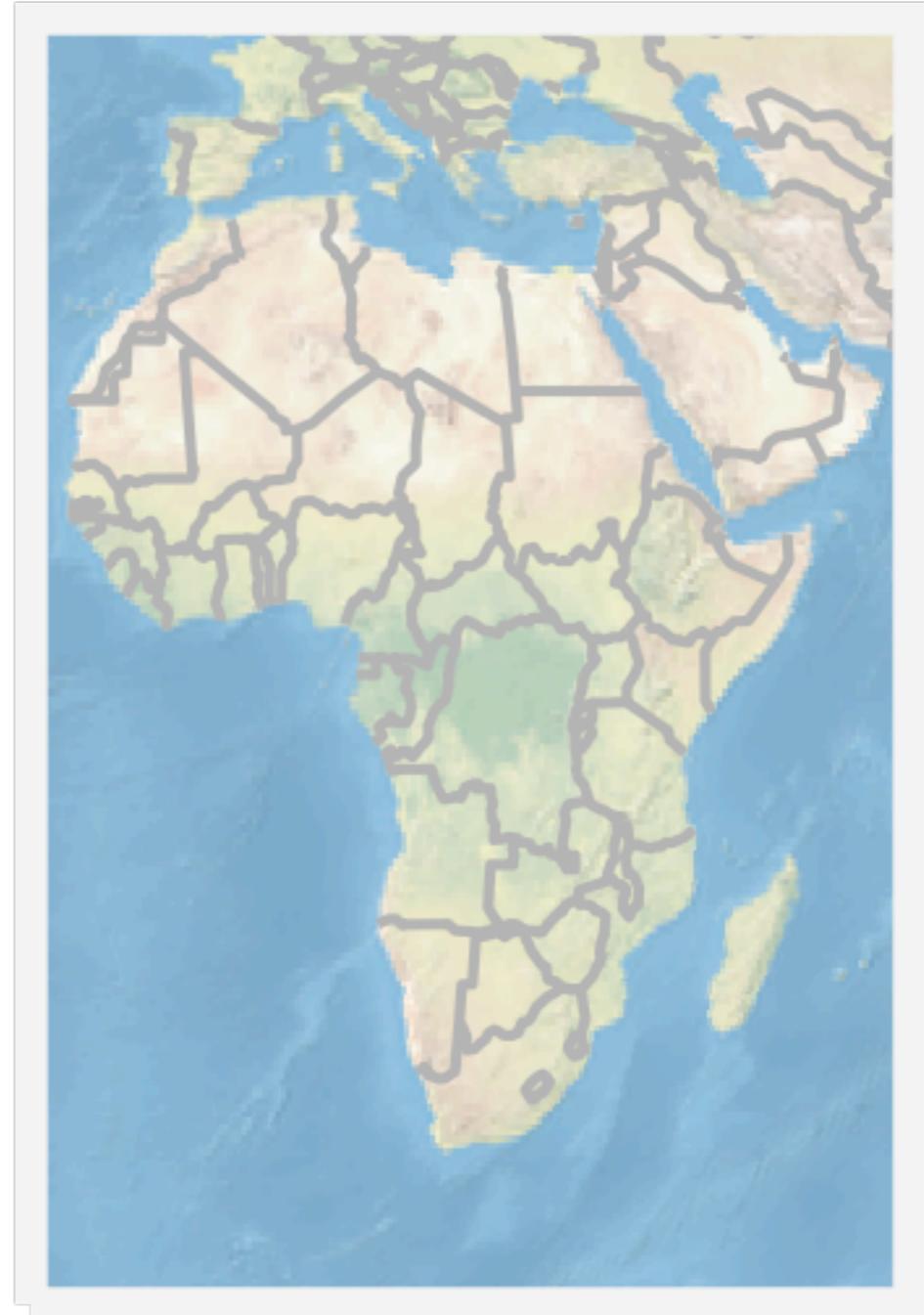
- We can plot arbitrary data within the map using the usual `ax.plot()` method and the appropriate `transform` argument
 - `ccrs.Geodetic()` - follow the great circles
 - `ccrs.PlateCarree()` - draw a straight line



Cartopy

scitools.org.uk/cartopy/

- To add a stock image as a background we can use `ax.stock_img()`
- Arbitrary images can be loaded with `ax.imshow()` and the appropriate `transform` argument



Cartopy

scitools.org.uk/cartopy/

- To switch to a globe representation, all we have to do is to change our projection to a globe based one:

- Mollweide
- Orthographic
- Robinson
- InterruptedGoodeHomolosine
- etc...

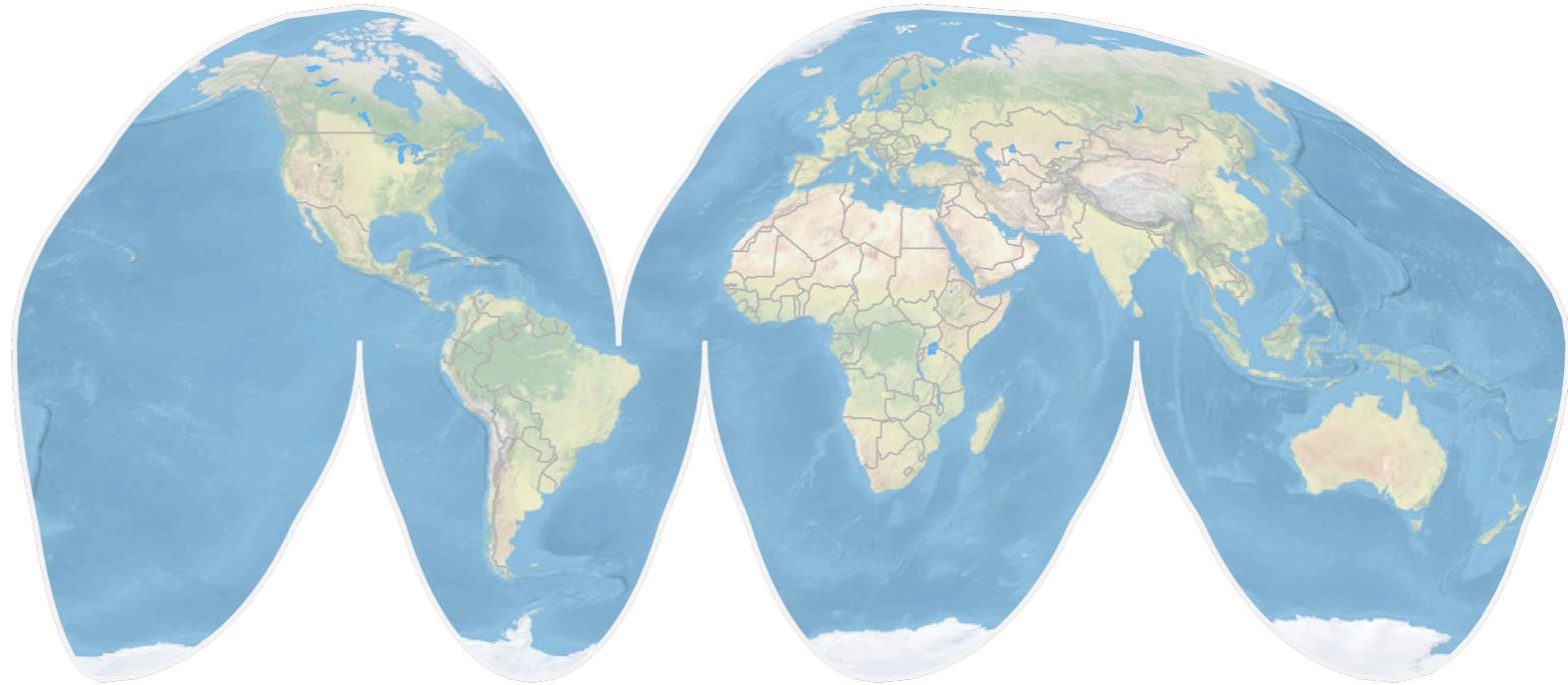


Cartopy

scitools.org.uk/cartopy/

- To switch to a globe representation, all we have to do is to change our projection to a globe based one:

- Mollweide
- Orthographic
- Robinson
- Interrupted Goode Homolosine
- etc...





Code - Mapping
<https://github.com/DataForScience/DataViz>

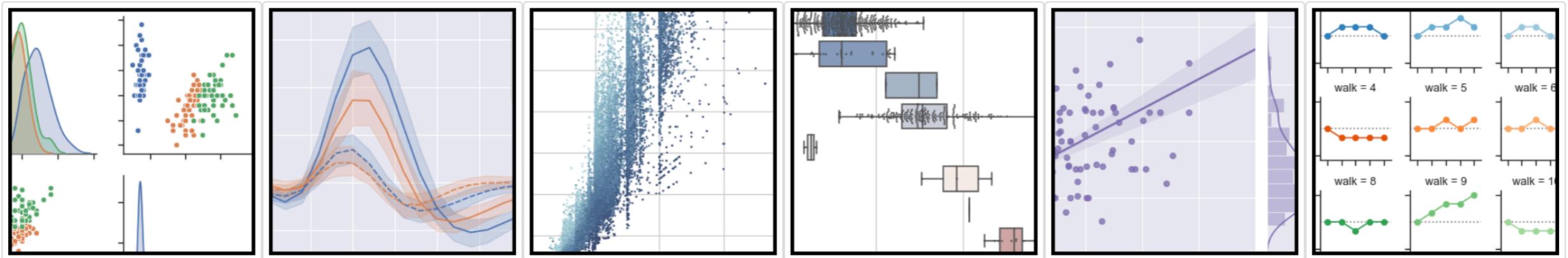


6. Seaborn

Why **seaborn**?

seaborn.pydata.org

seaborn: statistical data visualization



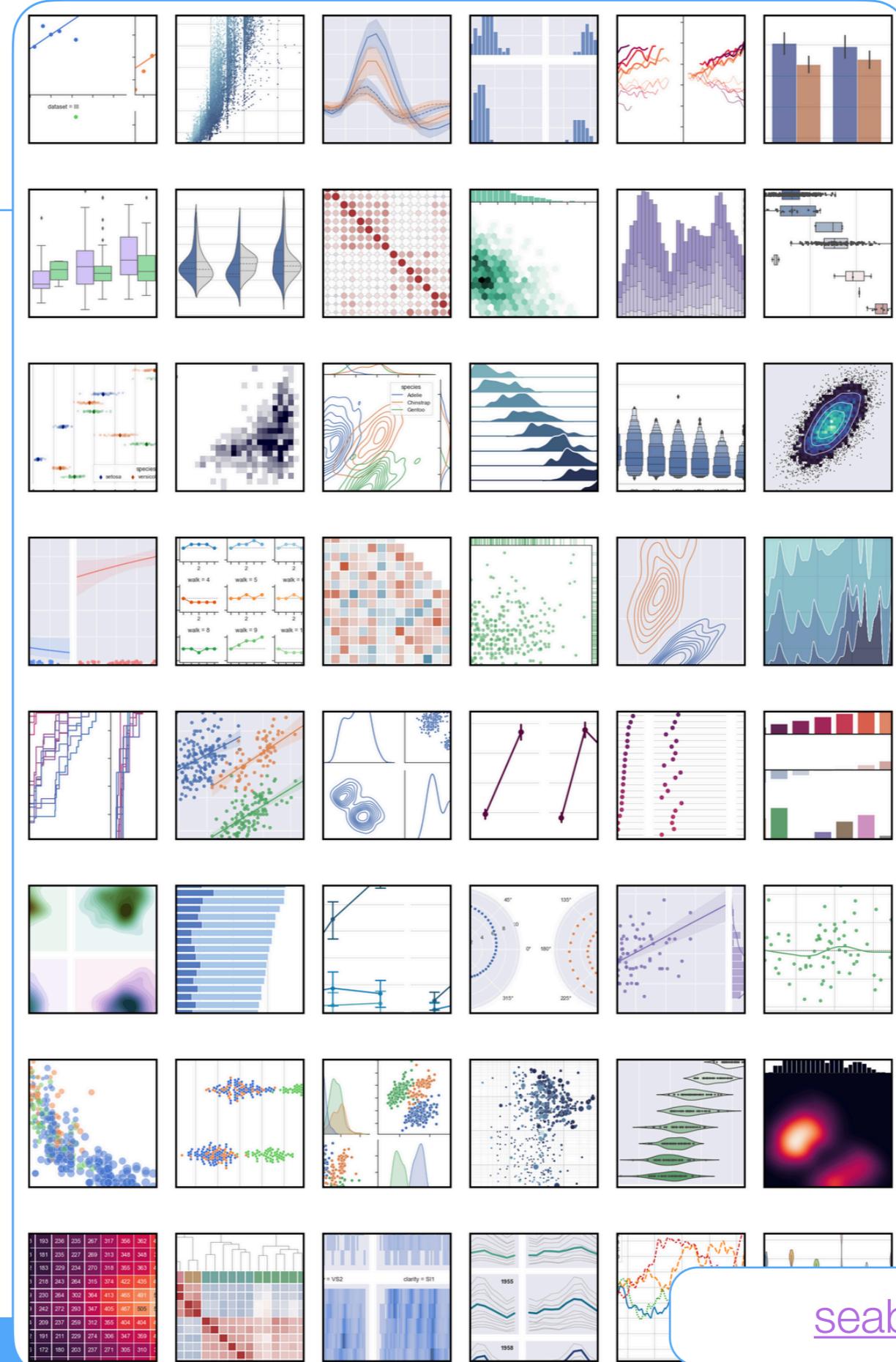
"**Seaborn** is a library for making statistical graphics in Python. It is built on top of **matplotlib** and closely integrated with **pandas** data structures.

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."



seaborn

Extensive gallery
that provides the
source code for all
the figures shown



seaborn.pydata.org

@bgoncalves



seaborn.pydata.org

- 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.

Datasets

seaborn.pydata.org

- 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 standard datasets.
- The datasets can be accessed easily by calling the `load_dataset` function with the file name (sans extension) as a parameter.

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

Datasets

seaborn.pydata.org

- 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 standard datasets.
- The datasets can be accessed 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

Types of plots

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

- `sns.scatterplot()`

Relationship plots

- `sns.lineplot()`

- `sns.stripplot()`

Categorical plots

- `sns.swarmplot()`

- `sns.boxplot()`

- `sns.violinplot()`

- `sns.boxenplot()`

- `sns.pointplot()`

- `sns.barplot()`

- `sns.countplot()`

Types of plots

Axes level

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

Figure level

Relationship plots

`sns.relplot()`

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

Categorical plots

`sns.catplot()`

- `kind="scatter"`
- `kind="line"`
- `kind="strip"`
- `kind="swarm"`
- `kind="box"`
- `kind="violin"`
- `kind="boxen"`
- `kind="point"`
- `kind="bar"`
- `kind="count"`

Figure vs Axes level

github.com/DataForScience/Seaborn

Figure level

```
1 fg = sns.relplot(x="total_bill", y="tip", data=tips, kind="scatter")
```

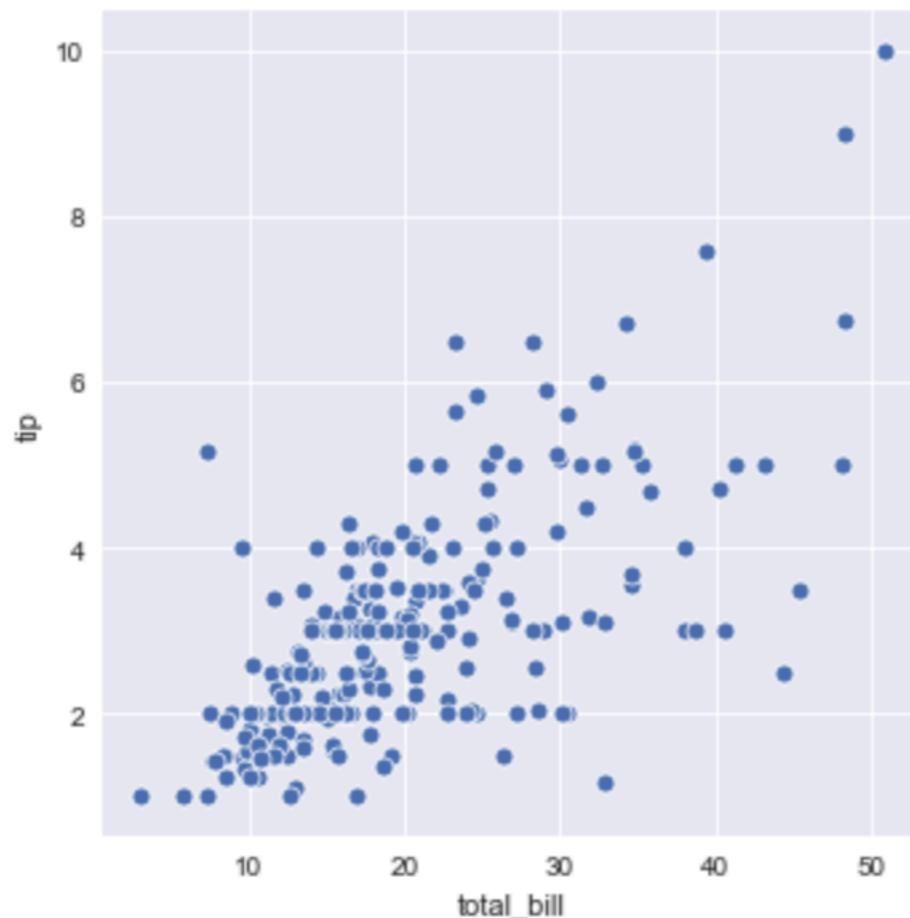
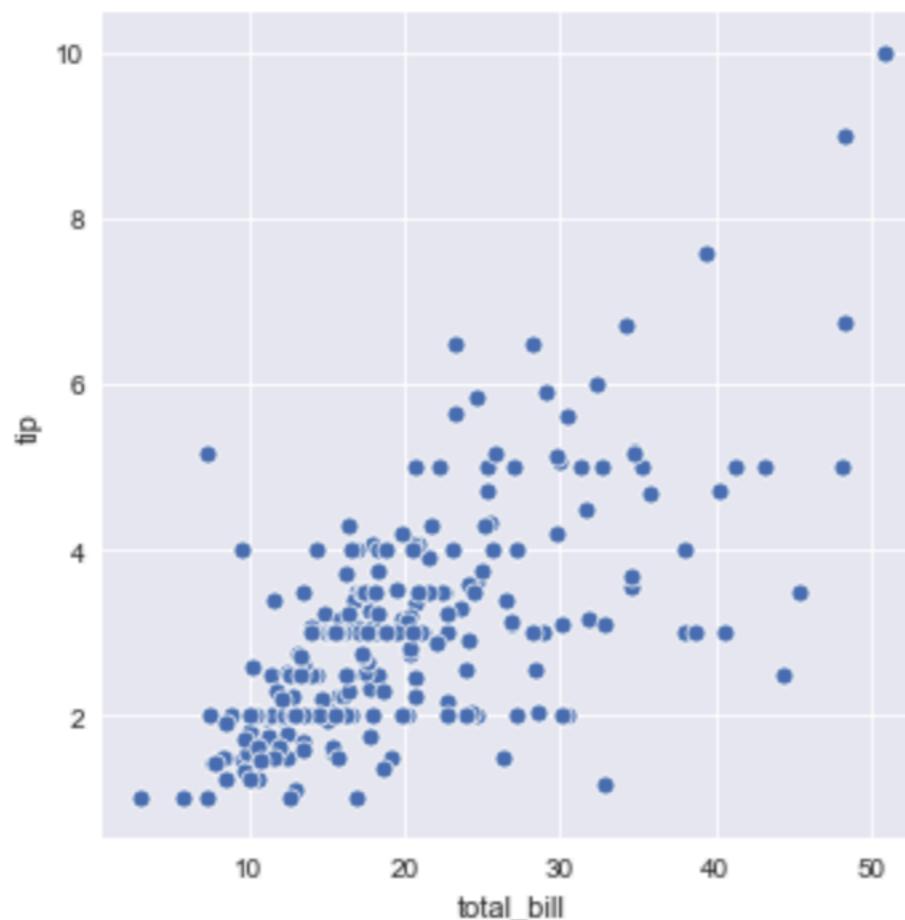


Figure vs Axes level

github.com/DataForScience/Seaborn

Figure level

```
1 fg = sns.relplot(x="total_bill", y="tip", data=tips, kind="scatter")
```



Axes level

```
1 ax = sns.scatterplot(x="total_bill", y="tip", data=tips)
```

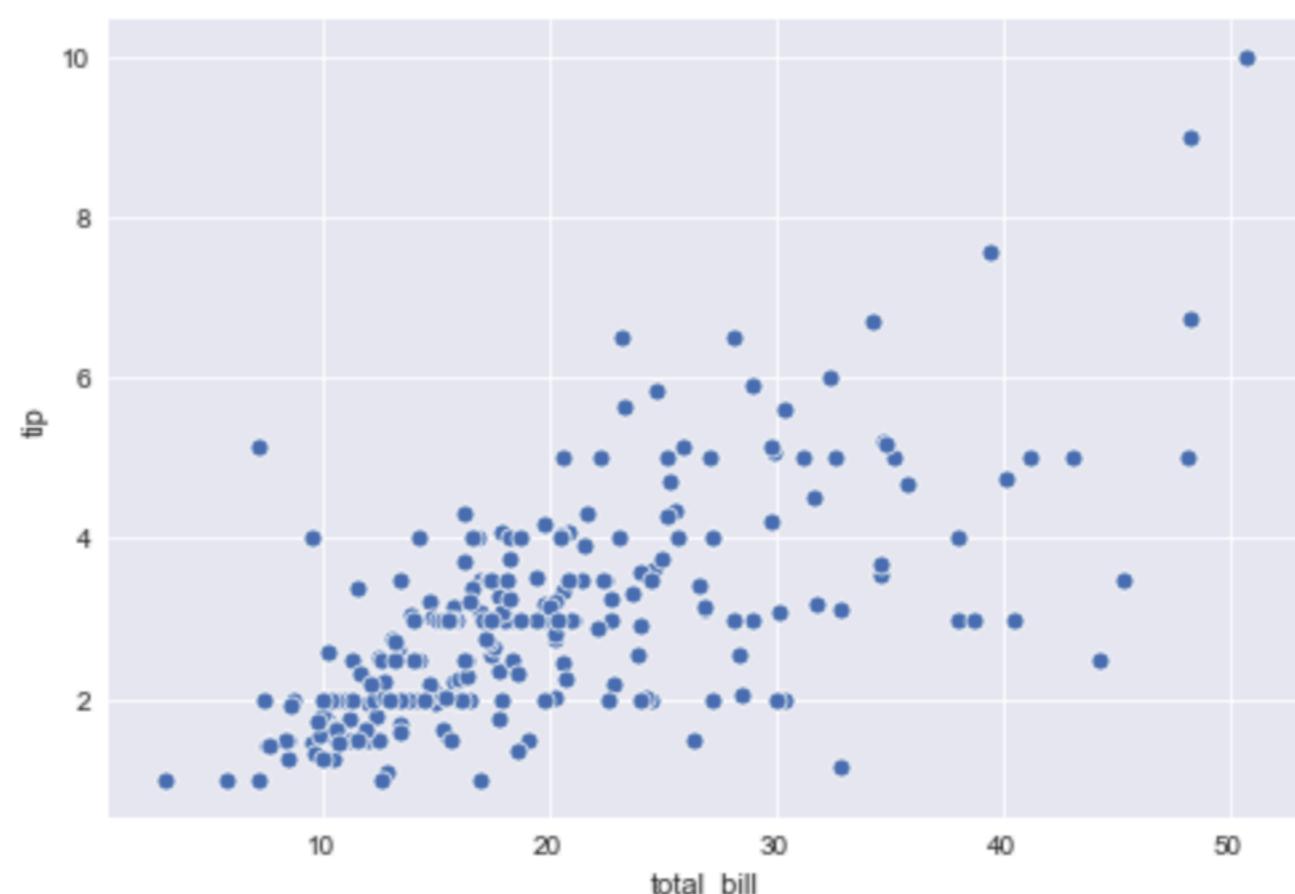
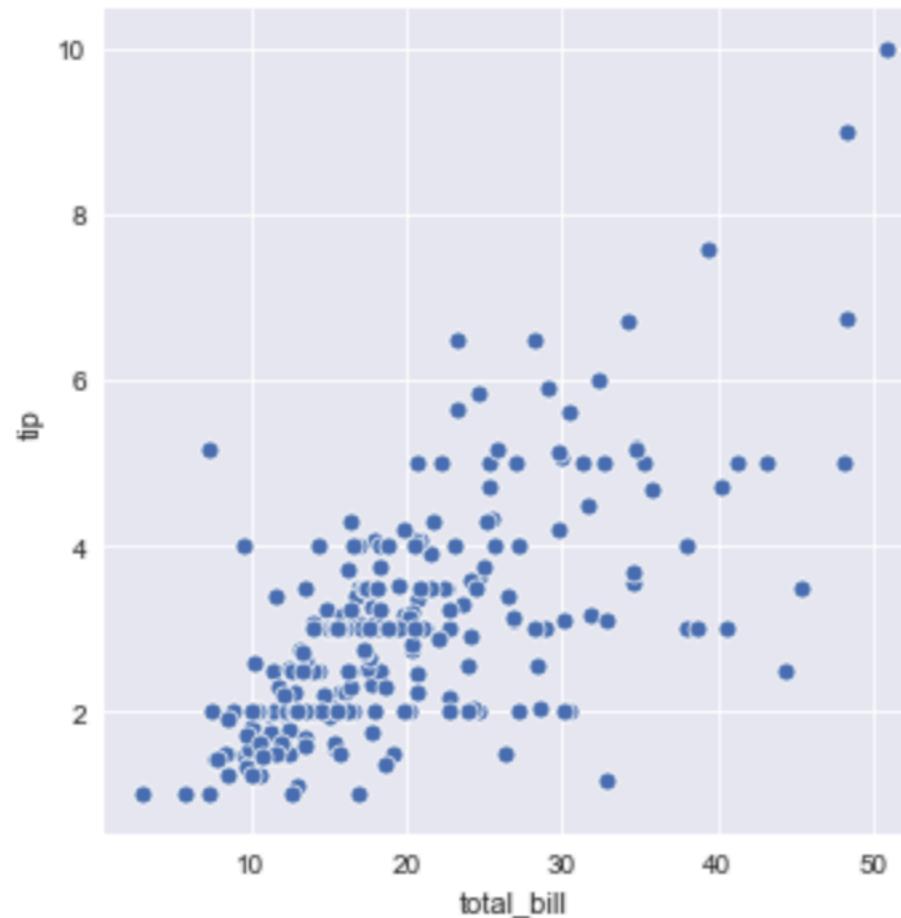


Figure vs Axes level

github.com/DataForScience/Seaborn

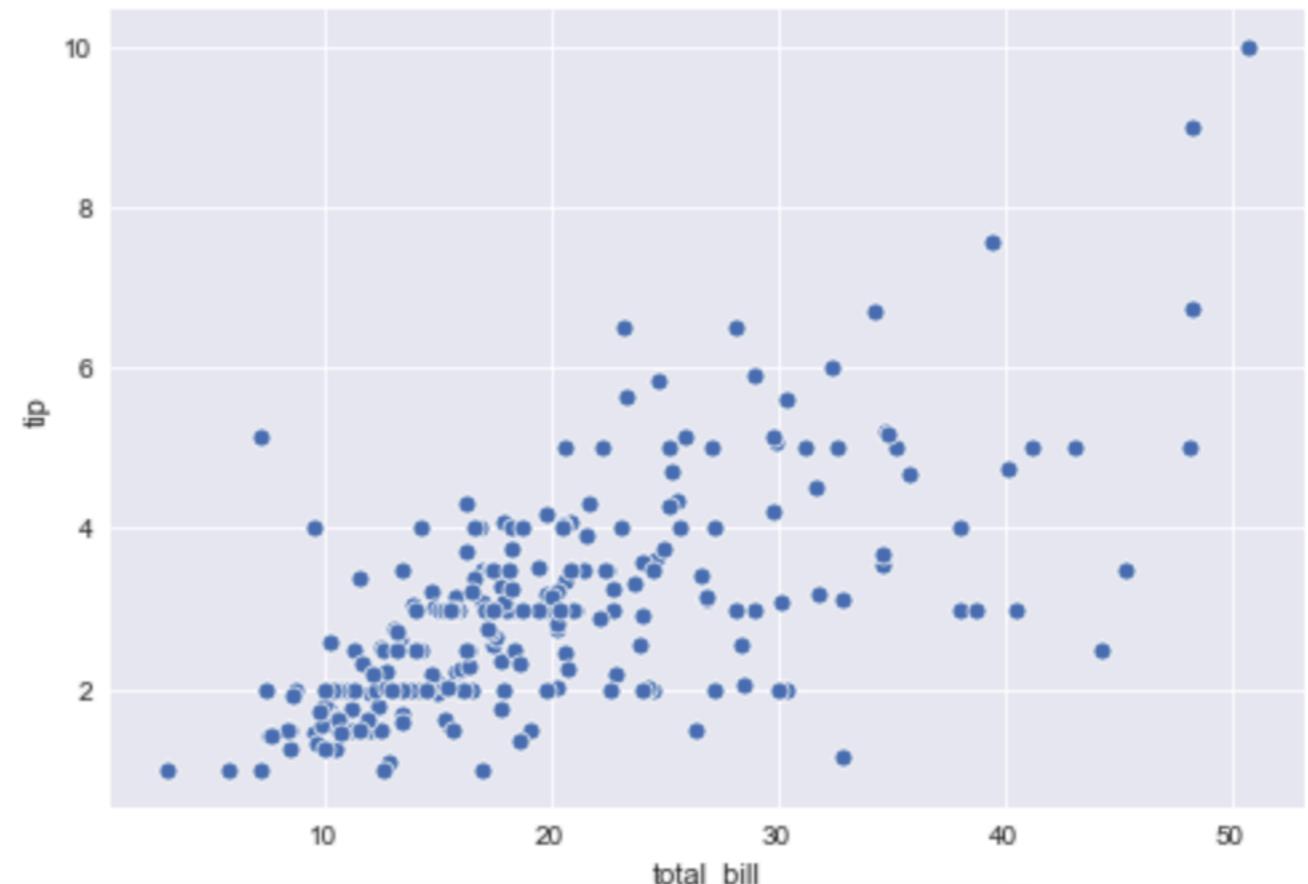
Figure level

```
1 fg = sns.relplot(x="total_bill", y="tip", data=tips, kind="scatter")
```



Axes level

```
1 ax = sns.scatterplot(x="total_bill", y="tip", data=tips)
```



The two approaches produce slightly different results.
For convenience, we will focus mostly on **Figure level** functions

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 **DataFrame** 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.

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

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.

Figure level functions

github.com/DataForScience/Seaborn

```
1 sns.relplot(x="total_bill", y="tip", data=tips, kind="scatter",
2               hue="sex", size="size",
3               col='smoker', row='day', style='time',
4               row_order=["Thur", "Fri", "Sat", 'Sun']
5 )
```

Figure level functions

github.com/DataForScience/Seaborn

```
1 sns.relplot(x="total_bill", y="tip", data=tips, kind="scatter",
2             hue="sex", size="size",
3             col='smoker', row='day', style='time',
4             row_order=["Thur", "Fri", "Sat", 'Sun']
5 )
```

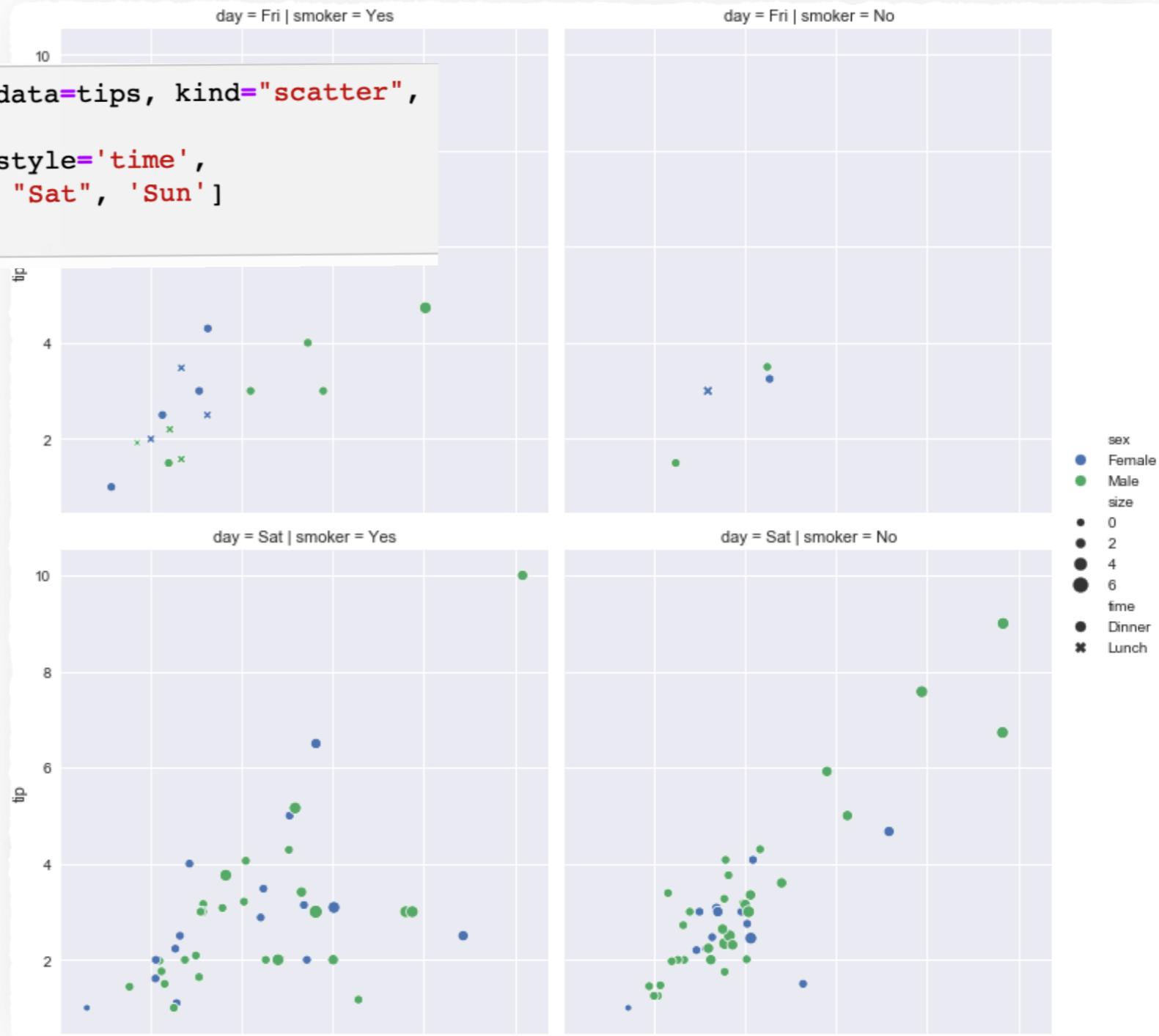


Figure level functions

github.com/DataForScience/Seaborn

```
1 sns.relplot(x="total_bill", y="tip", data=tips, kind="scatter",
2             hue="sex", size="size",
3             col='smoker', row='day', style='time',
4             row_order=["Thur", "Fri", "Sat", 'Sun']
5 )
```



FacetGrid

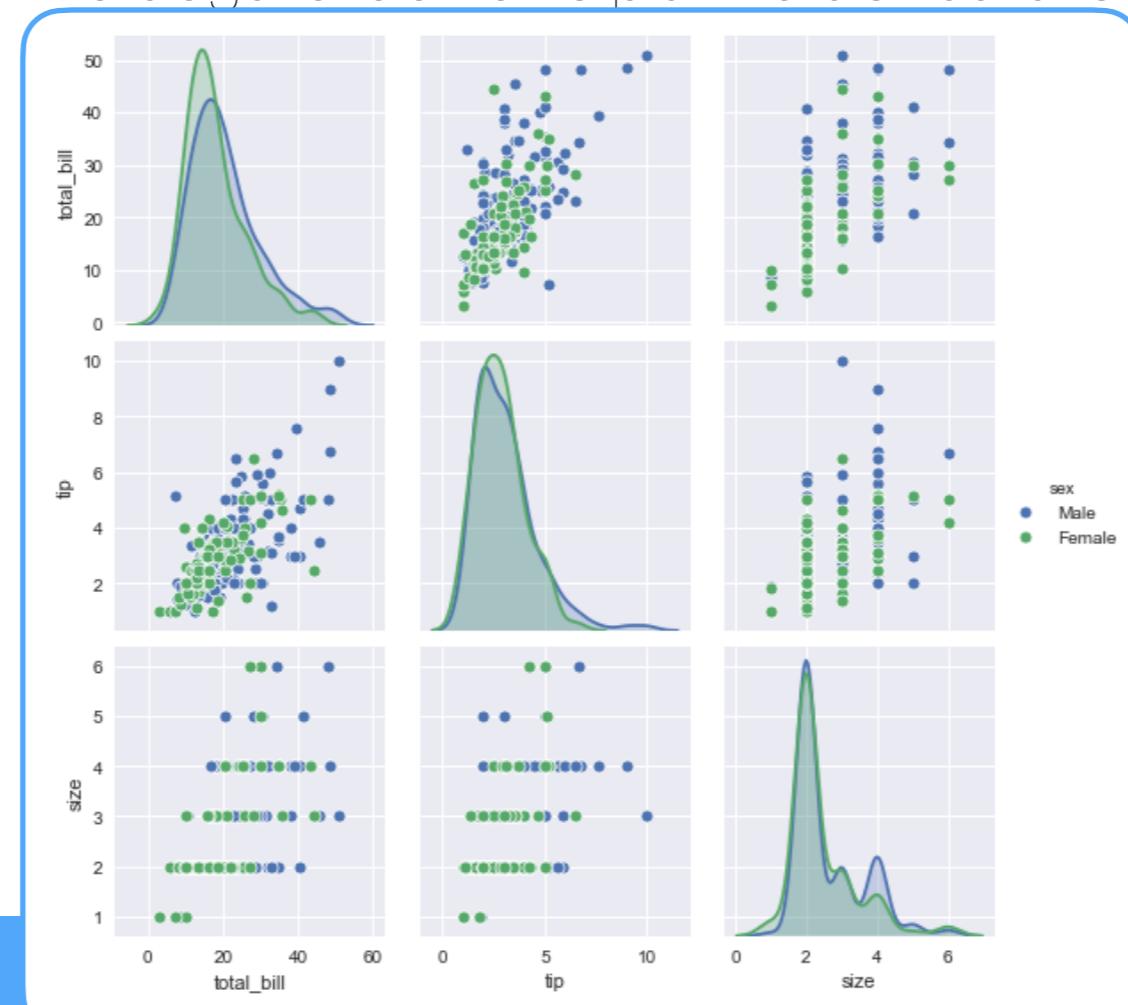
<https://seaborn.pydata.org/generated/seaborn.FacetGrid.html>

- **FacetGrid** is the object that **seaborn** uses in the background to generate the subplots.
- You can easily use **FacetGrid** with any **matplotlib** "compatible" plotting function.
- The process is divided into two parts:
 - 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 **function** that will generate the plot along with any extra parameters you require.
- Any function that modifies the currently active **Axis** object can be used (both **matplotlib** functions and **custom** functions)
- Any extra parameters passed to **map()** will be passed directly to the plotting function. This function should accept a ****kwargs** argument
- **FacetGrid** will pass the sliced data as **Series** to the plotting function

FacetGrid

<https://seaborn.pydata.org/generated/seaborn.FacetGrid.html>

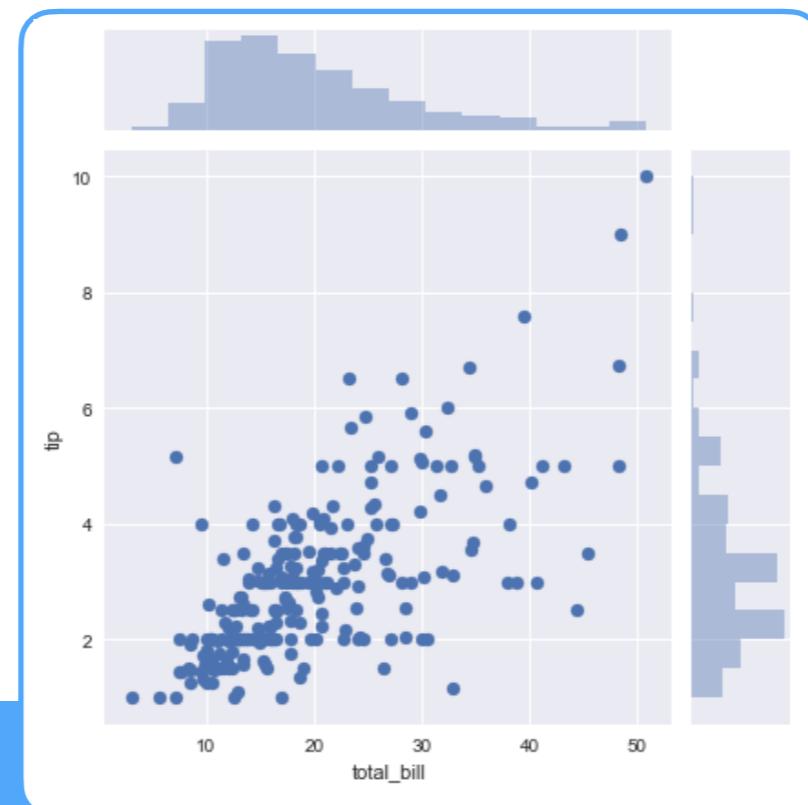
- After plotting the **FacetGrid** will contain a field **axes** containing an array with references to all the **matplotlib Axes** objects it generated.
- We can further customize the plot by directly using **matplotlib** functionality.
- **FacetGrid** is a general concept with several specialized variations within **seaborn**
 - **PairPlot** - specialized **FacetGrid** that automatically plots every pairwise relationship of numerical features. The diagonal elements plot the distributions of the respective feature.



FacetGrid

<https://seaborn.pydata.org/generated/seaborn.FacetGrid.html>

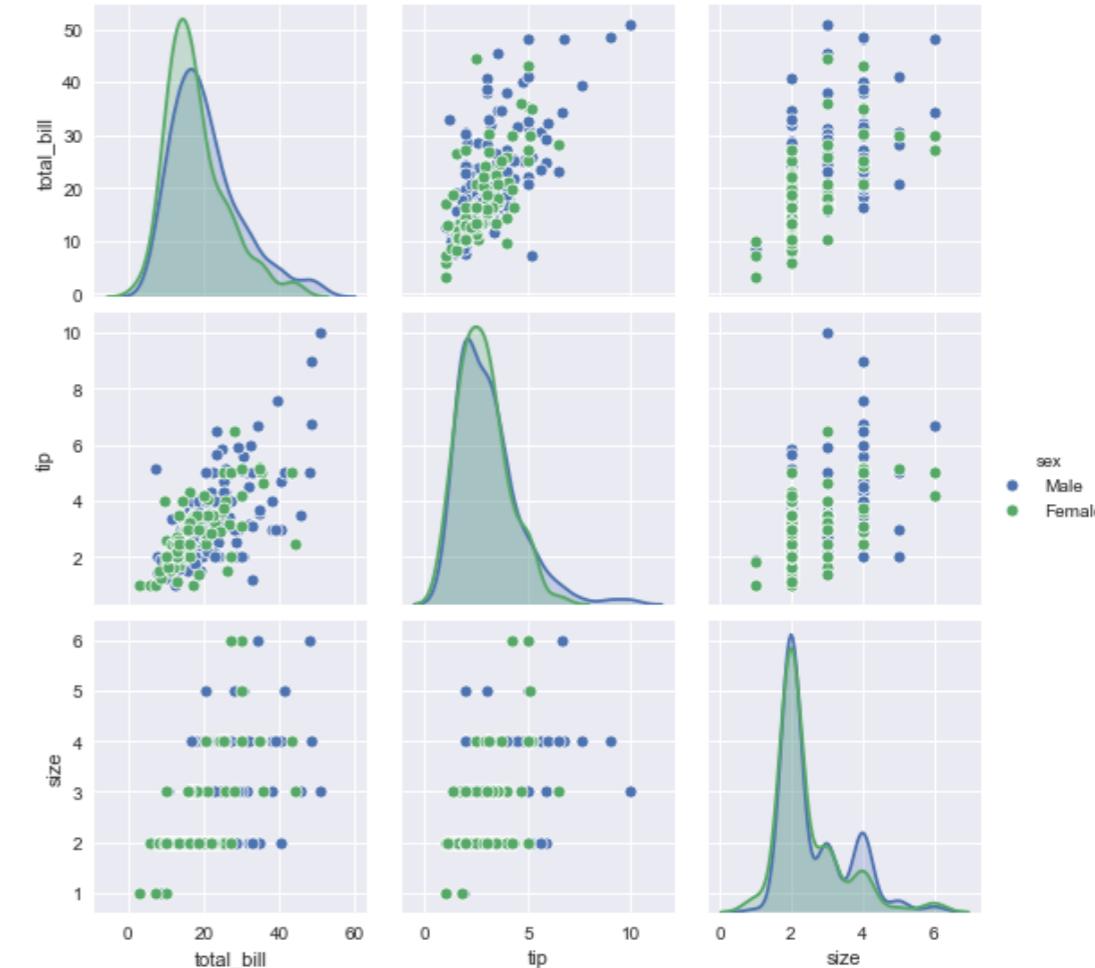
- After plotting the **FacetGrid** will contain a field **axes** containing an array with references to all the **matplotlib Axes** objects it generated.
- We can further customize the plot by directly using **matplotlib** functionality.
- **FacetGrid** is a general concept with several specialized variations within **seaborn**
 - **PairPlot** - specialized **FacetGrid** that automatically plots every pairwise relationship of numerical features. The diagonal elements plot the distributions of the respective feature.
 - **JointPlot** - specialized **FacetGrid** that uses marginal right and top plots to visualize the distributions of the X and Y features.



PairPlot

<https://seaborn.pydata.org/generated/seaborn.pairplot.html>

- Plots every pairwise relationship of numerical features.
- Diagonal elements plot the distributions of each feature.
- **hue** (and **hue_order**) - further subdivide the data set by color.
- **varslist** - list of variables to use. Default is to use all numerical variables.
- **x_vars, y_vars** - list of variables to use along the x and y axis
- **kind** - kind of plot to use for pairwise relationships: '**scatter**', '**kde**', '**hist**', '**reg**'
- **diag_kind** - kind of plot to use for in the diagonal plots: '**hist**', '**kde**', **None**
- **markers** - **matplotlib** marker(s) to use. If list, it should have same length as **hue**
- Array of **Axes** objects in **axes** (like **FacetGrid**)



JointPlot

<https://seaborn.pydata.org/generated/seaborn.jointplot.html>

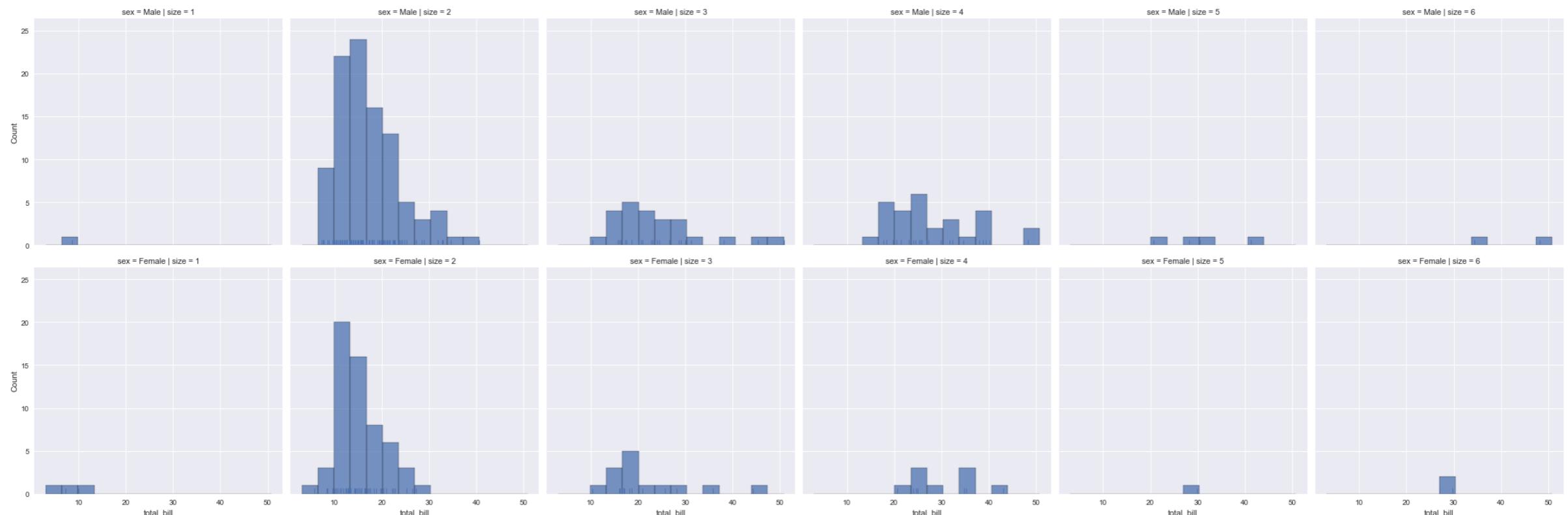
- Draw a plot of two variables with bivariate and univariate graphs.
- Marginal plots on the top and right hand side visualize the distributions of the X and Y features.
- **hue** (and **hue_order**) - further subdivide the data set by color.
- **kind** - The type of plot to draw: “**scatter**”, “**kde**”, “**hist**”, “**hex**”, “**reg**”, “**resid**”
- **Axes** objects are available as in specific fields: **ax_join**, **ax_marg_x**, **ax_marg_y**
- Additional layers can be added to the JointPlot using **plot_joint()** and **plot_marginals()**



Distribution plots

<https://seaborn.pydata.org/generated/seaborn.displot.html>

- We have covered the two main kinds of **Figure Level** plotting functions, `sns.relplot()` for Relationship Plots and `sns.catplot()` for Categorical Plots.
- The **third class** of function is `sns.displot()` for Distribution Plots and it allows us to generate a **FacetGrid** plot of the distributions underlying our data.
- Supports univariate and bivariate distribution of data
- Allows slicing using `hue`, `row`, `col`, etc.



Distribution plots

<https://seaborn.pydata.org/generated/seaborn.displot.html>

- `sns.displot(x, y, hue, data, kind)`

- **x, y** - column names to plot in each axes. Specify only **x** for univariate distribution and both **x** and **y** for bivariate distributions
- **hue** - column name specifying how to color each data element
- **data** - dataframe containing the data to plot
- **row, col** - Column names to define the rows and columns of the **FacetGrid**
- **kind** - string specifying the type of plot to produce
- **rug** - Boolean flag determining whether or not to add a rug plot highlighting the locations of the data points
- **kind** - The kind of plot to generate:
 - “**hist**” - Plot 1D or 2D histogram
 - “**kde**” - Plot histogram using Kernel Density Estimation
 - “**ecdf**” - Plot empirical Cumulative Distribution Function

Types of plots

Axes level

- `sns.histplot()`
- `sns.kdeplot()`
- `sns.ecdfplot()`

Figure level

Distribution plots
`sns.displot()`

- `kind="hist"`
- `kind="kde"`
- `kind="ecdf"`



Code - Seaborn

<https://github.com/DataForScience/Seaborn>



Code - Seaborn
<https://github.com/DataForScience/DataViz>

Question

- How was the technical level?

- 1 — Too Low

- 2 — Low

- 3 — Just Right

- 4 — High

- 5 — Too High

Question

- How was the level of Python code/explanations?

- 1 — Too Low

- 2 — Low

- 3 — Just Right

- 4 — High

- 5 — Too High

Events



graphs4sci.substack.com



<http://paypal.me/data4sci>



Deep Learning for Everyone

Apr 22, 2021 - 5am-9am (PST)

Time Series for Everyone

May 4, 2021 - 5am-9am (PST)

Advanced Time Series for Everyone

May 20, 2021 - 5am-9am (PST)

Natural Language Processing for Everyone

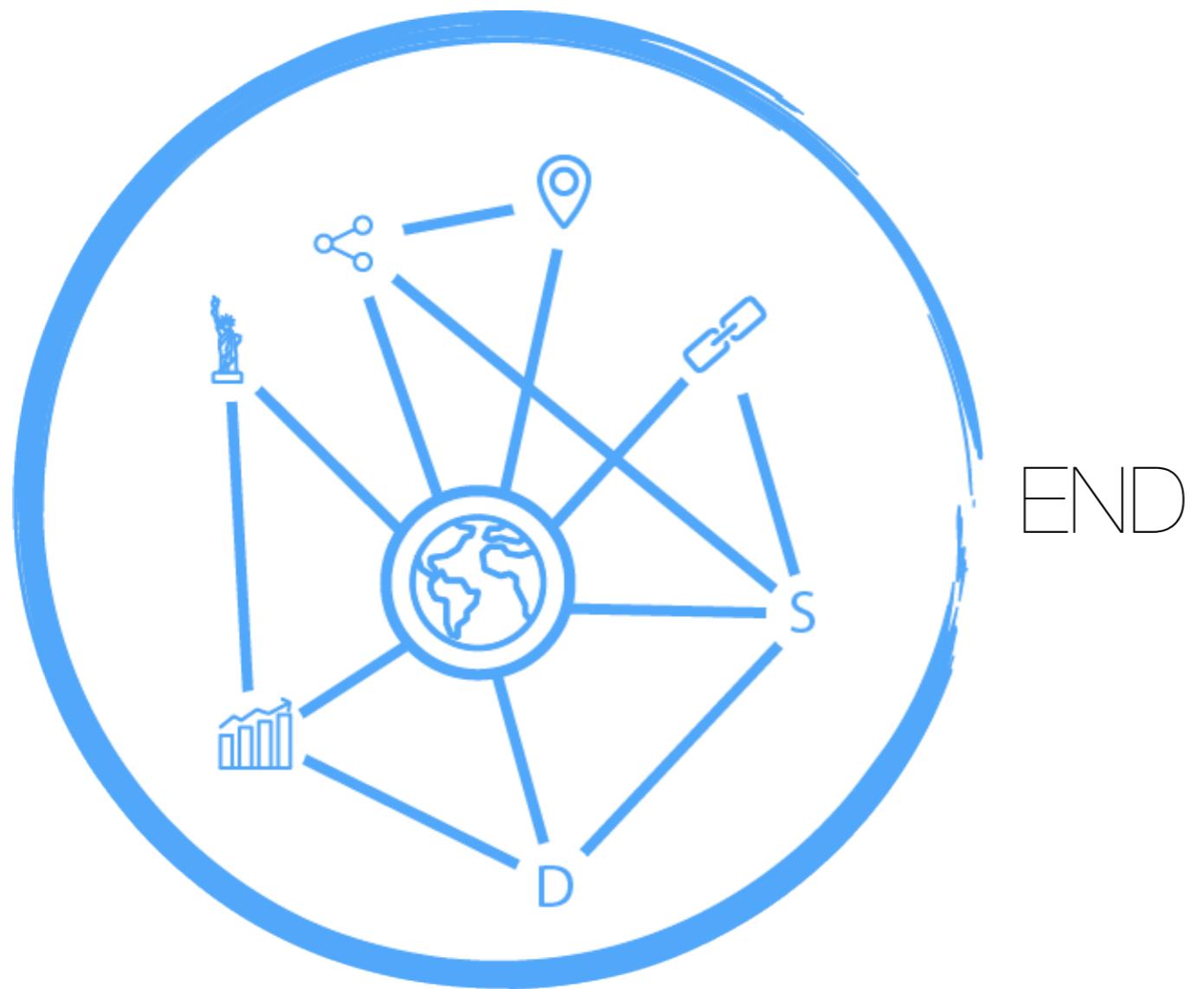
Jun 9, 2021 - 5am-9am (PST)

Advanced NLP for Everyone

Jun 25, 2021 - 5am-9am (PST)

Natural Language Processing (NLP) from Scratch

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



END