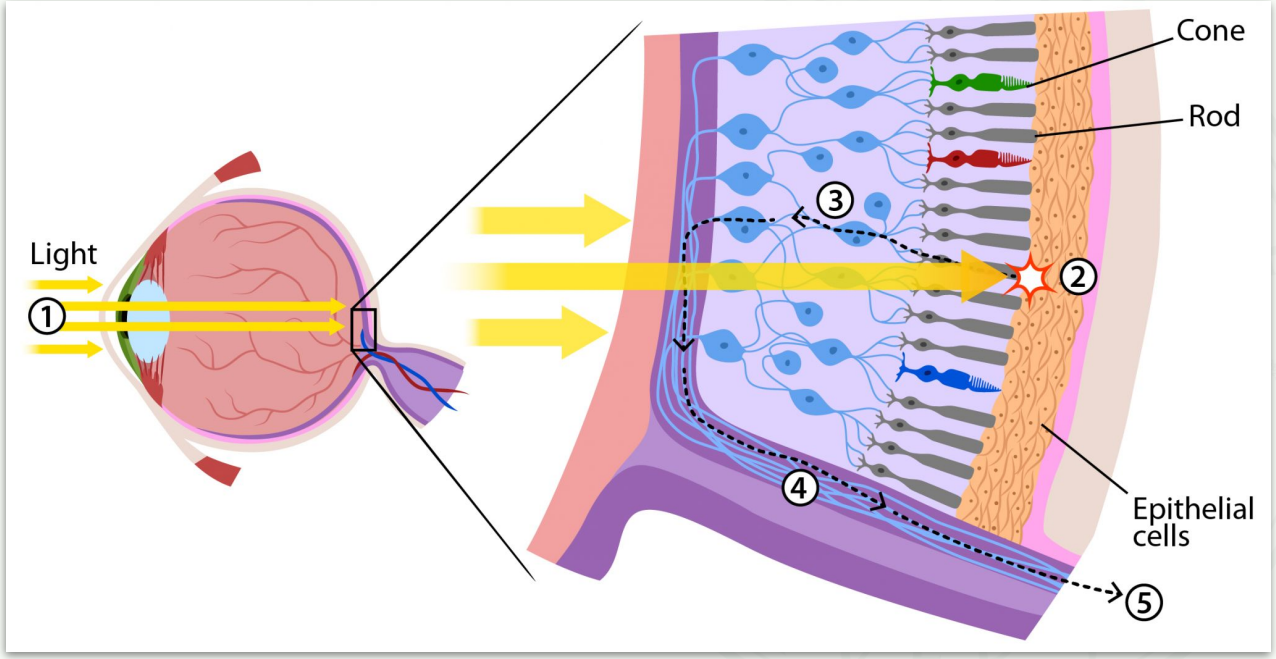
Colors, Storytelling, and IDA/EDA

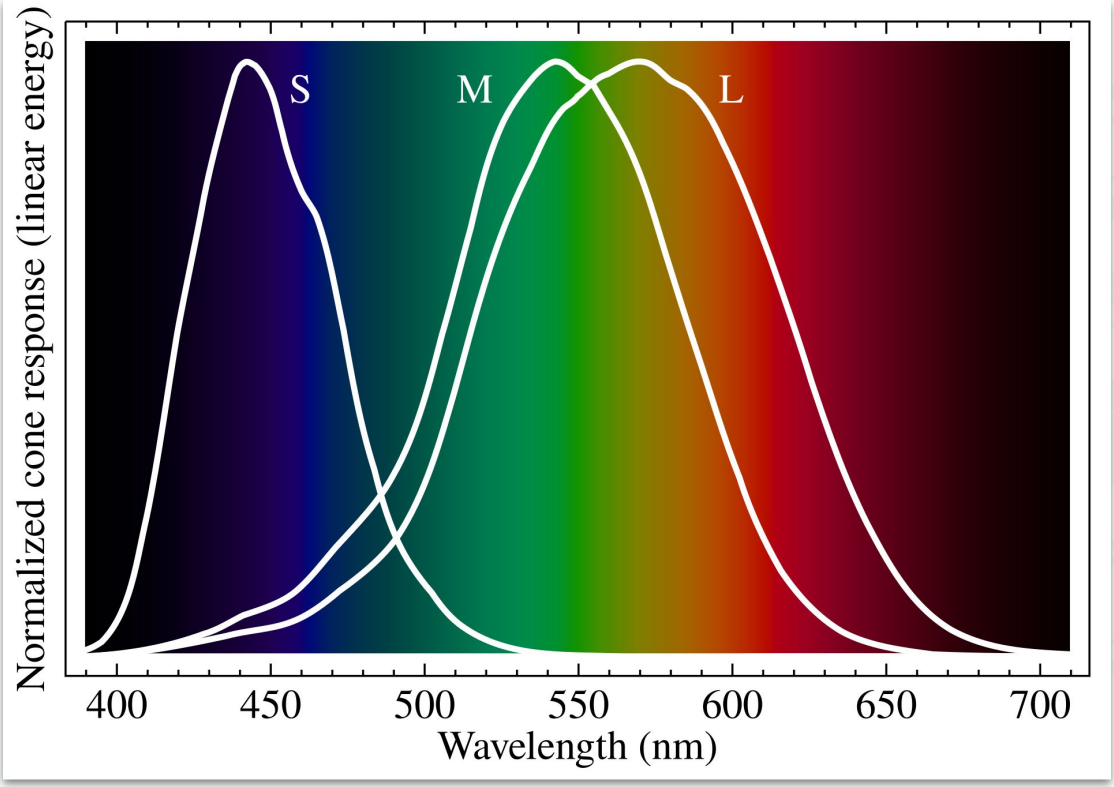
Prof. Murillo
Computational Mathematics, Science and Engineering
Michigan State University

Color: How Do Eyes Work?



- What your brain “sees” is not what your eye “sees” (e.g., blind spot from optic nerve, image is reversed, there are floaties the fluid, etc.)
- human eye is “backwards”
- our detectors are in the form of rods and cones

Color: Cone Response

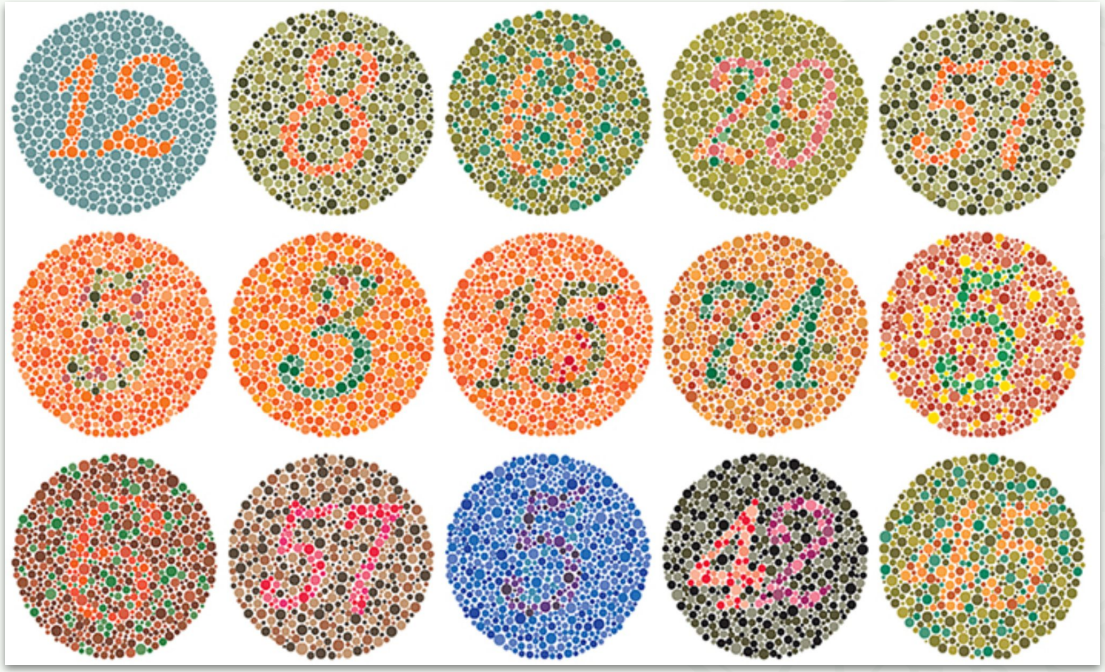


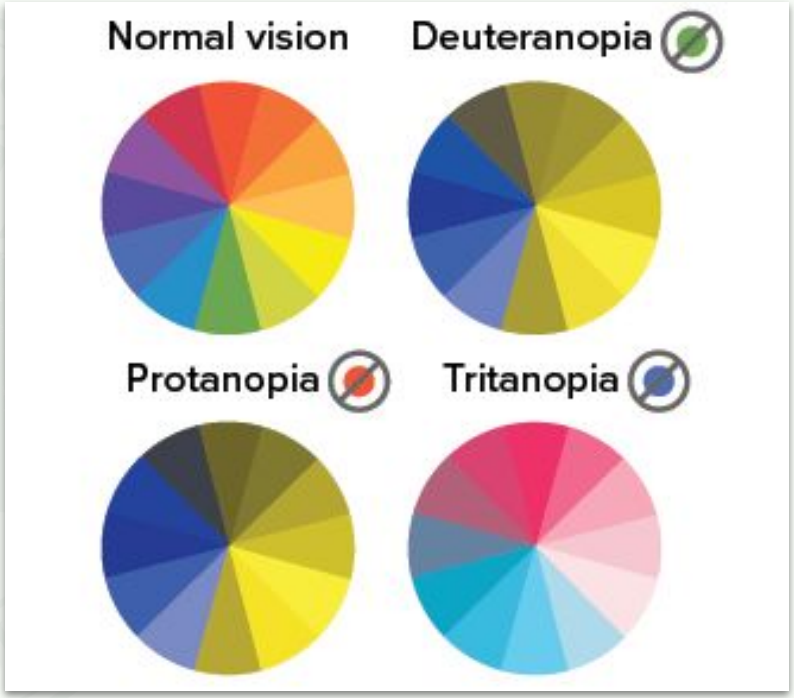
wavelengths:

- S = short
- M = medium
- L = long

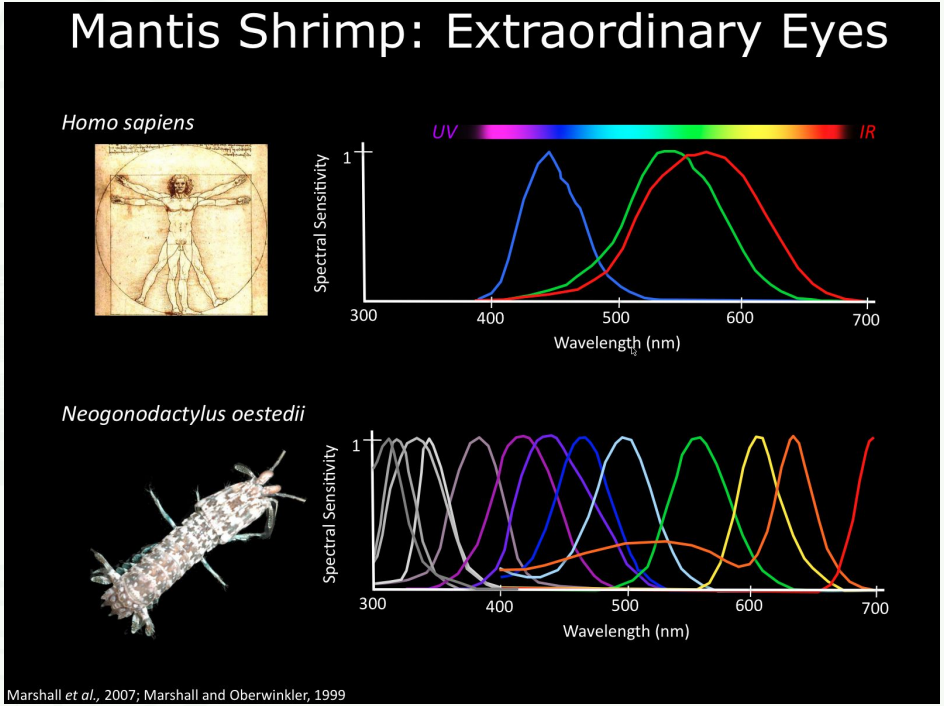
A very small percentage of the human population are tetrachromes.

Color: Types of Blindness





Color: Mantis Shrimp (We are all (somewhat) colorblind!)



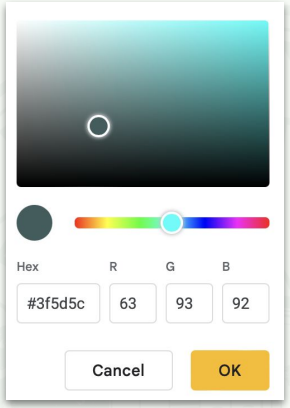
Matplotlib Provides Useful Palettes

Choosing Colormaps in Matplotlib

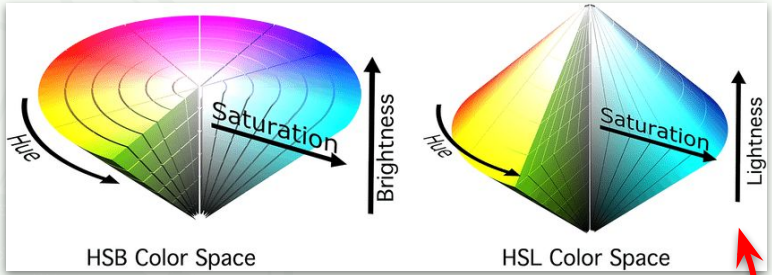
Matplotlib has a number of built-in colormaps accessible via `matplotlib.colormaps`. There are also external libraries that have many extra colormaps, which can be viewed in the Third-party colormaps section of the Matplotlib documentation. Here we briefly discuss how to choose between the many options. For help on creating your own colormaps, see Creating Colormaps in Matplotlib.

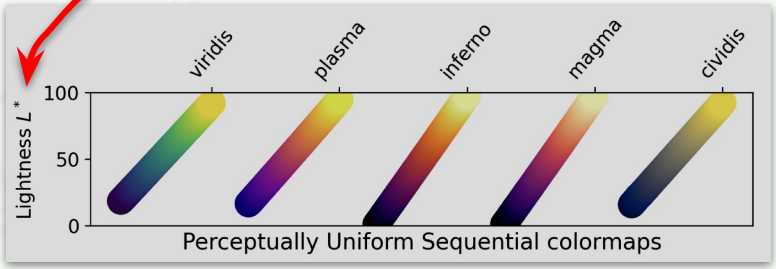
To get a list of all registered colormaps, you can do:

```
from matplotlib import colormaps
list(colormaps)
```



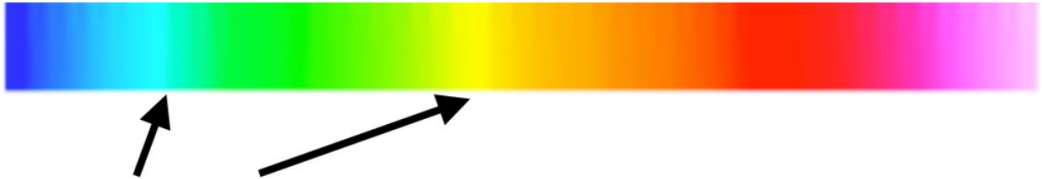
Be aware of color spaces: there are many with various uses.





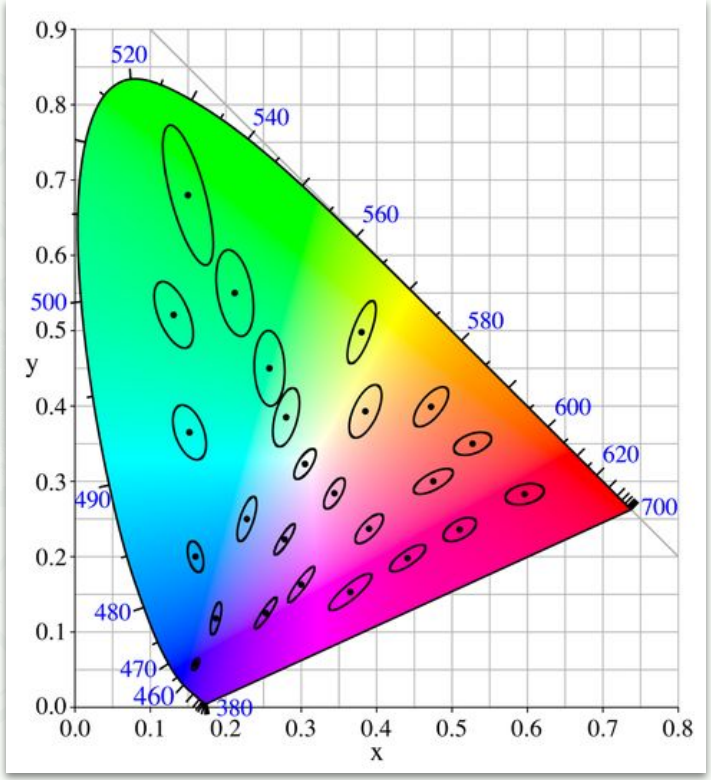
Perceptual Uniformity

Non Perceptual Uniform Colormap



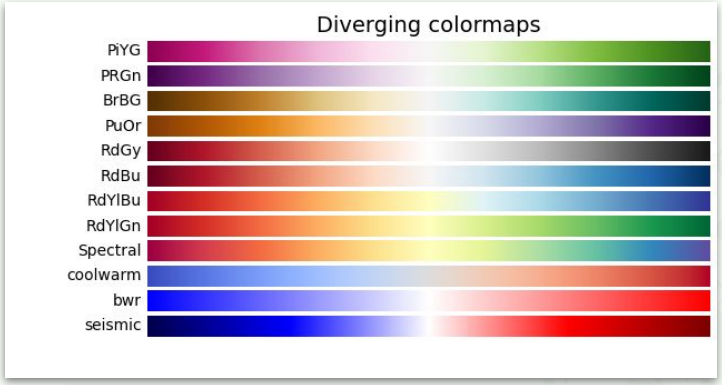
Features of the Colormap not of Changes in Data

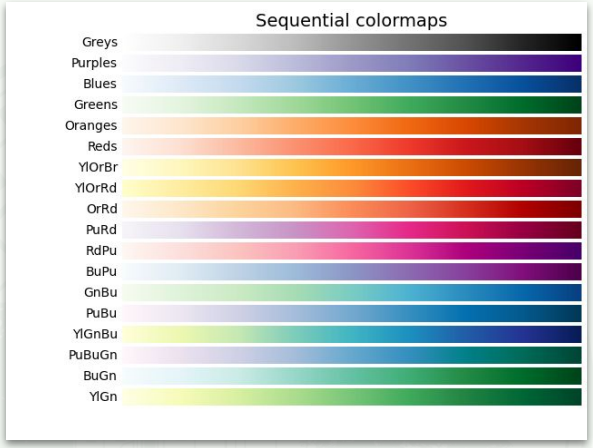
Perceptual Uniform Colormap

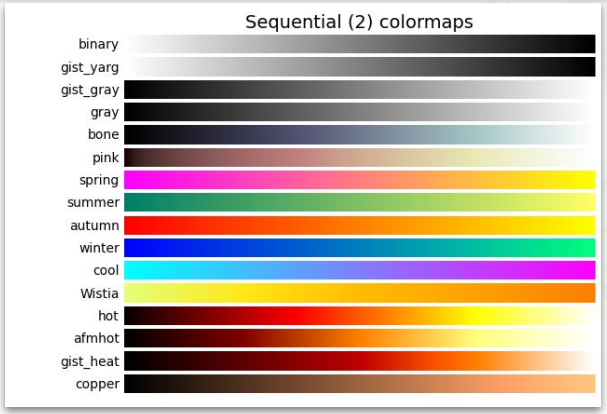


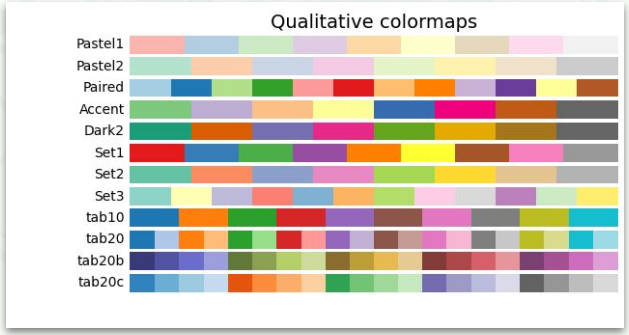
MacAdam Ellipses

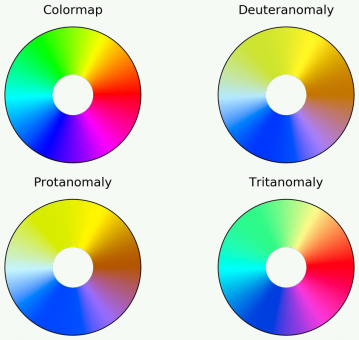
Colors Communicate In Different Ways

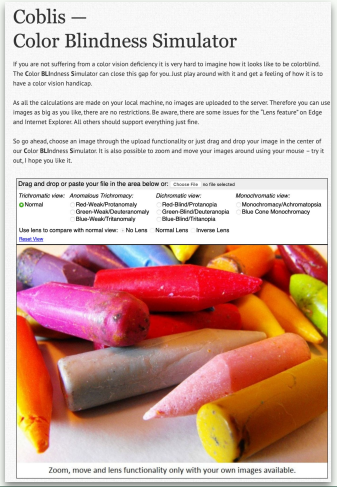








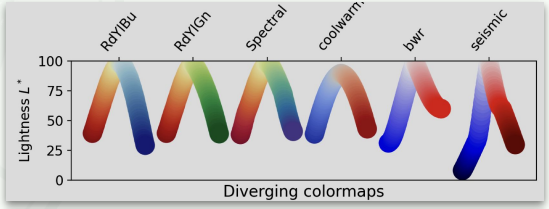


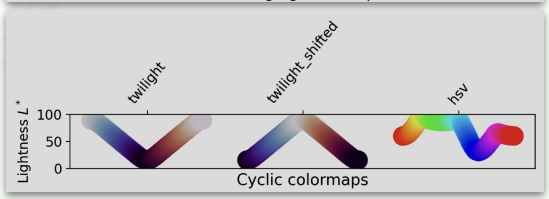


Think Carefully About Your Project's Use of Color

Palette Type	Description	Use Cases	Examples
Sequential	Ordered series of colors with a clear progression	Representing ordered data or continuous variables	Light blue to dark blue; Yellow to red
Diverging	Contrasting colors on the extremes with a neutral color in the middle	Showing deviation from a central value or emphasizing extremes	Blue-White-Red; Purple-Gray-Green
Qualitative	Distinct colors with no inherent order	Representing categorical data or discrete classes	Set of primary and secondary colors
Cyclic	Colors that form a natural cycle	Representing periodic data or angular values	Rainbow colors; Day-night cycle
Binary	Two contrasting colors	Representing binary or boolean data	Black and white; Blue and orange
Multi-sequential	Multiple sequential scales combined	Representing multiple related variables simultaneously	Two-tone heatmaps
Accent	Main palette with one or more standout colors	Highlighting specific data points or categories	Grayscale with a bright accent color

Palette Type	Colorblind-friendly	Perceptually uniform	Domain-specific
Sequential	✓	✓✓	✓
Diverging	✓	✓✓	✓
Qualitative	✓✓	N/A	✓
Cyclic	✓	✓	✓
Binary	✓✓	N/A	✓
Multi-sequential	✓	✓	✓
Accent	✓	N/A	✓

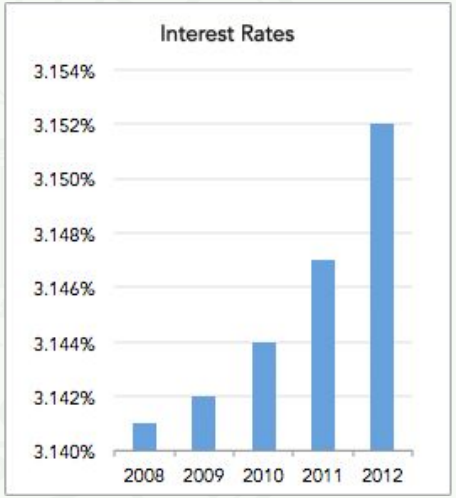


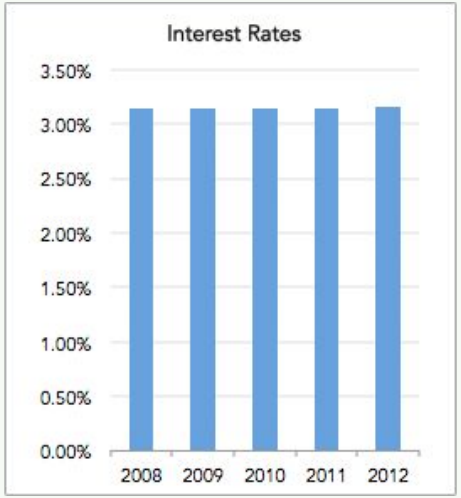


- **diverging**: there is a baseline (e.g., temperature, belief, budget)
- **cyclic**: angle, direction, daily/weekly (e.g., seasons, compass)

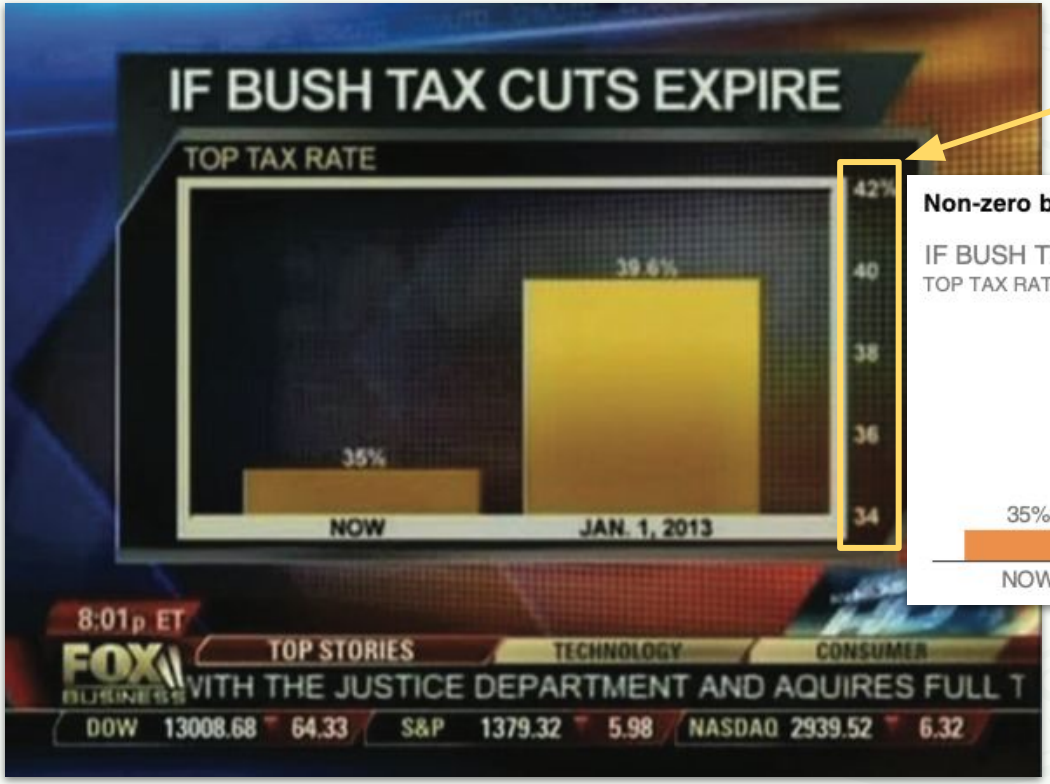
Psychology of Visualization

Same Data, Different Y-Axis

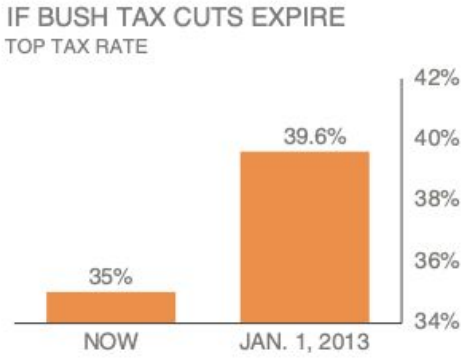




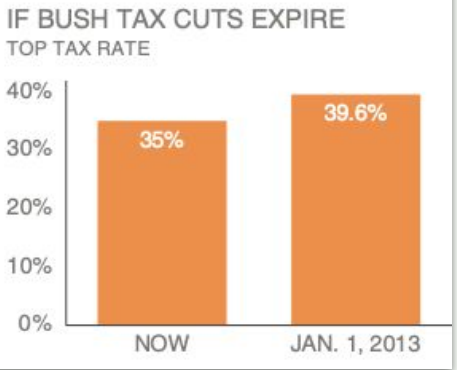
In The News: Example of Misleading Plot



Non-zero baseline: as originally graphed



Zero baseline: as it should be graphed



Overuse Of Color Can Mislead

This creates a “puzzle” for the viewer, which is to keep in their mind the ordering of colors in the rainbow as they look over the chart.

Worse, the viewer has to “turn off” social biases, such as “red = bad”, or bright colors are more important (e.g., yellow).

Country Level Sales Rank Top 5 Drugs

Rainbow distribution in color indicates sales rank in given country from #1 (red) to #10 or higher (dark purple)

Country	A	B	C	D	E
AUS	1	2	3	6	7
BRA	1	3	4	5	6
CAN	2	3	6	12	8
CHI	1	2	8	4	7
FRA	3	2	4	8	10
GER	3	1	6	5	4
IND	4	1	8	10	5
ITA	2	4	10	9	8
MEX	1	5	4	6	3
RUS	4	3	7	9	12
SPA	2	3	4	5	11
TUR	7	2	3	4	8
UK	1	2	3	6	7
US	1	2	4	3	5

Top 5 drugs: country-level sales rank

RANK	1	2	3	4	5+
COUNTRY | DRUG	A	B	C	D	E
Australia	1	2	3	6	7
Brazil	1	3	4	5	6
Canada	2	3	6	12	8
China	1	2	8	4	7
France	3	2	4	8	10
Germany	3	1	6	5	4
India	4	1	8	10	5
Italy	2	4	10	9	8
Mexico	1	5	4	6	3
Russia	4	3	7	9	12
Spain	2	3	4	5	11
Turkey	7	2	3	4	8
United Kingdom	1	2	3	6	7
United States	1	2	4	3	5

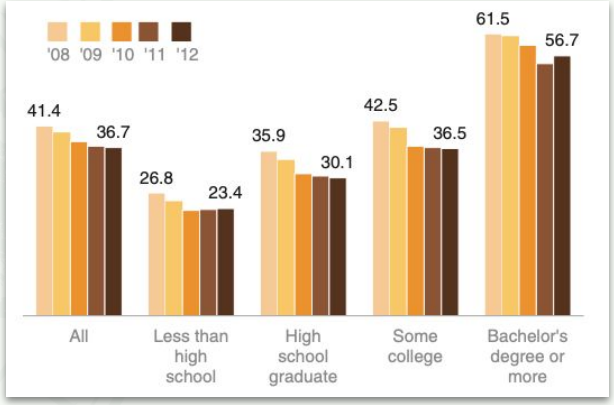
remake

Walkthrough of Plot Design

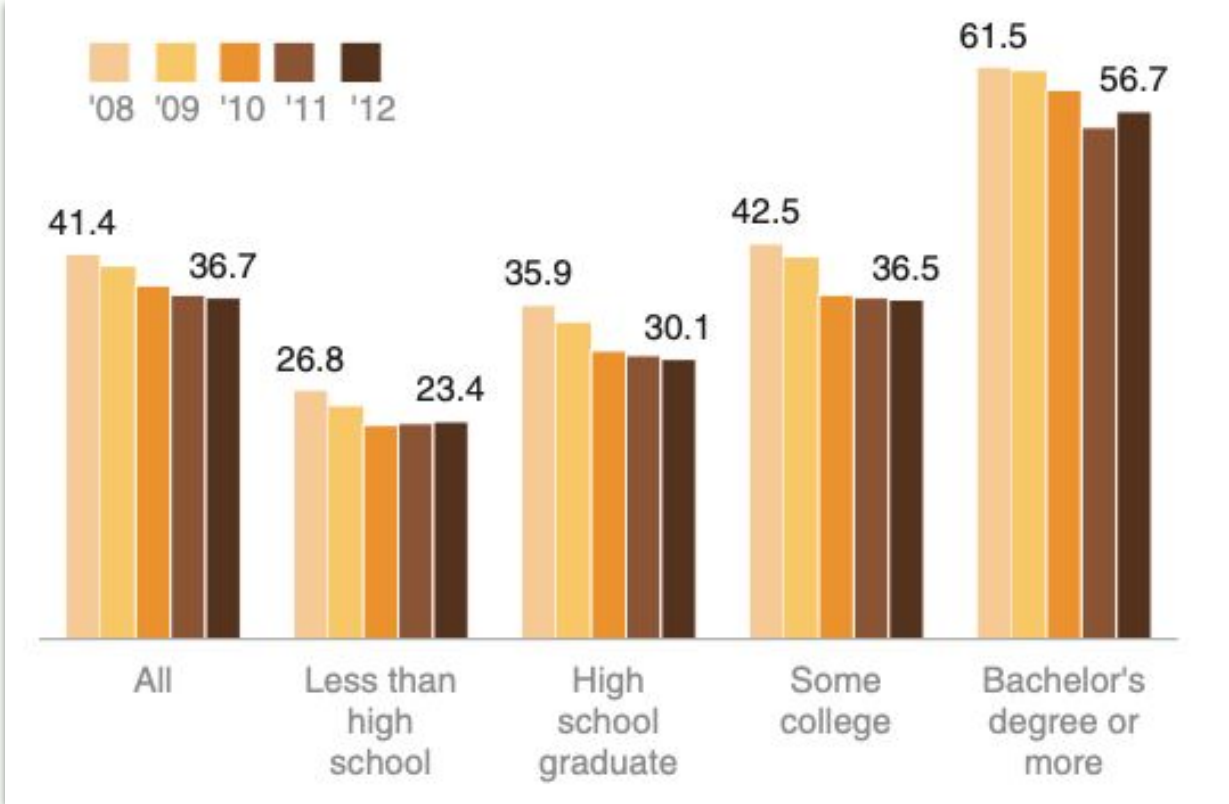
Let's take some reasonable plots and see if we can apply some of these ideas to improve them.

The most important goal is to be able to communicate what our point is, not just throw data at the viewer to try to figure out.

Tell your story.

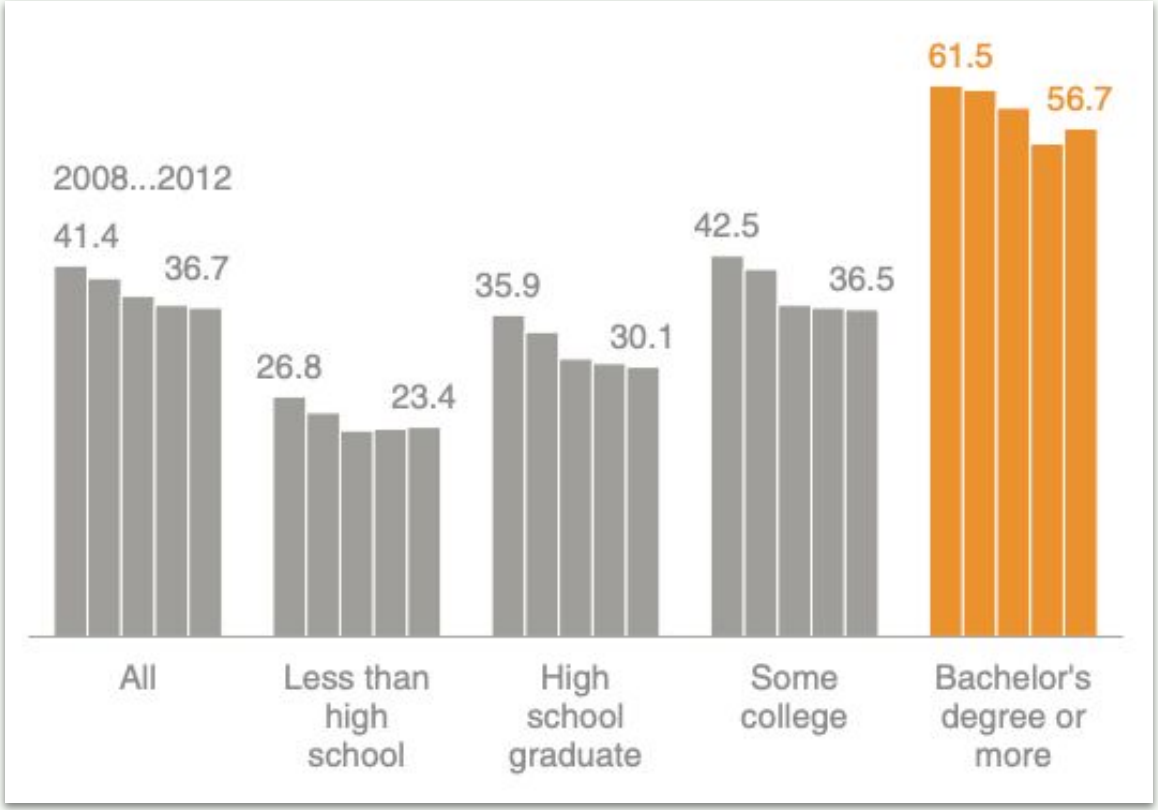


First Version - nice!

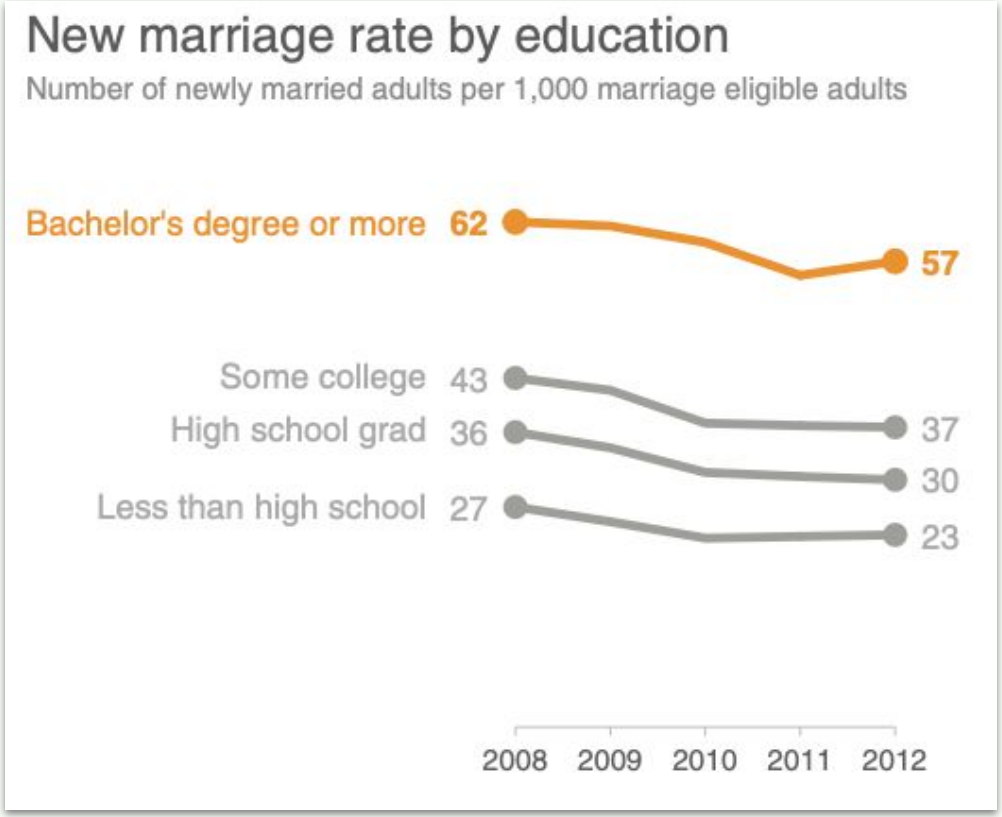


Redirect Attention To Your Story

What you might do at this step greatly depends on the story you wish to tell.



Think Outside The Box



Is this an easier way to make your point?

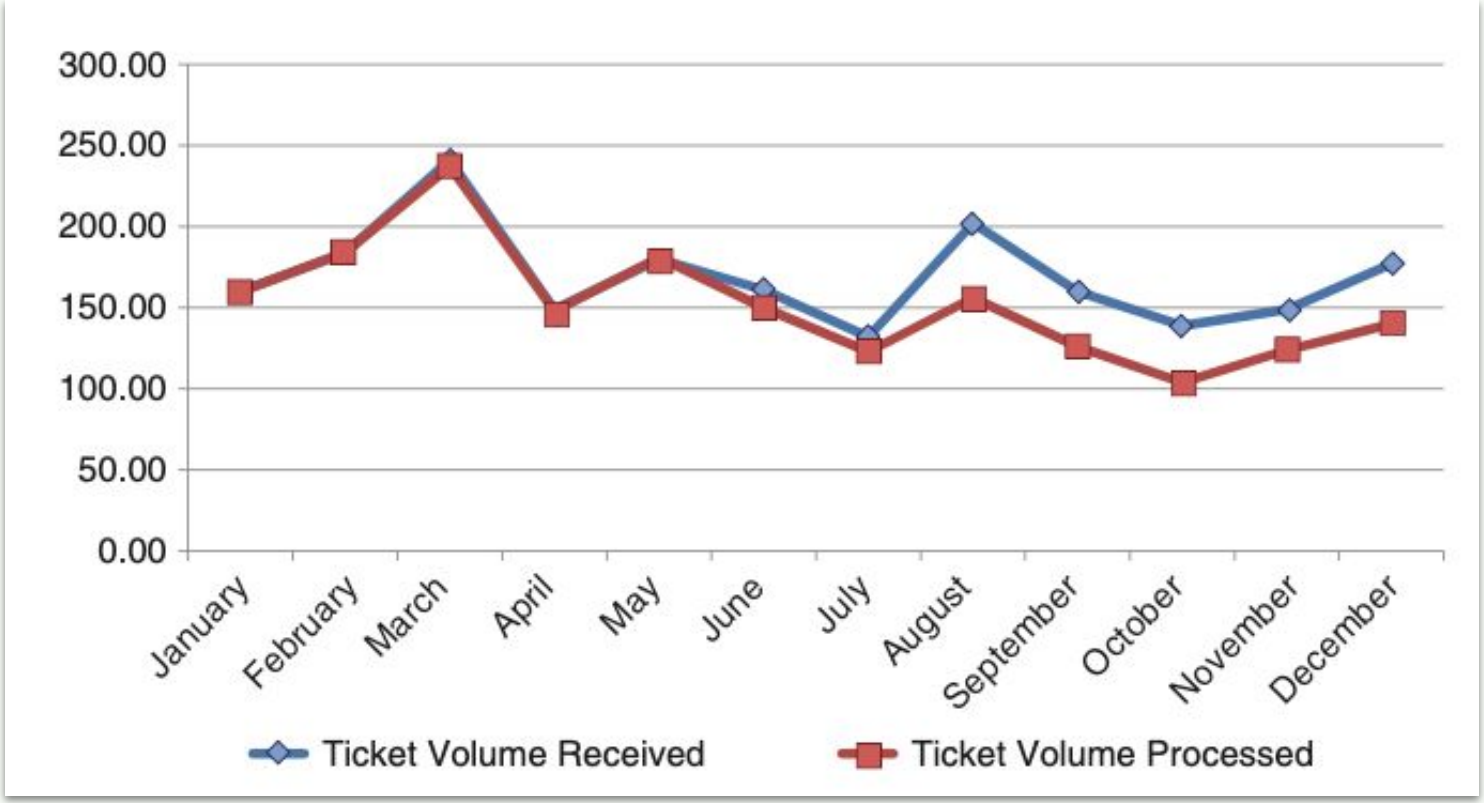
Walkthrough of Plot Design

CMSE 830 Attendance Survey - Lecture 4 9/17/2024

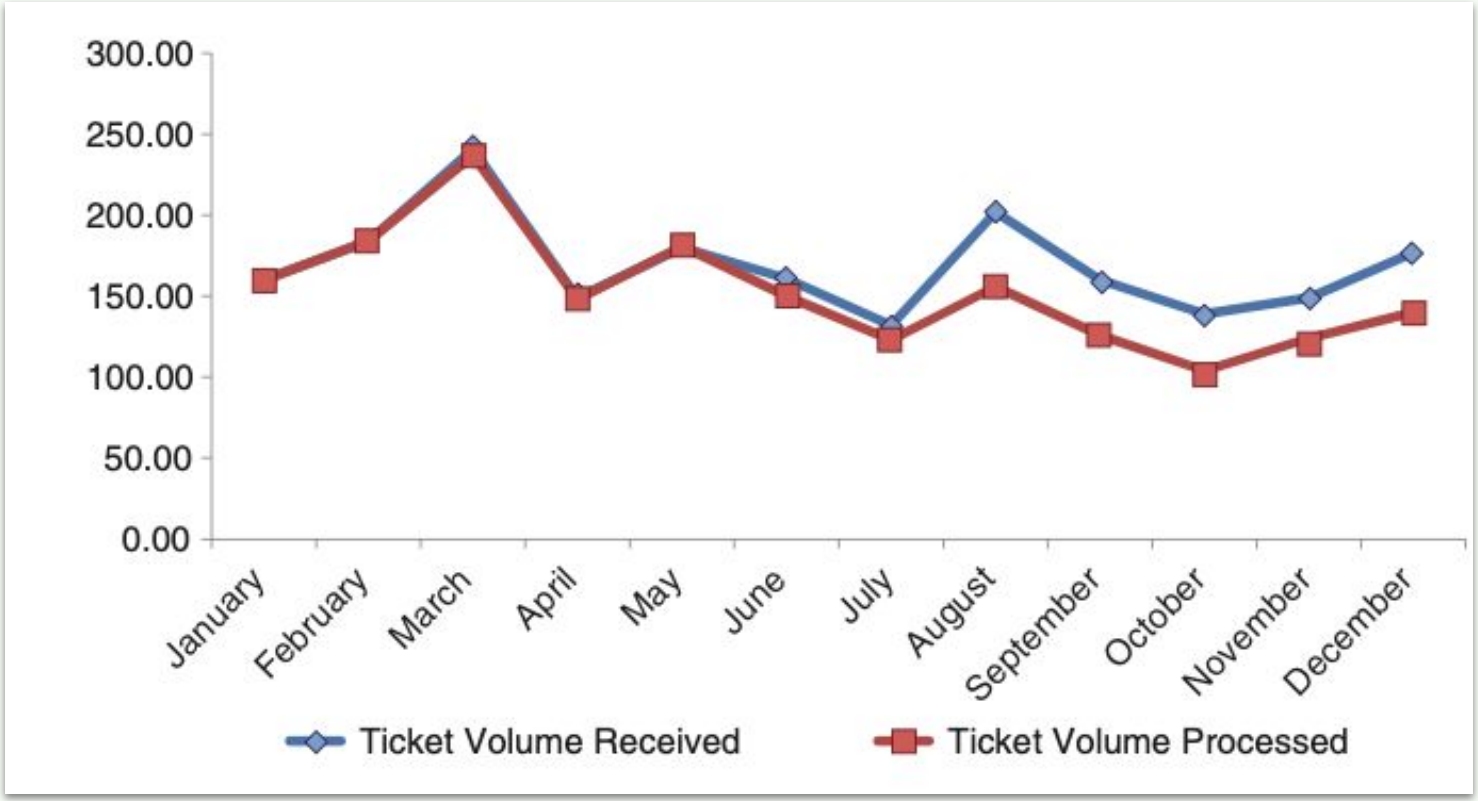
Share responder link

Let's do another one in **nine** steps.

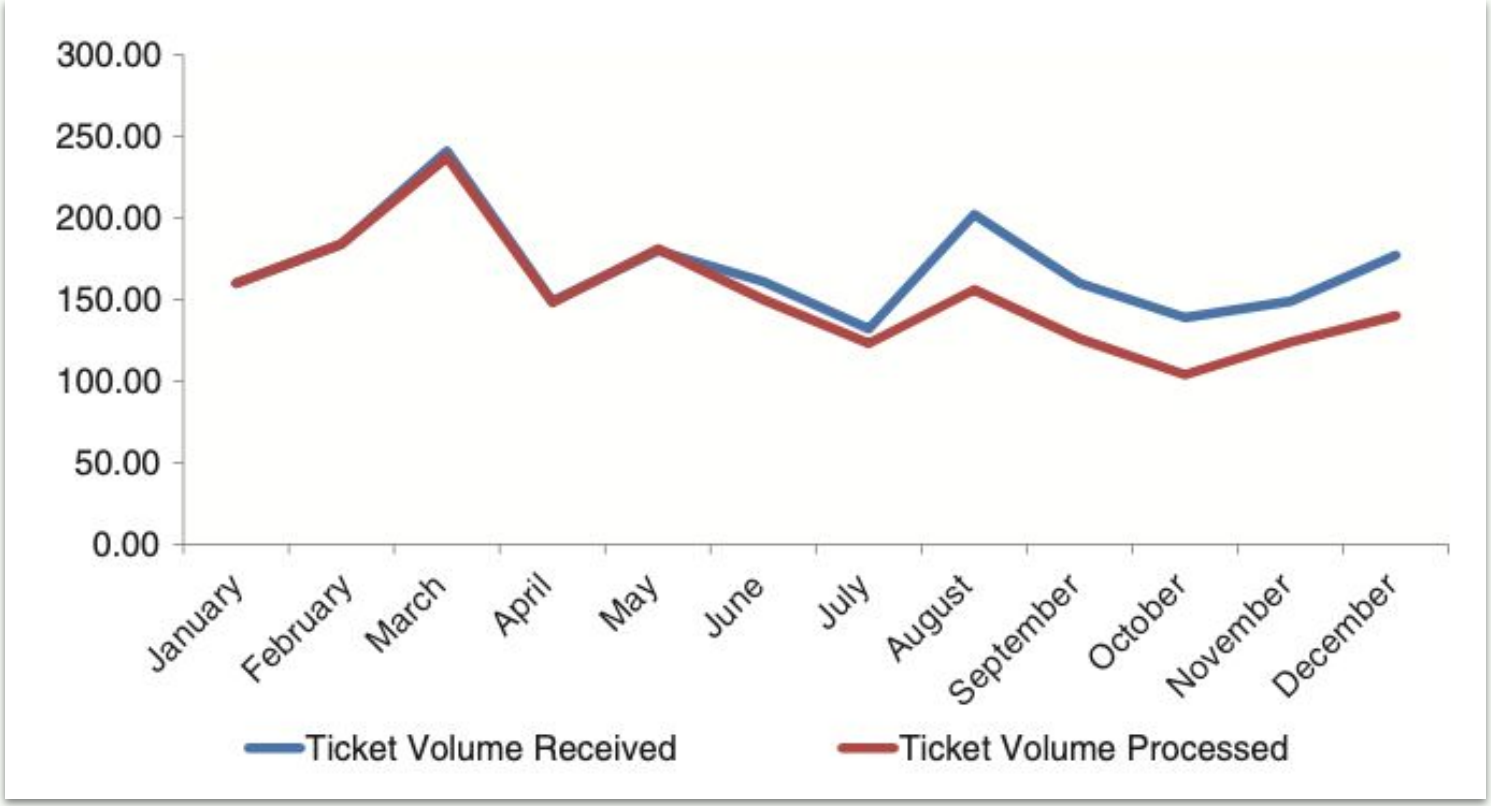
Starting Point (Not Bad!)



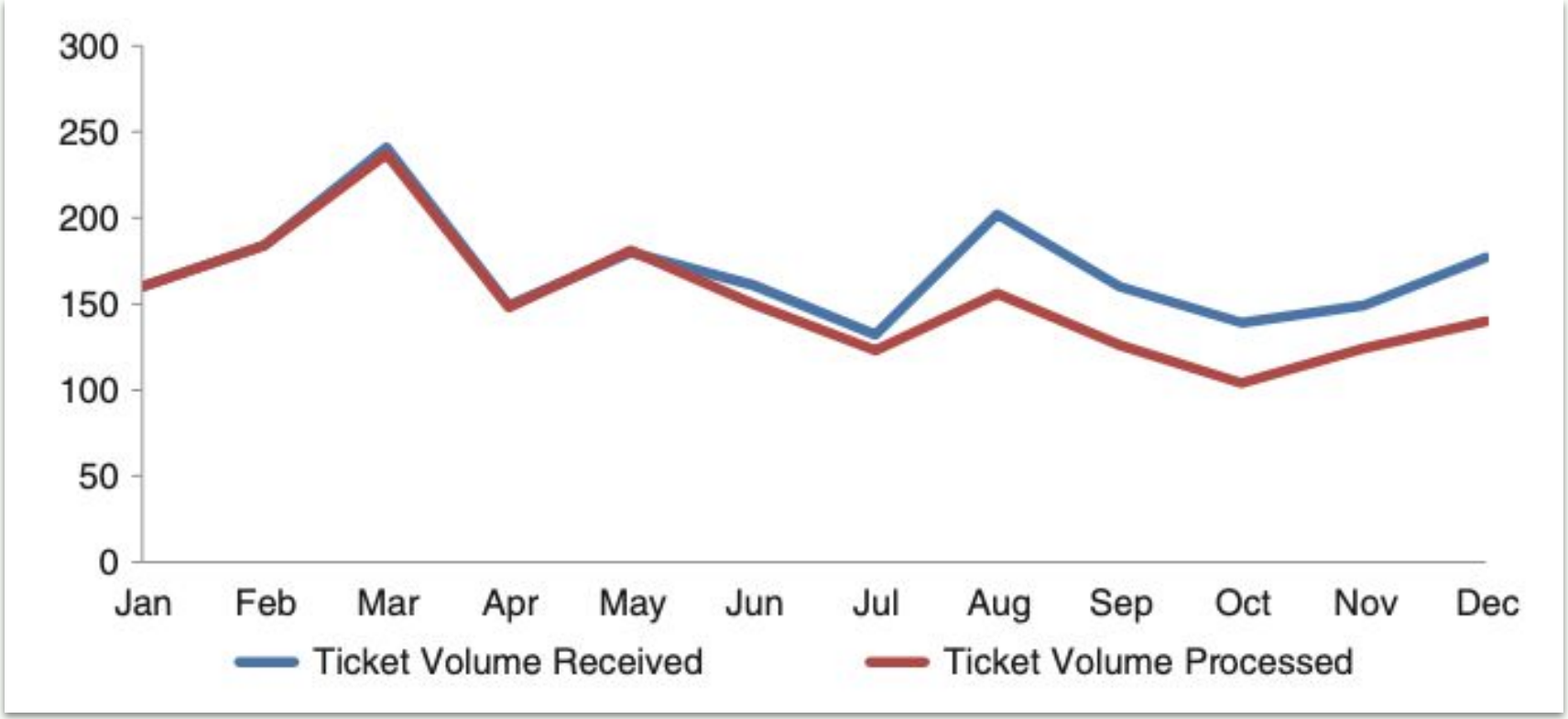
Step 2: Do We Need Grid Lines?



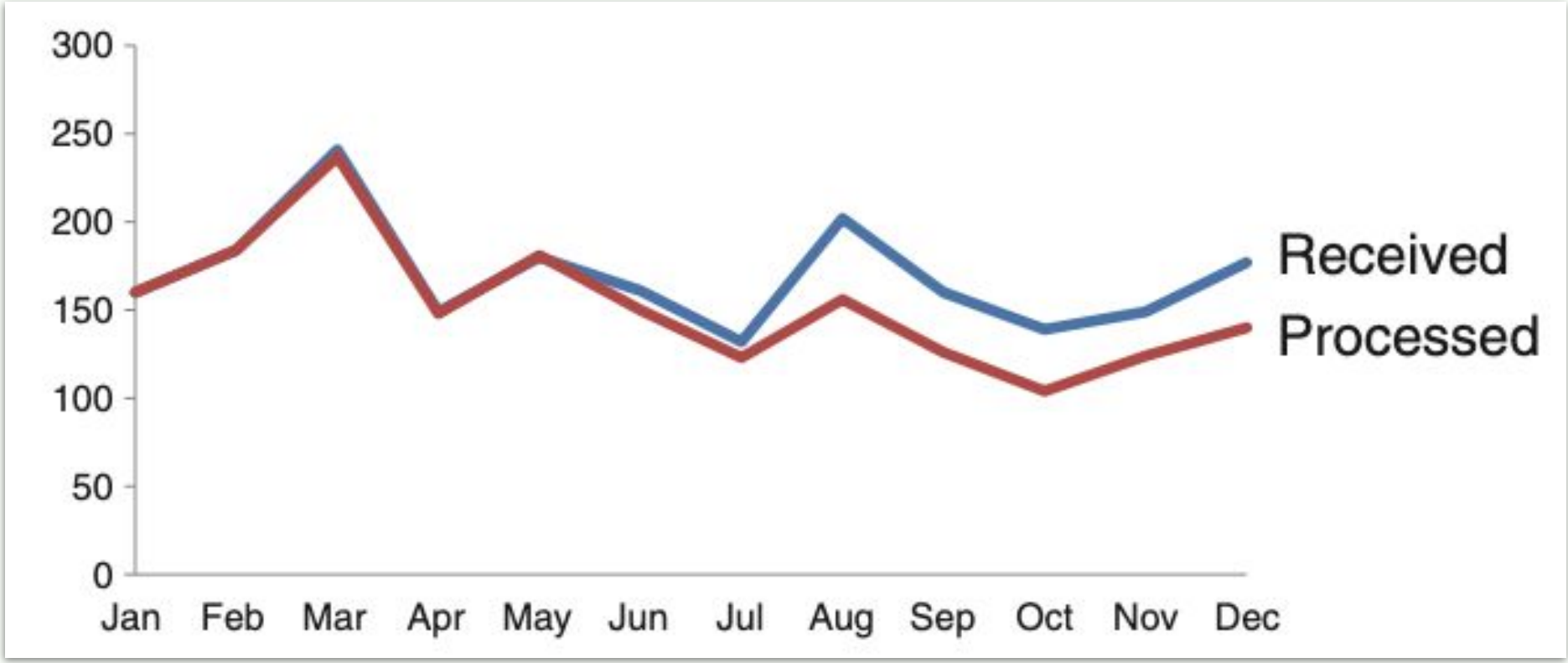
Step 3: Markers Are Redundant?



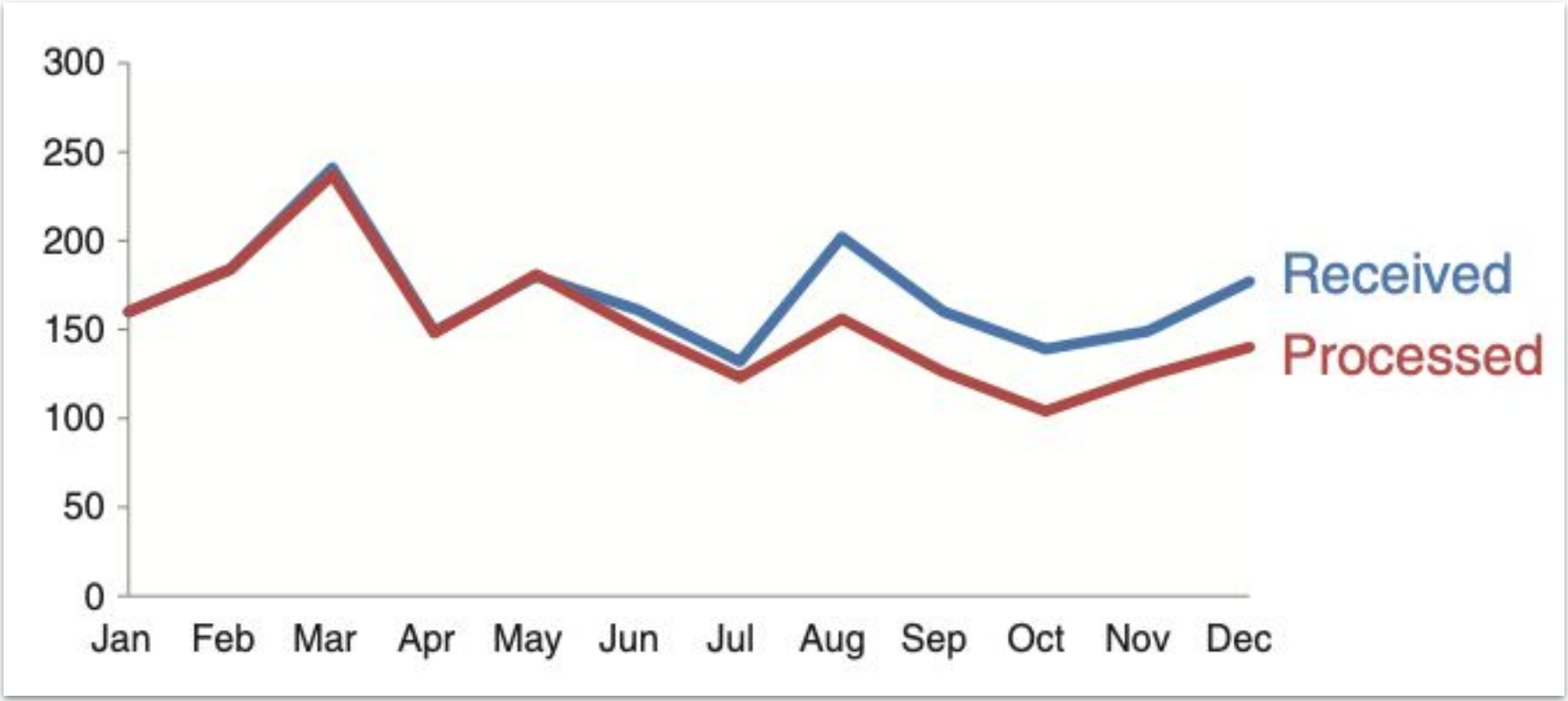
Step 4: Too Much Axis-Label Clutter



Step 5: Move Legend To Data

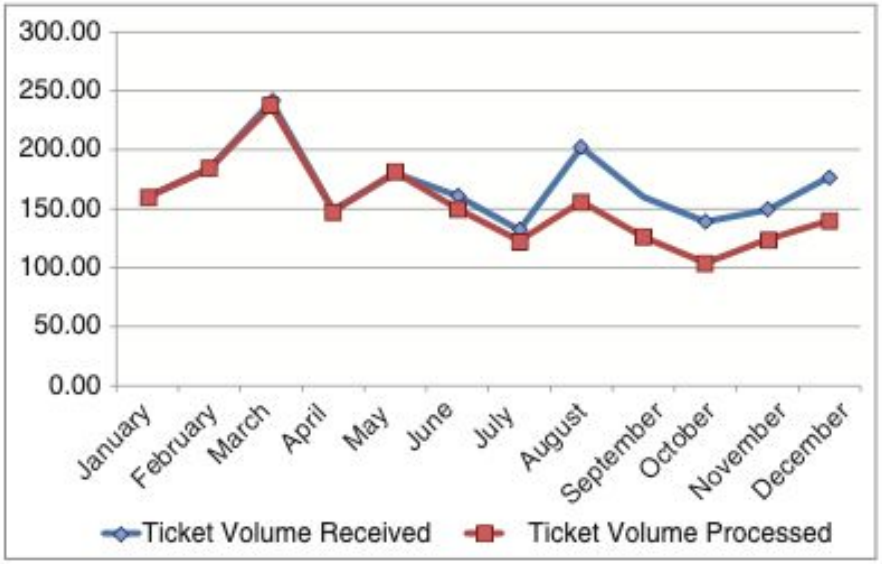


Step 6: Color Code Labels To Data

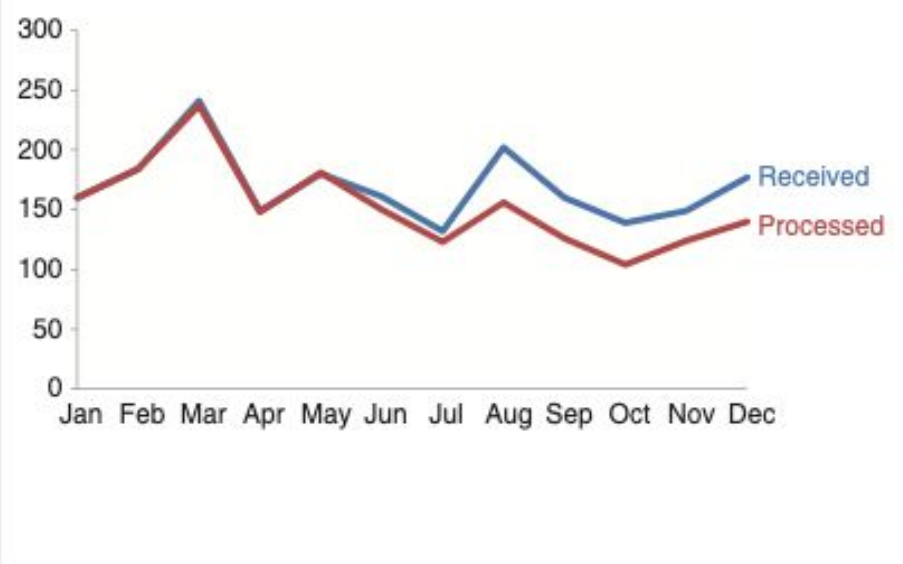


Summary So Far

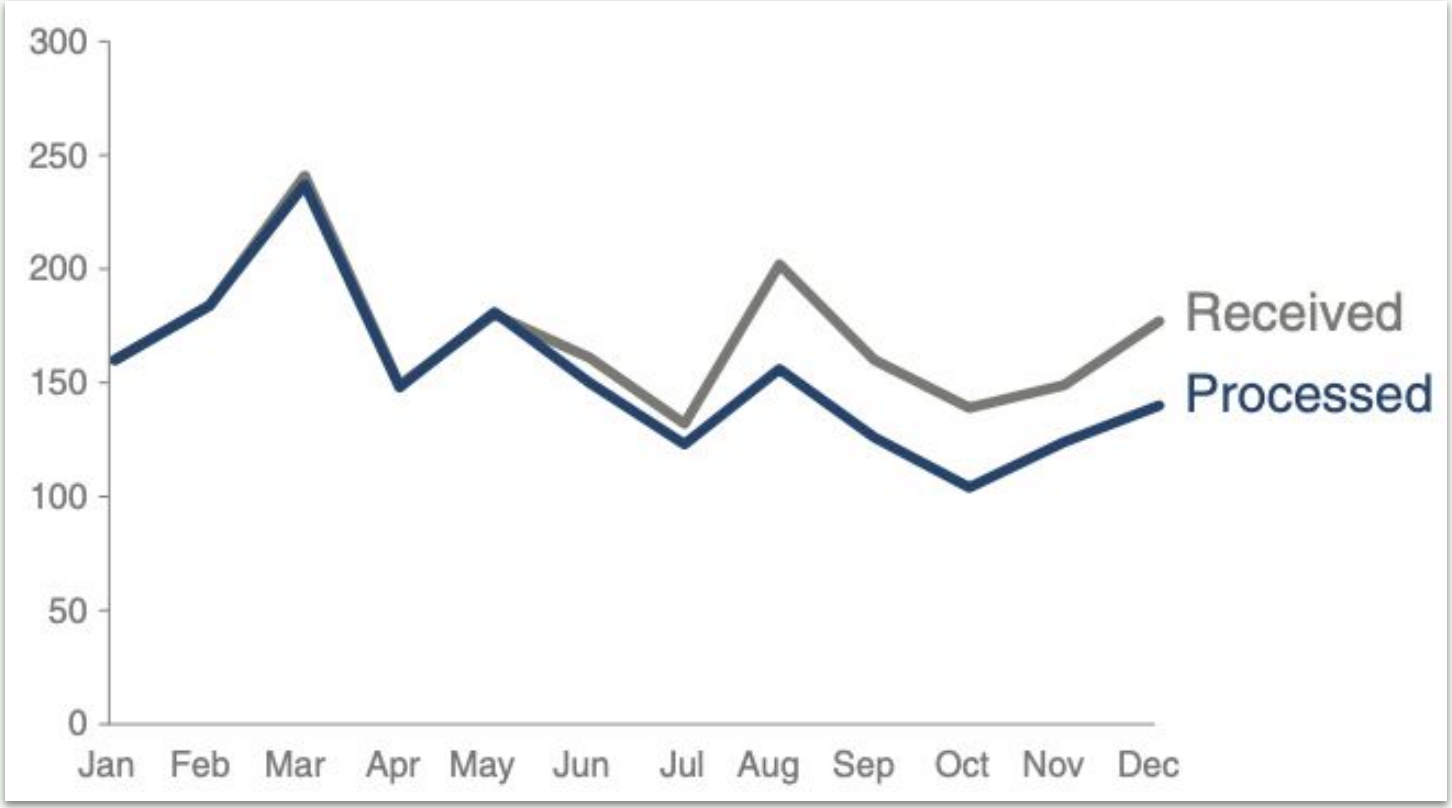
starting point



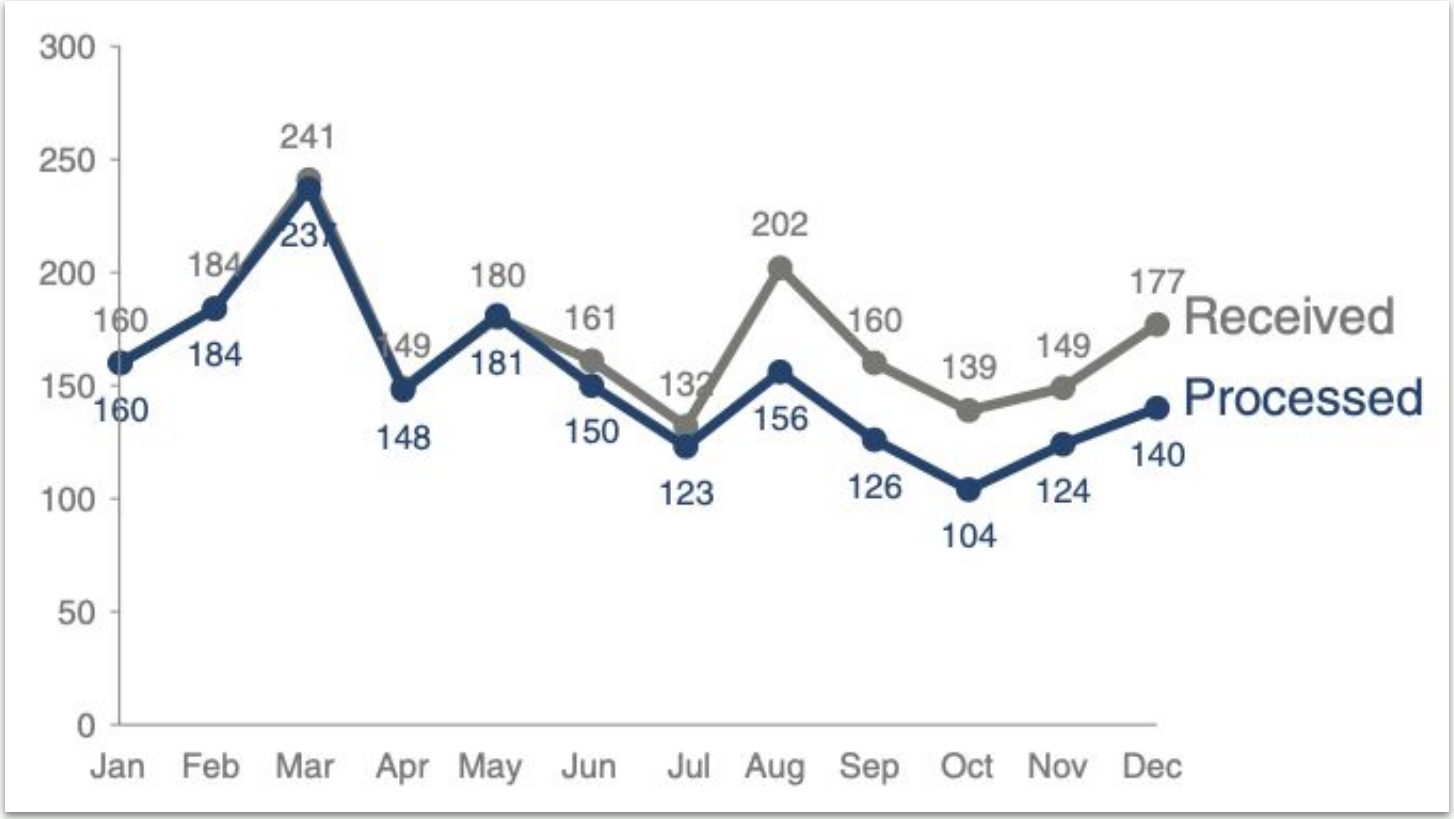
clean, quick to comprehend!



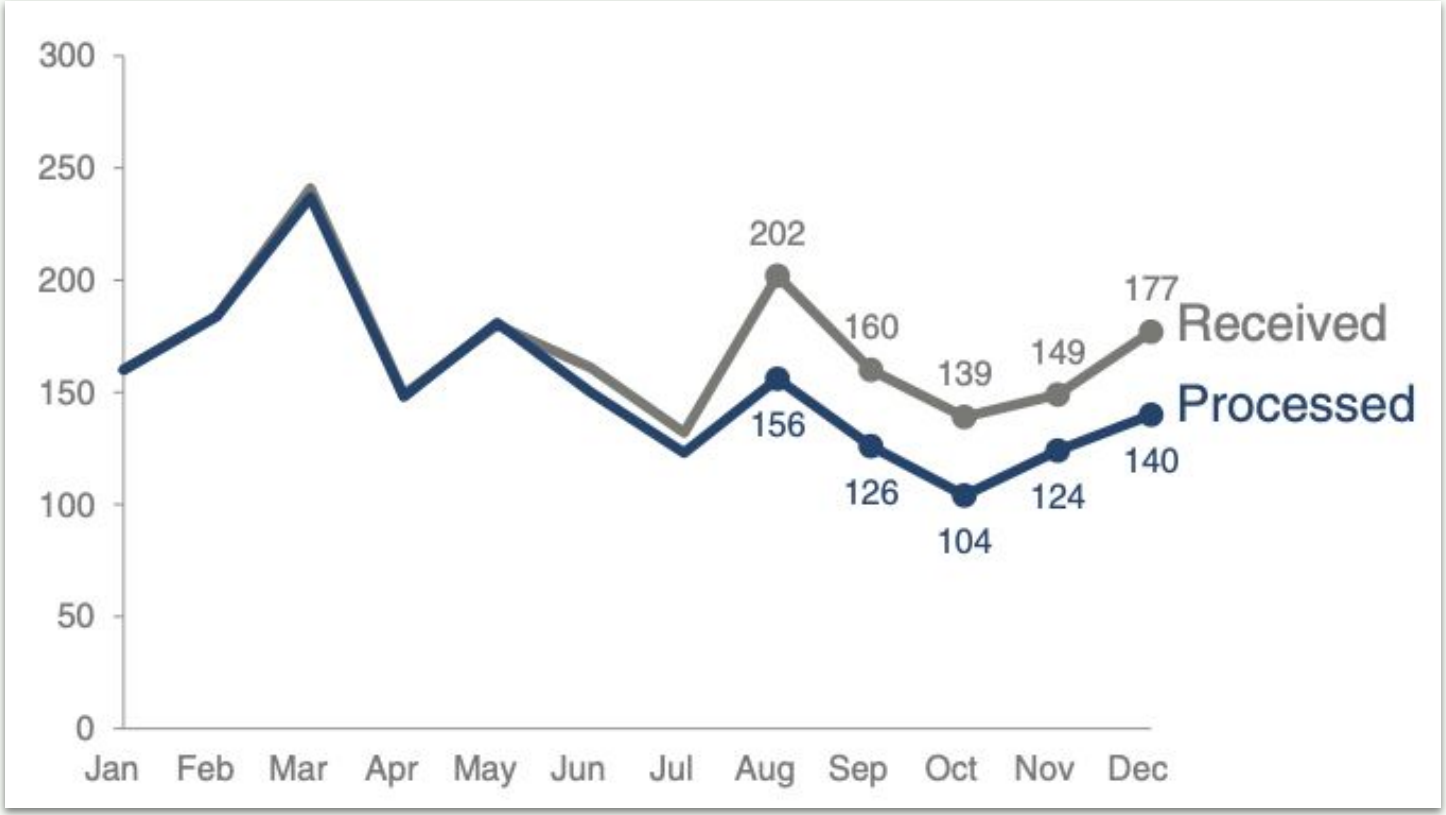
Step 7: Gray-Out Less Important Story



Step 8: Highlight Specific Values

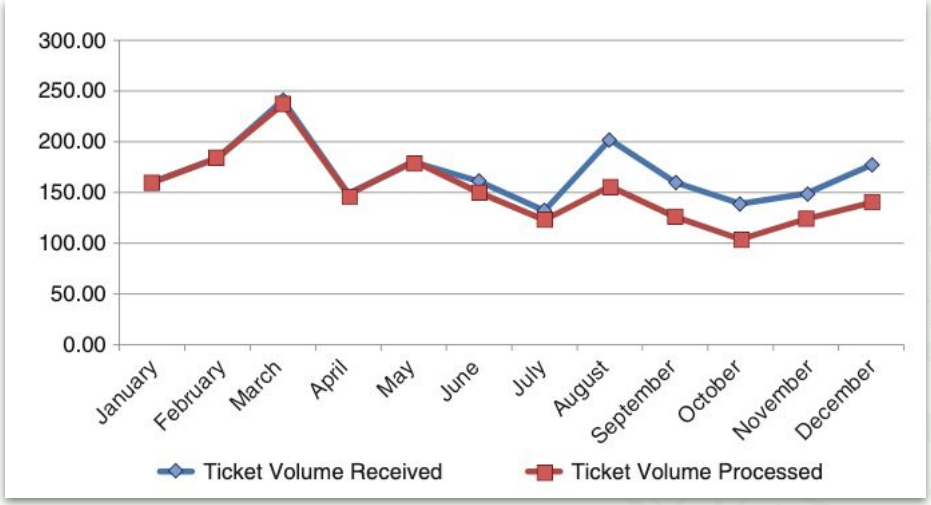


Step 9: But, Only Where The Story Is!

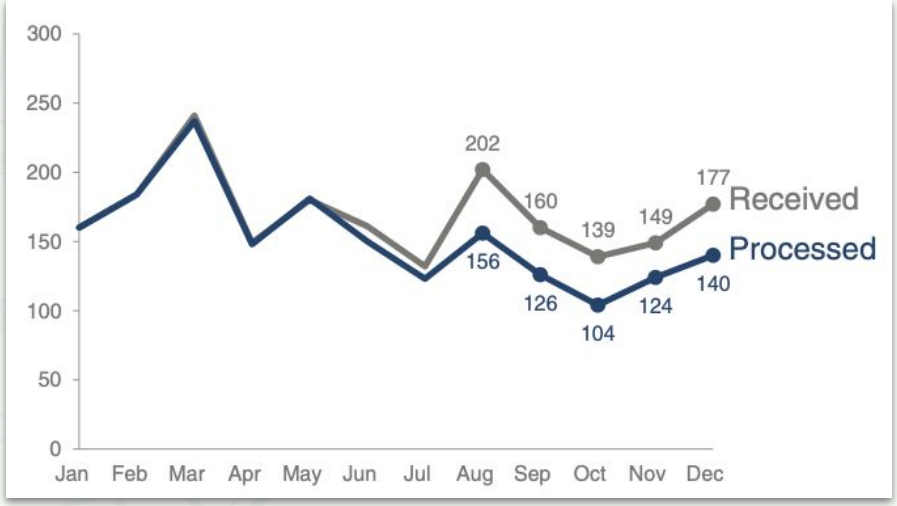


Final Visual!

starting point



A story has emerged...



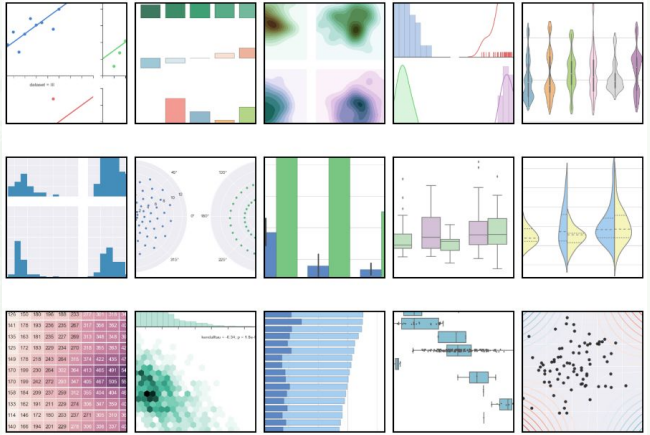
EDA Approaches

Explore the data to allow it to reveal hypotheses to you.

Out[13]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
S.No.	7253.0	NaN	NaN	NaN	3626.0	2093.905084	0.0	1813.0	3626.0	5439.0	7252.0
Name	7253	2041	Mahindra XUV500 W8 2WD	55	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Location	7253	11	Mumbai	949	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Year	7253.0	NaN	NaN	NaN	2013.365366	3.254421	1996.0	2011.0	2014.0	2016.0	2019.0
Kilometers_Driven	7253.0	NaN	NaN	NaN	58699.063146	84427.720583	171.0	34000.0	53416.0	73000.0	6500000.0
Fuel_Type	7253	5	Diesel	3852	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Transmission	7253	2	Manual	5204	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Owner_Type	7253	4	First	5952	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Mileage	7251.0	NaN	NaN	NaN	18.14158	4.562197	0.0	15.17	18.16	21.1	33.54
Engine	7207.0	NaN	NaN	NaN	1616.57347	595.285137	72.0	1198.0	1493.0	1968.0	5998.0
Power	7078.0	NaN	NaN	NaN	112.765214	53.493553	34.2	75.0	94.0	138.1	616.0
Seats	7200.0	NaN	NaN	NaN	5.280417	0.809277	2.0	5.0	5.0	5.0	10.0
New_price	1006.0	NaN	NaN	NaN	22.779692	27.759344	3.91	7.885	11.57	26.0425	375.0
Price	6019.0	NaN	NaN	NaN	9.479468	11.187917	0.44	3.5	5.64	9.95	160.0

numerical/statistical analysis



visualization

We would like to have the visualization aid the statistical analysis.

EDA Using the seaborn Library

1. Why do we need a library like Seaborn?
2. What is the API for Seaborn?
3. What does Seaborn provide to enhance data science?

Michael Waskom

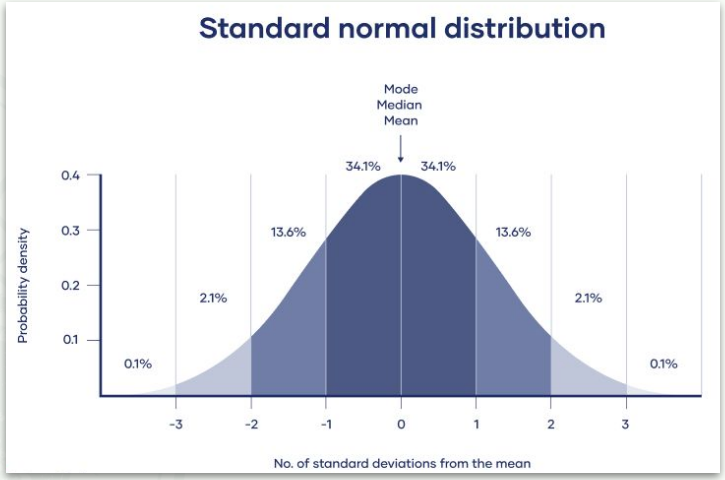
Some Statistics

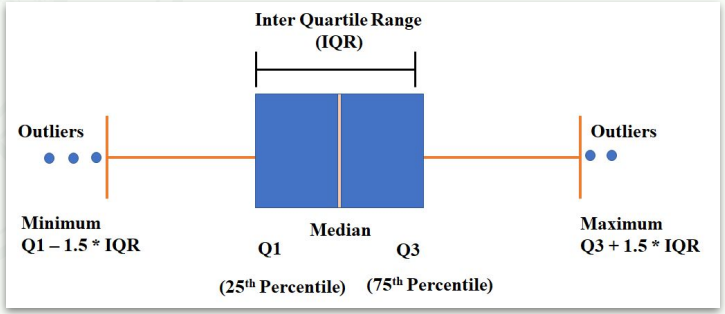
1. distributions
2. correlations

Distributions

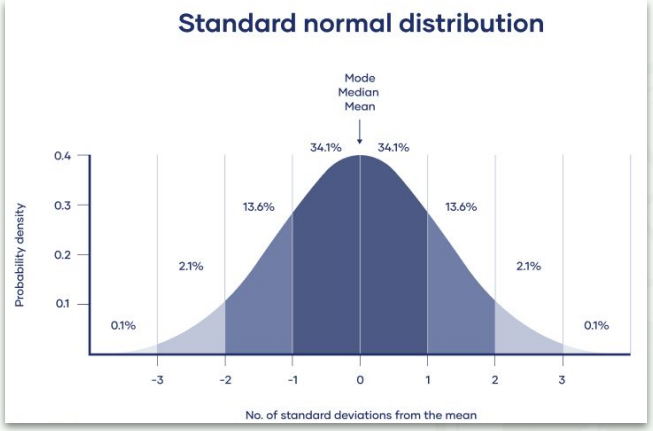
```
1 import seaborn as sns
2 df_iris = sns.load_dataset("iris")
3 df_iris.describe()
```

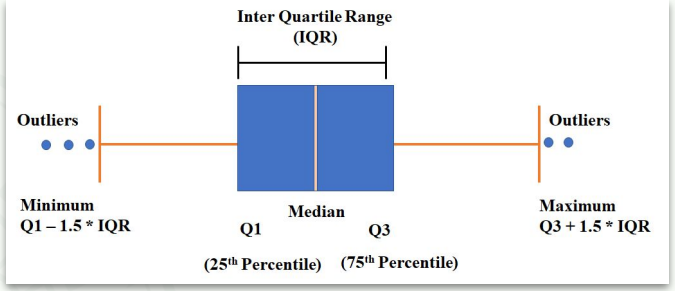
	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000



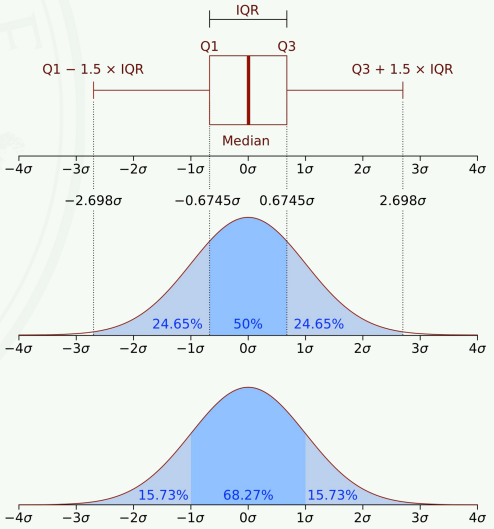


Distributions

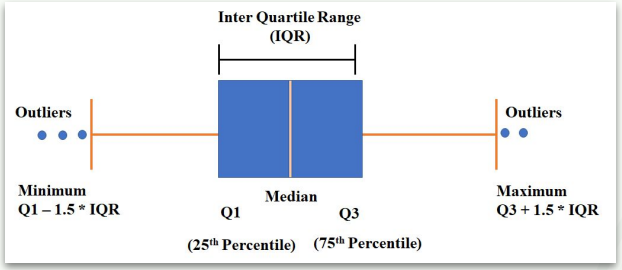




$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$



Violin, Box, Swarm and Raincloud Plots



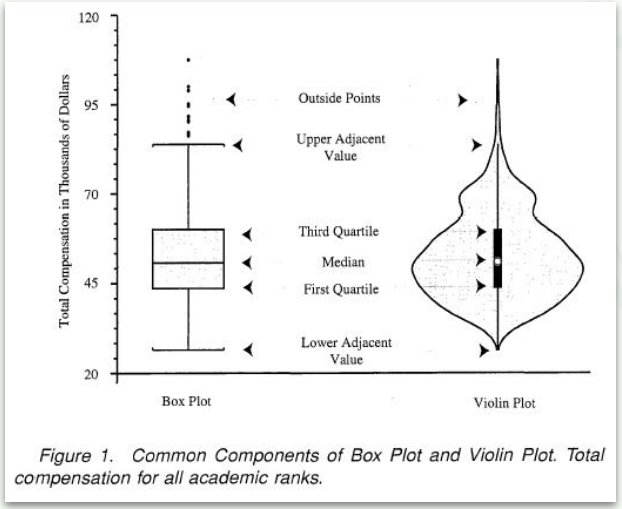
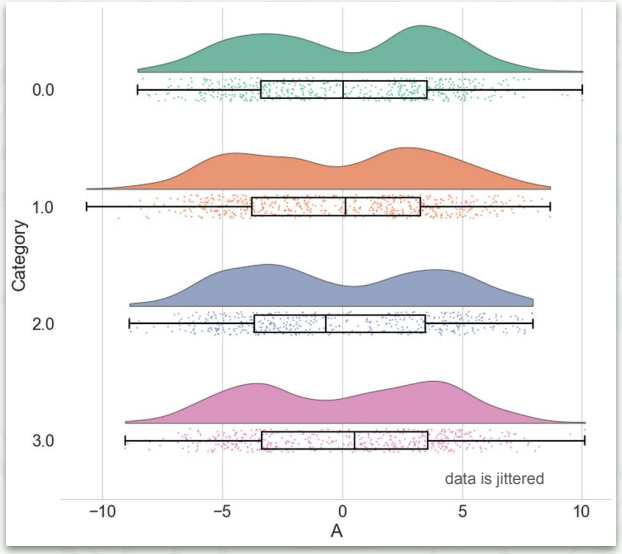
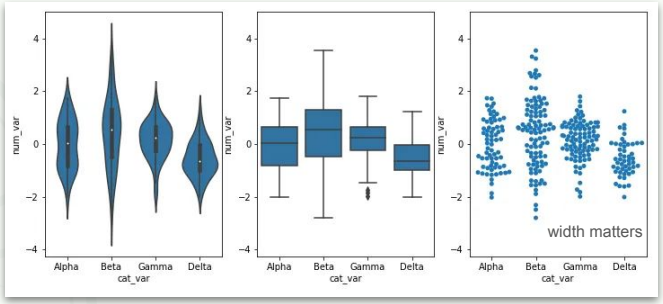


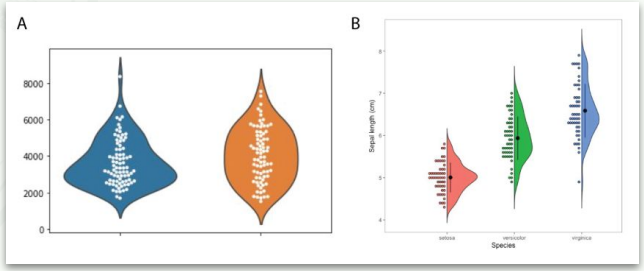
Figure 1. Common Components of Box Plot and Violin Plot. Total compensation for all academic ranks.



Raincloud

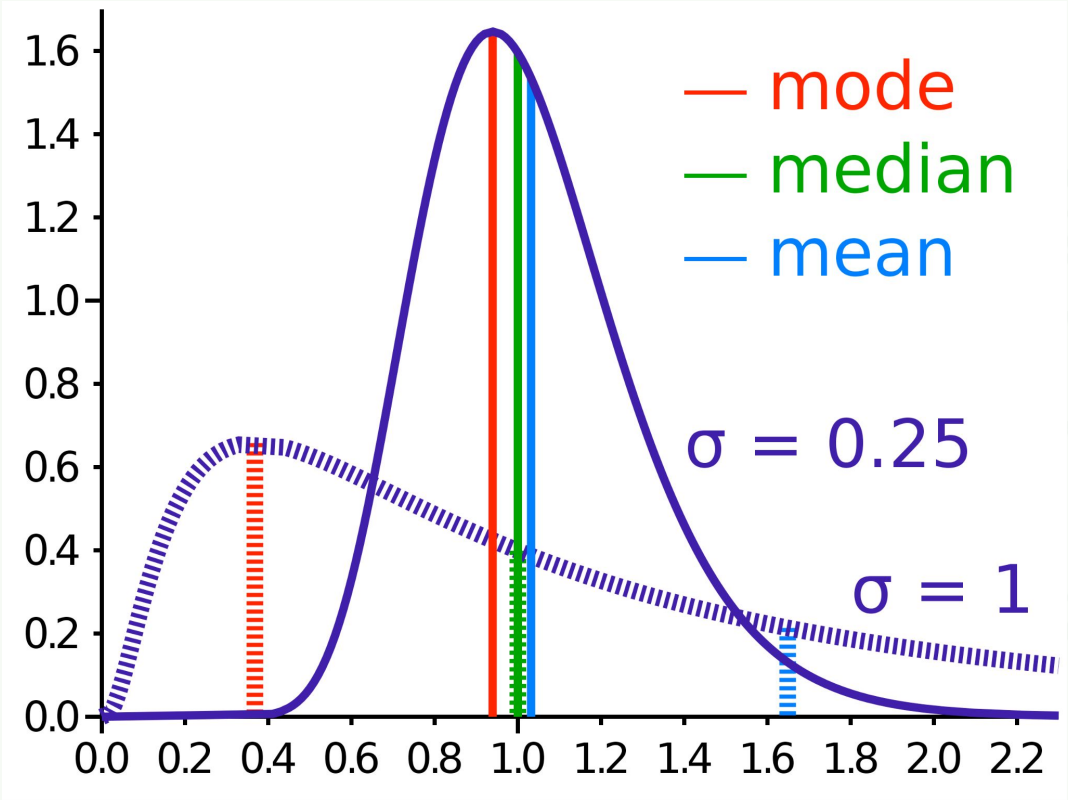
Swarm

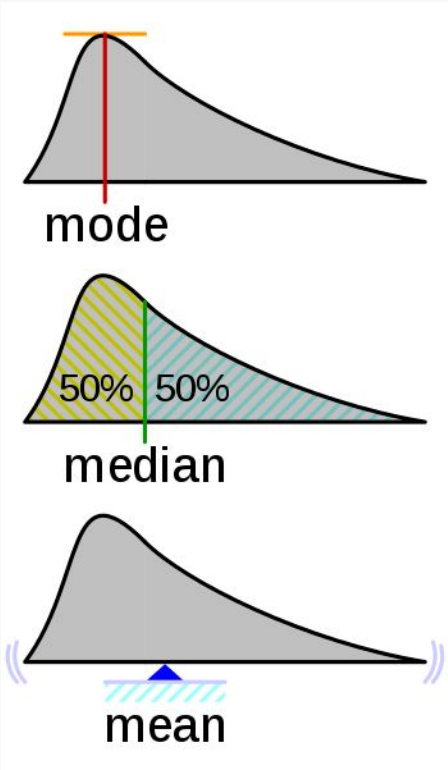




Be creative!

Mode, Median and Mean: Which is Best For Your Work?

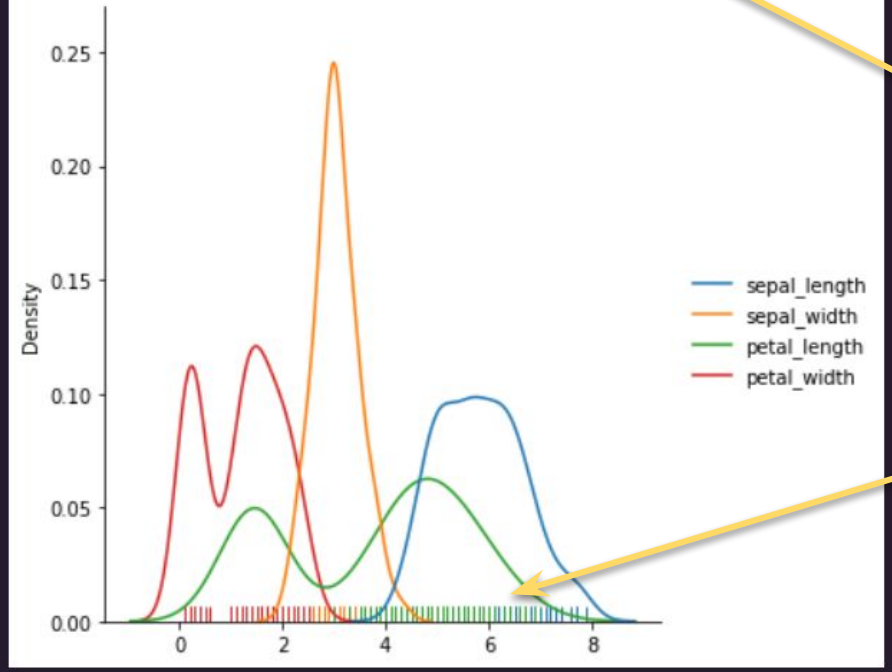




Distributions: Iris Example

```
[9]   1 sns.displot(df_iris, kind="kde", rug=True)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fcd810a7110>
```



kde = kernel density estimation

“rug plot”

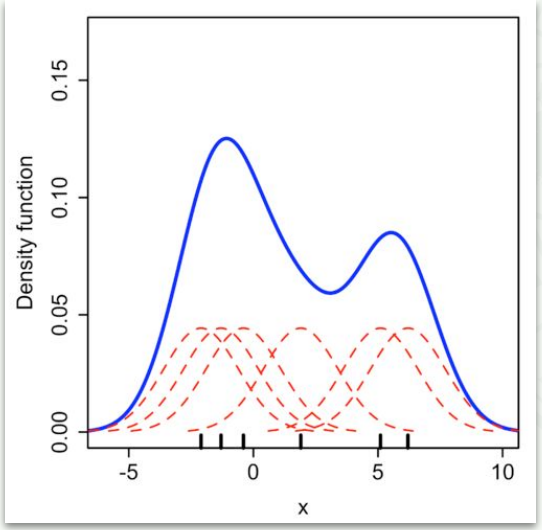
Note that these look nothing like a normal distribution!

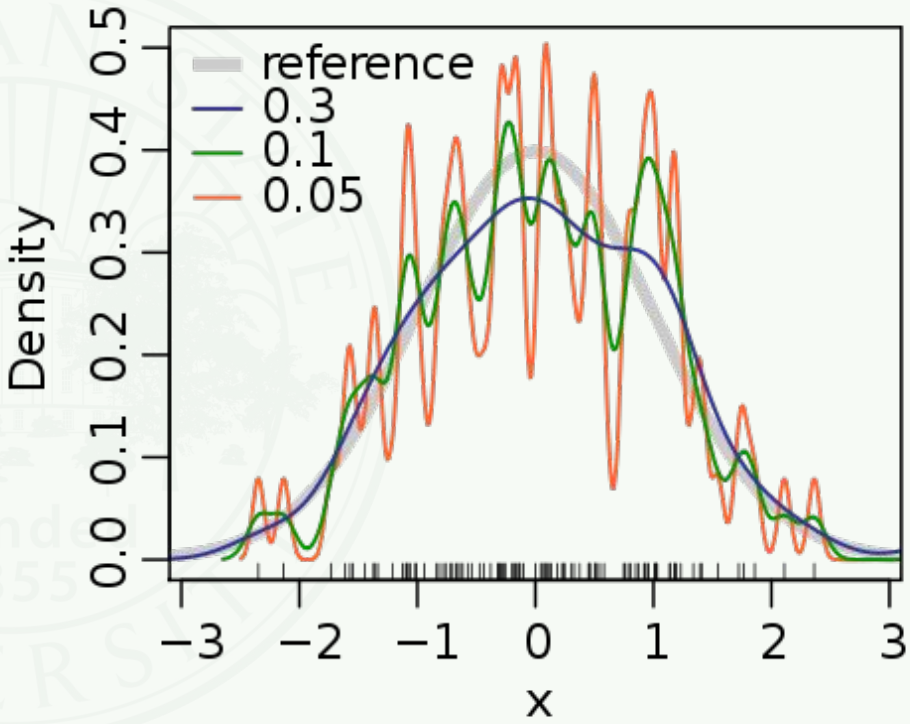
(With the possible exception of sepal width.)

Kernel Density Estimation

K is the "kernel" & w is the "bandwidth"

$$p(x) = \frac{1}{Nw} \sum_{i=1}^{N} K\left(\frac{x - x_i}{w}\right)$$

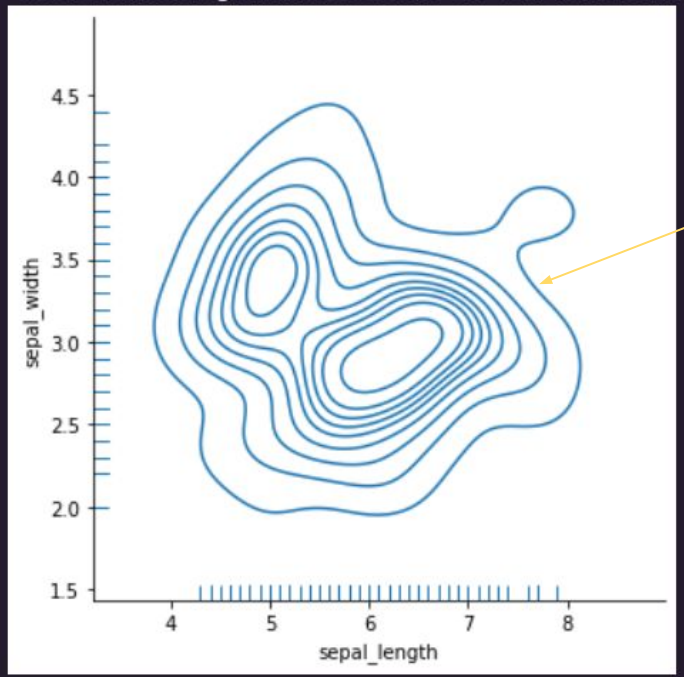




Distributions: Iris Example in 2D

```
[18]  1 sns.displot(df_iris, x = "sepal_length", y="sepal_width", kind="kde", rug=True)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fcd809b1d50>
```



contours

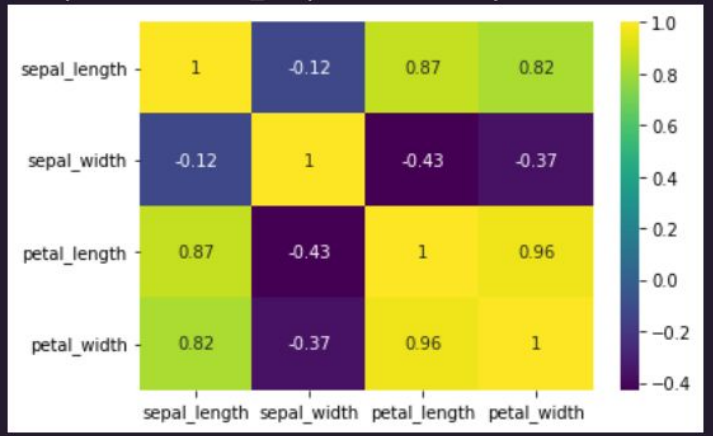
Correlations

```
[10]  1 df_iris.corr()
```

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.117570	0.871754	0.817941
sepal_width	-0.117570	1.000000		
petal_length	0.871754	-0.428440		
petal_width	0.817941	-0.366126		

```
[15]  1 sns.heatmap(df_iris.corr(), annot=True, cmap="viridis")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcd80b4db10>
```



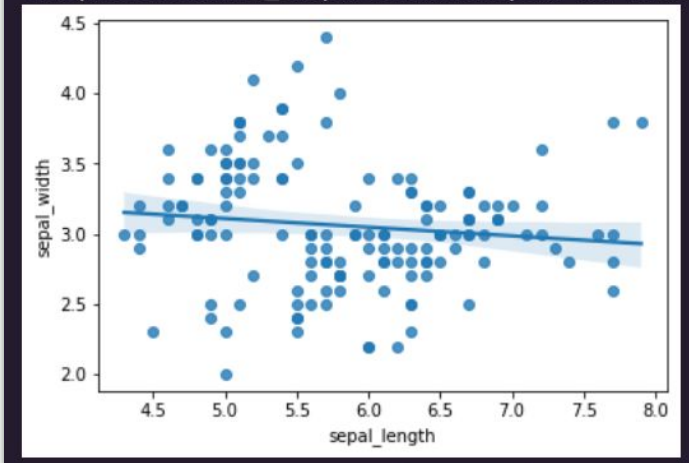
$$\rho_{X,Y} = \frac{\mathbb{E}[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

Pearson correlation coefficient

Correlations and Distributions

```
1 sns.regplot(data=df_iris,x = "sepal_length", y="sepal_width")
```

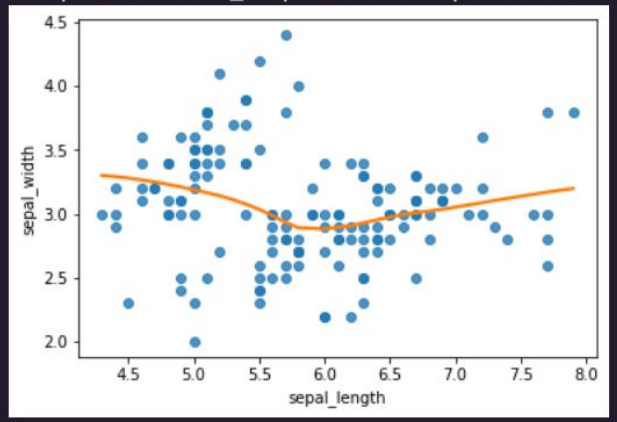
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcd7edc3c50>
```



Finding trends in distributions is very tricky!

```
[24] 1 sns.regplot(data=df_iris, lowess=True, line_kws={"color": "C1"},
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcd7ed17790>
```



A Bit About Matplotlib

Matplotlib versus Pyplot

Pyplot: high-level interface

```
import matplotlib.pyplot as plt
```

MATLAB®

John Hunter
1968-2012

MPL: low-level interface

```
import matplotlib as mpl
```

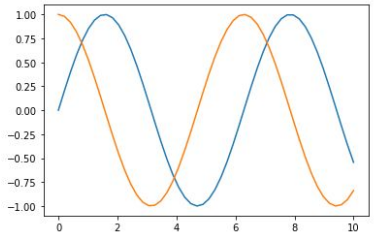
Deferred Rendering (Example 1)

jupyter MyFirstNotebook Last Checkpoint: 04/14/2021 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
In [1]: %matplotlib inline
        from matplotlib import pyplot as plt
        import numpy as np

        x = np.linspace(0,10,50)

        plt.plot(x, np.sin(x))
        plt.plot(x, np.cos(x))

        plt.show()
```




```
In [ ]:
```

A Bit About Matplotlib

Matplotlib versus Pyplot

Pyplot: high-level interface

```
import matplotlib.pyplot as plt
```

MATLAB®

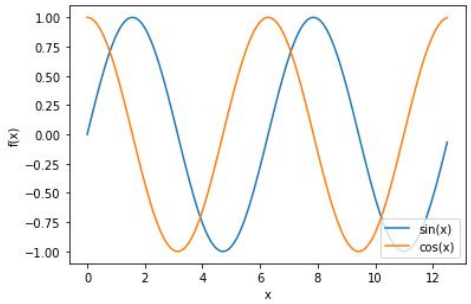
↑

MPL: low-level interface

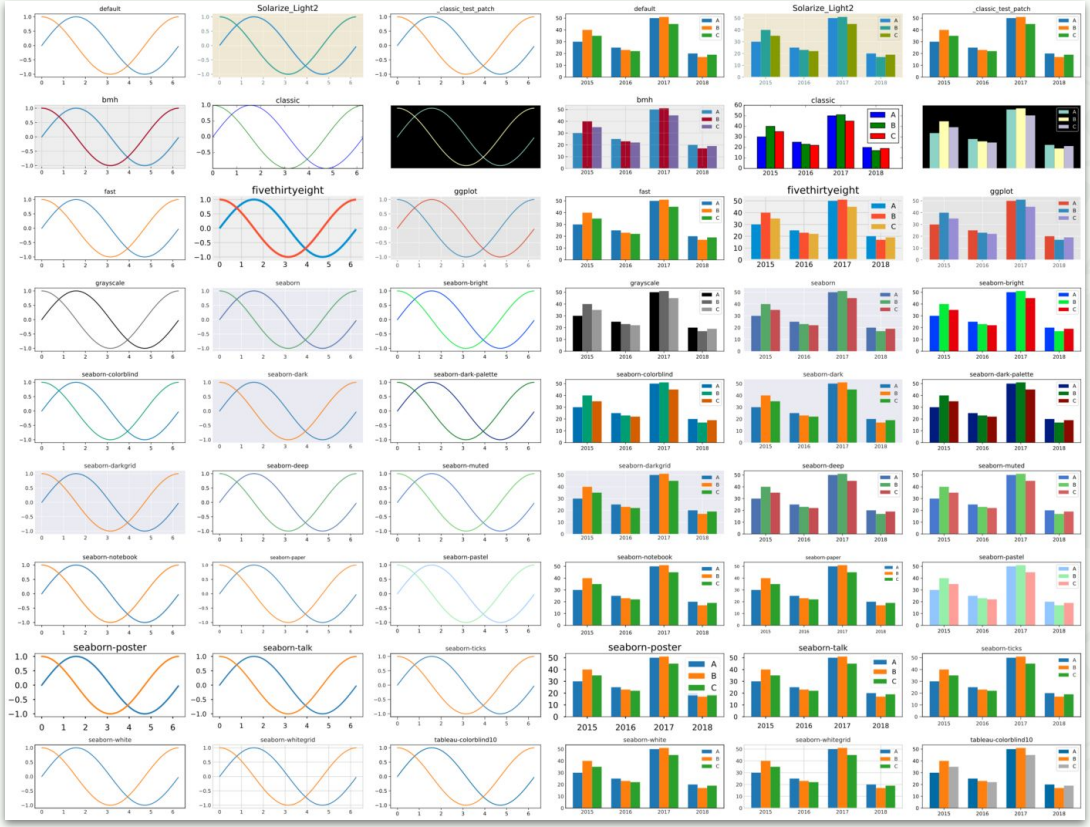
```
import matplotlib as mpl
```

Deferred Rendering (Example 2)

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt

        x = np.arange(0, np.pi*4, 0.1)
        y1 = np.sin(x)
        y2 = np.cos(x)

        plt.plot(x,y1, label='sin(x)')
        plt.plot(x,y2, label='cos(x)')

        plt.xlabel('x')
        plt.ylabel('f(x)')

        plt.legend(loc='lower right')

Out[1]: <matplotlib.legend.Legend at 0x23c8b058b08>
```



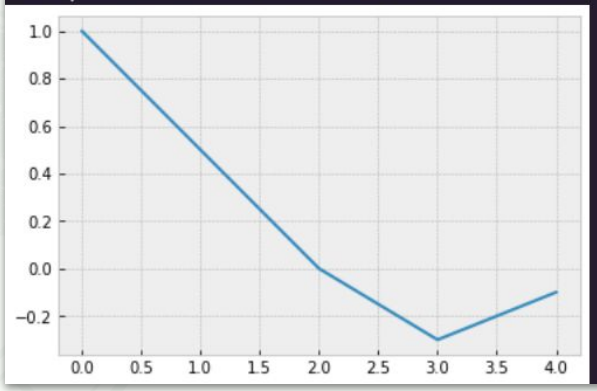
Styles



`plt.style.use('bmh')`

```
1 import matplotlib.pyplot as plt
2 plt.style.use('bmh')
3 plt.plot([0,1,2,3,4], [1, 0.5, 0, -0.3, -0.1])
```

```
[<matplotlib.lines.Line2D at 0x7f8ec8fa9690>]
```

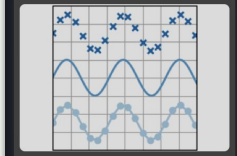


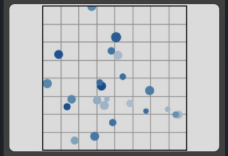
If you are interested in creating your own style, let me know!

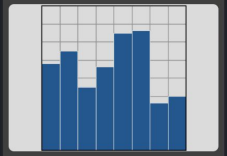
Plot Types (https://matplotlib.org/stable/plot_types/index.html)

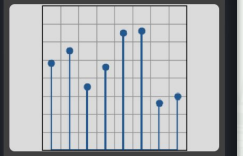
Pairwise data

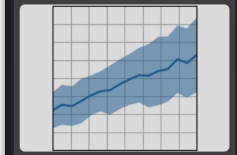
Plots of pairwise (x, y), tabular $(var\_0, \cdots, var\_n)$, and functional $f(x) = y$ data.

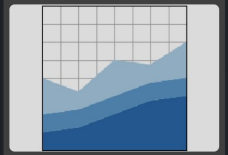

plot(x, y)

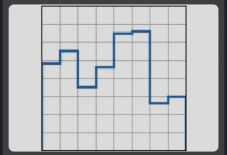

scatter(x, y)


bar(x, height)


stem(x, y)

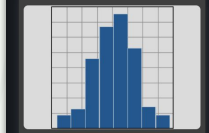

fill_between(x, y1, y2)

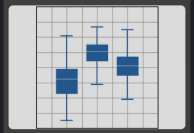

stackplot(x, y)

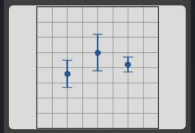

stairs(values)

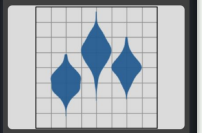
Statistical distributions

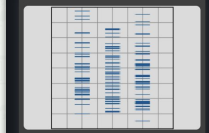
Plots of the distribution of at least one variable in a dataset. Some of these methods also compute the distributions.


hist(x)

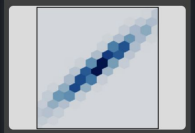

boxplot(X)


errorbar(x, y, yerr, xerr)

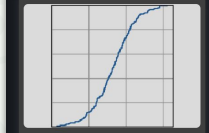

violinplot(D)


eventplot(D)


hist2d(x, y)


hexbin(x, y, C)

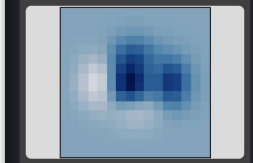

pie(x)

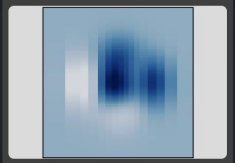

ecdf(x)

Plot Types (https://matplotlib.org/stable/plot_types/index.html)

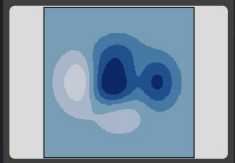
Gridded data

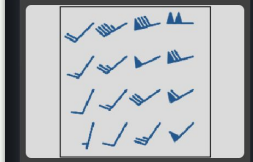
Plots of arrays and images $Z_{i,j}$ and fields $U_{i,j}$, $V_{i,j}$ on regular grids and corresponding coordinate grids $X_{i,j}$, $Y_{i,j}$.

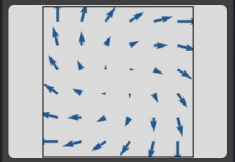

imshow(Z)


pcolormesh(X, Y, Z)


contour(X, Y, Z)


contourf(X, Y, Z)

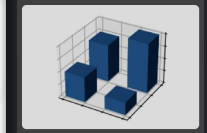

barbs(X, Y, U, V)

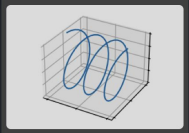

quiver(X, Y, U, V)

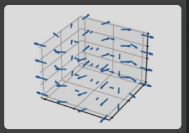

streamplot(X, Y, U, V)

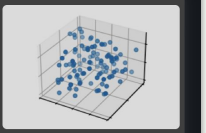
3D and volumetric data

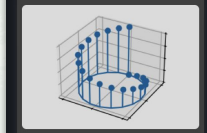
Plots of three-dimensional (x, y, z), surface $f(x, y) = z$, and volumetric $V_{x,y,z}$ data using the `mpl_toolkits.mplot3d` library.

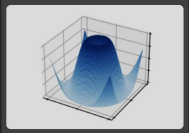

bar3d(x, y, z, dx, dy, dz)

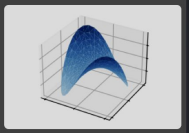

plot(xs, ys, zs)

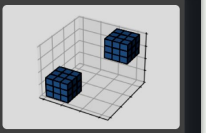

quiver(X, Y, Z, U, V, W)

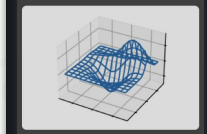

scatter(xs, ys, zs)


stem(x, y, z)


plot_surface(X, Y, Z)

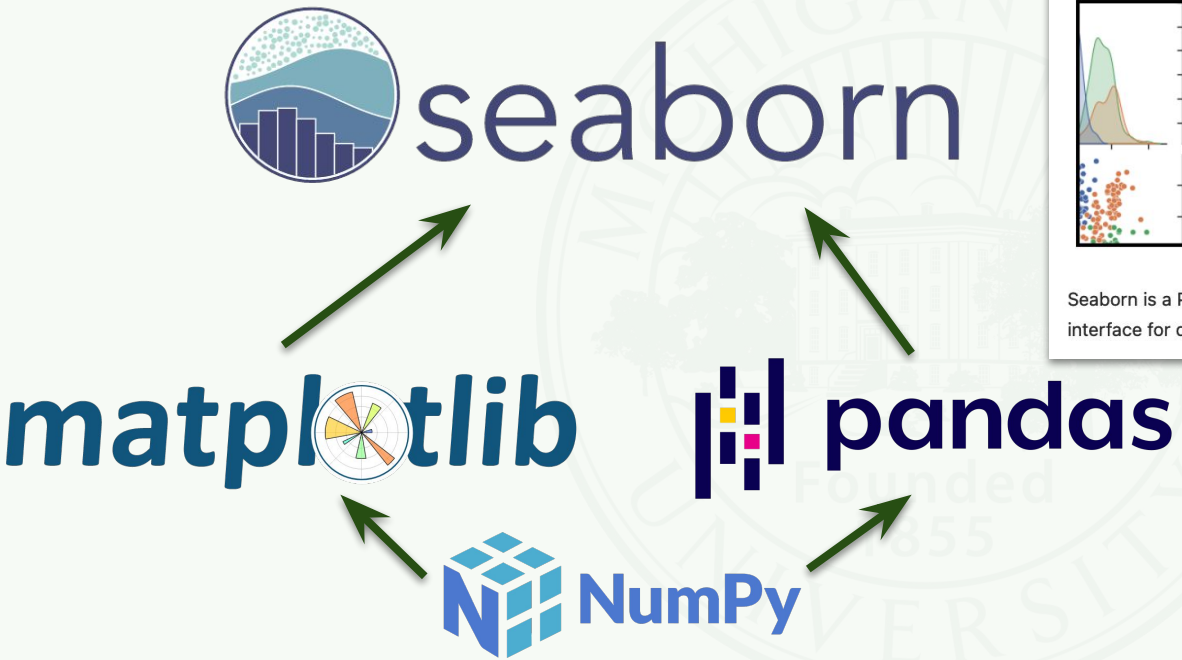

plot_trisurf(x, y, z)

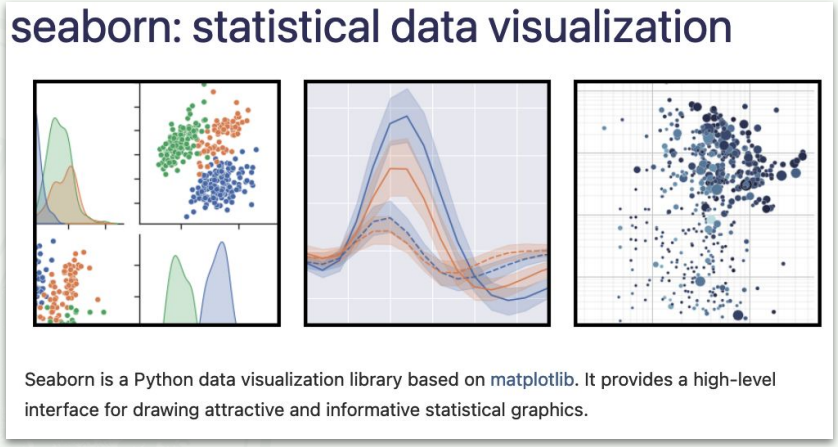

voxels([x, y, z], filled)


plot_wireframe(X, Y, Z)

What Seaborn Is and Is Not

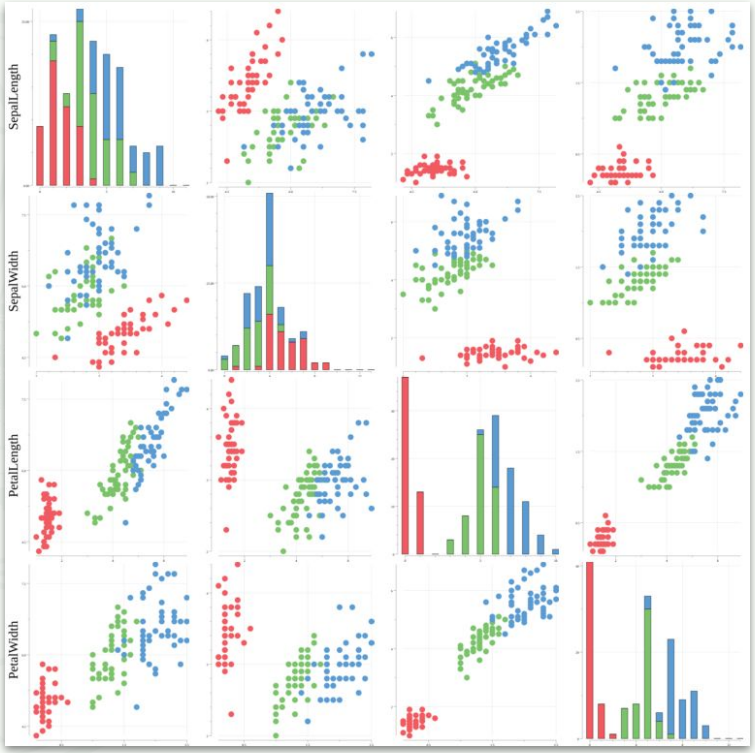
Seaborn is **not** a replacement for matplotlib!



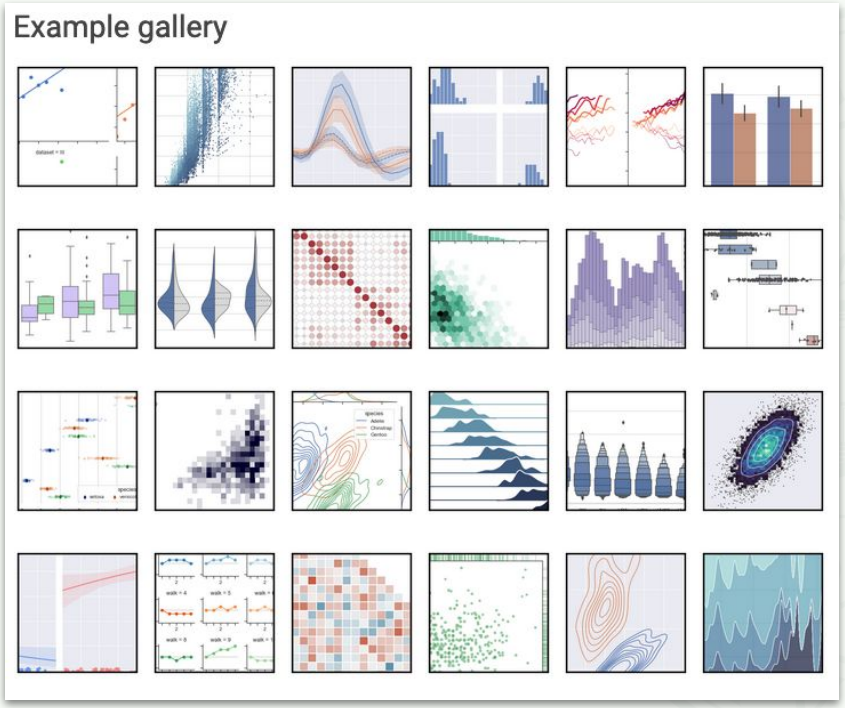


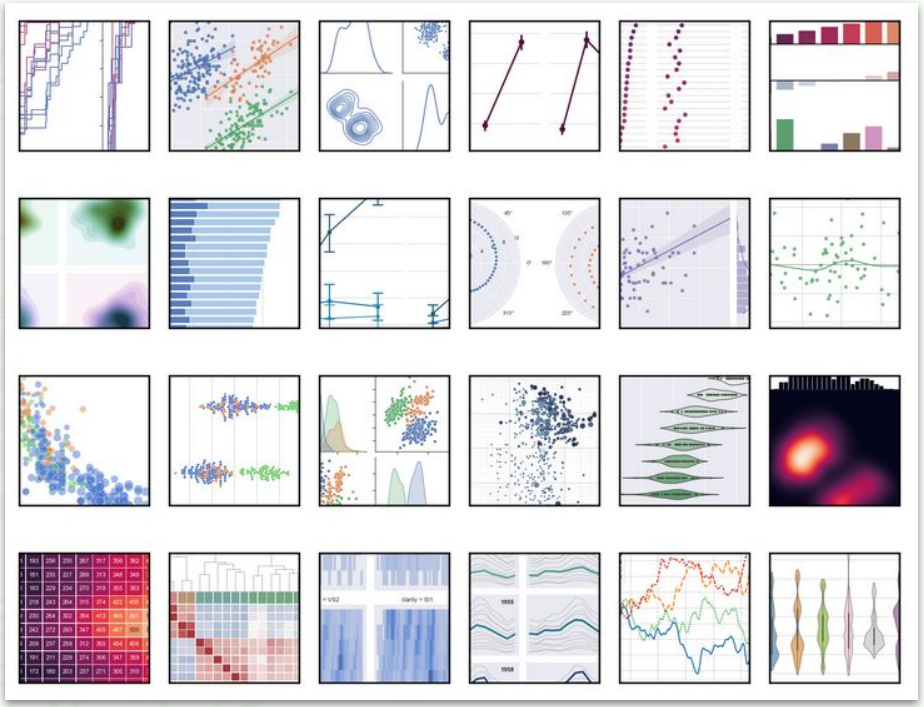
- data visualization
- statistical
- based on matplotlib

Seaborn Styles (`sns.set_style("whitegrid")`)

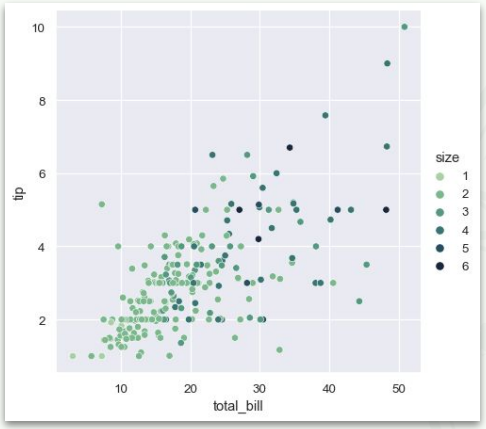


Visit Seaborn's Gallery

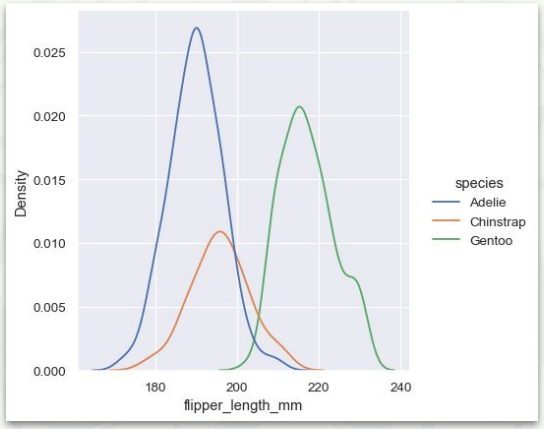




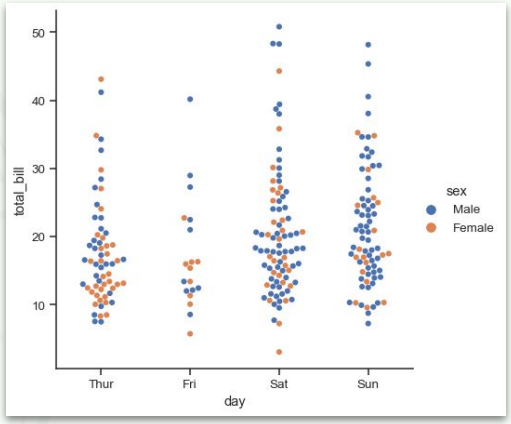
Seaborn “Thinks” in Three Categories



statistical relationships



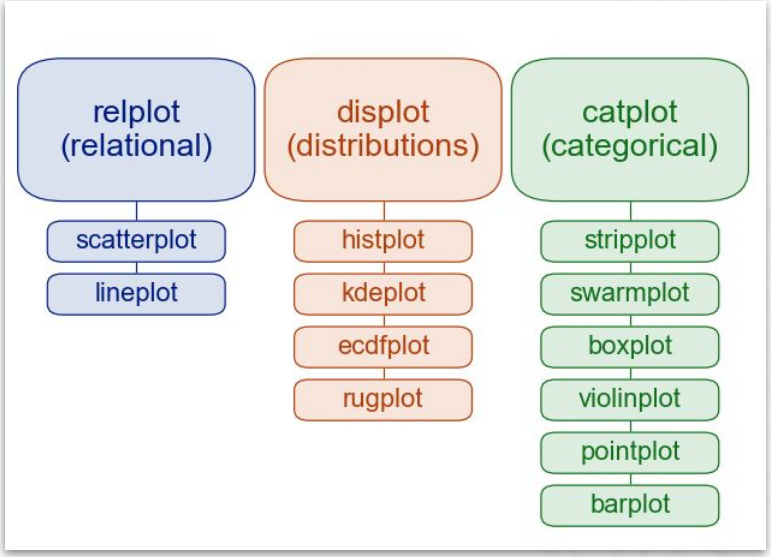
distributions



categories

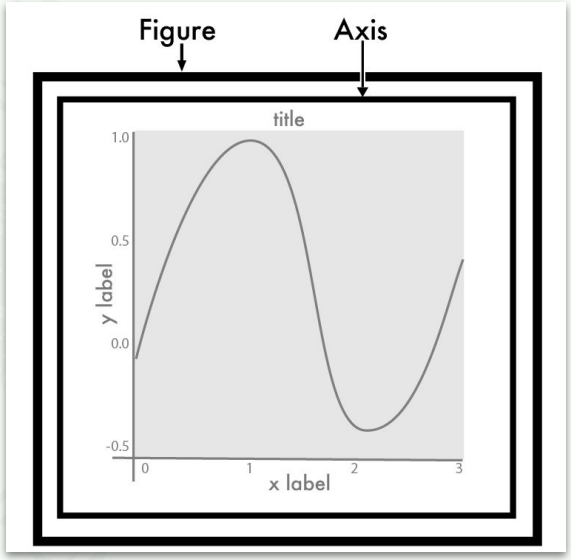
Seaborn Logic

Type of plot



The idea is that you can set up a figure-level plot (e.g., displot) for your data and quickly swap in and out the axis level options (e.g., kdeplot and rugplot) to explore the data in different ways. ***Very powerful!!***

Figure versus Axis level



Some modules refer to the container (figure level) and some refer to what you want to put inside the container (axis level).

Seaborn Can Combine Plots

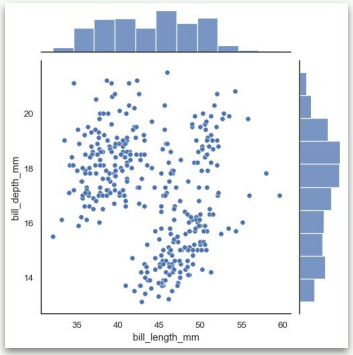
Seaborn gives you two capabilities.

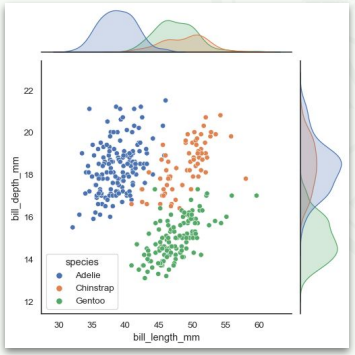
Seaborn is confusing if you don't realize this!

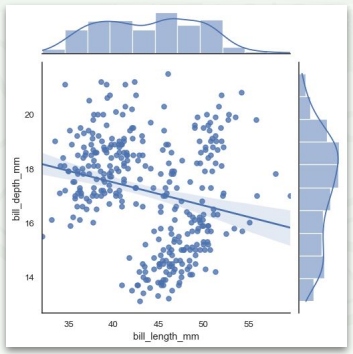
There are "outer" capabilities and there are "inner" capabilities.

Outer ("figure level")

Inner ("axis level")







Example 1

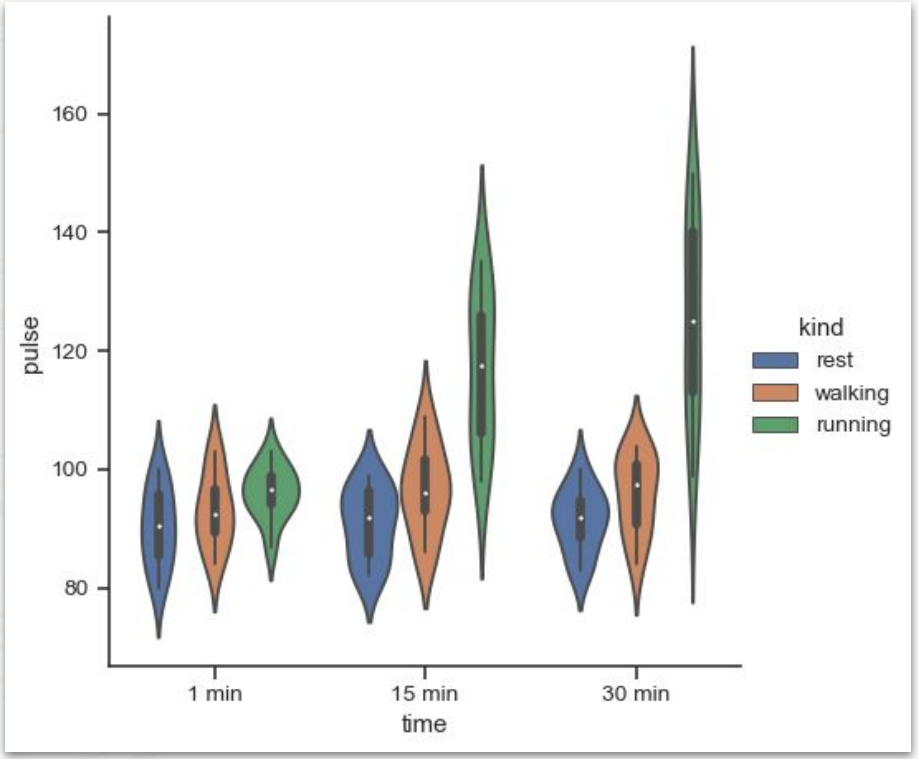
Seaborn works really well with Pandas.

Why?

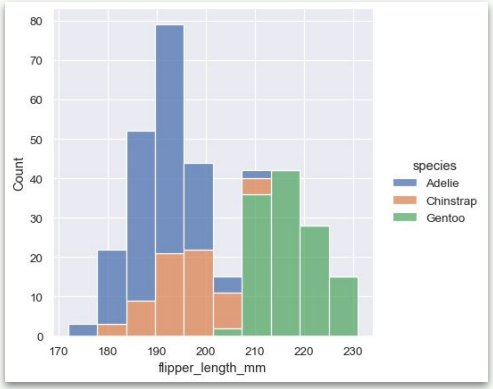
Dataframes contain **labeled** data through the column names.

Seaborn uses those labels.

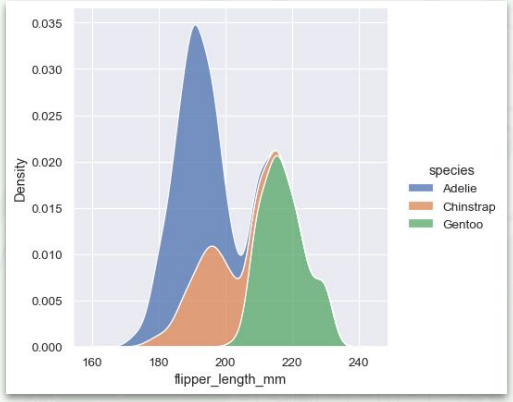
```
g = sns.catplot(x="time",
                y="pulse",
                hue="kind",
                data=exercise,
                kind="violin")
```



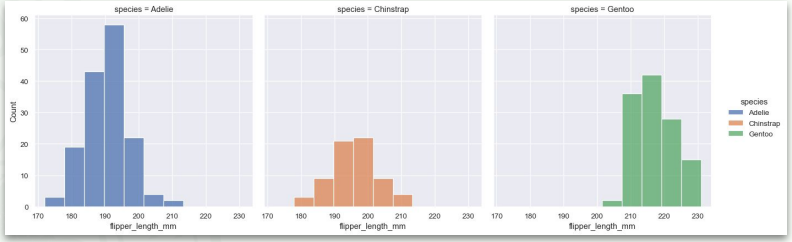
Example 2




```
sns.displot(data=penguins, x="flipper_length_mm", hue="species", multiple="stack")
```




```
sns.displot(data=penguins, x="flipper_length_mm", hue="species", multiple="stack", kind="kde")
```




```
sns.displot(data=penguins, x="flipper_length_mm", hue="species", col="species")
```

The figure level here is `displot` in all three cases, but the axis level can make what is in the container quite different.

This can be confusing unless you know which modules control which level.

seaborn.displot

seaborn.displot (*data=None, *, x=None, y=None, hue=None, row=None, col=None, weights=None, kind='hist', rug=False, rug_kws=None, log_scale=None, legend=True, palette=None, hue_order=None, hue_norm=None, color=None, col_wrap=None, row_order=None, col_order=None, height=5, aspect=1, facet_kws=None, **kwargs*)

Figure-level interface for drawing distribution plots onto a FacetGrid.

Check the documentation.

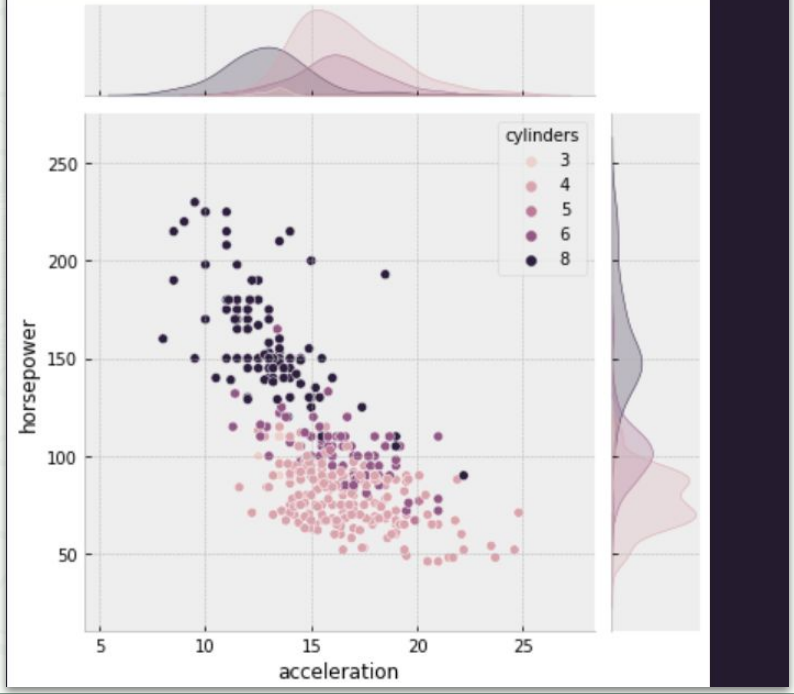
jointplot

Marginal distributions:

$$p(x) = \int dy\, P(x, y)$$

```
1 sns.jointplot(data=df_mpg, x="acceleration",
2                y="horsepower", hue="cylinders")
```

```
<seaborn.axisgrid.JointGrid at 0x7f8ec8790450>
```



Regression

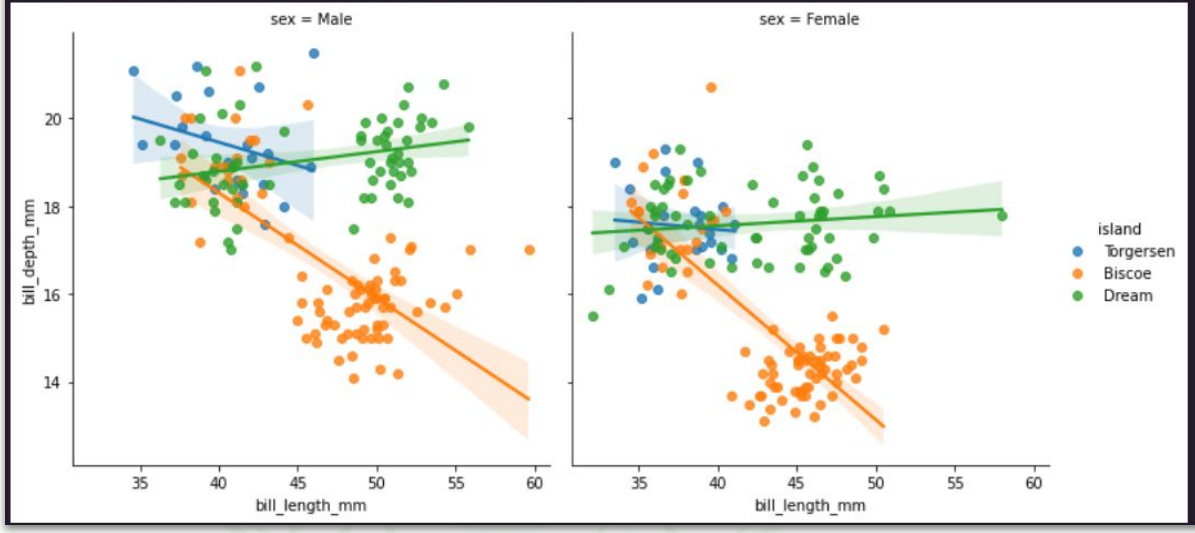
There are two libraries for performing regression:

- `regplot`
- `lmplot`

See the Seaborn documentation for the details of how these differ.

```
5 sns.lmplot(y='bill_depth_mm', x='bill_length_mm', hue='island', col='sex', lowess=False, data=df_p
```

```
<seaborn.axisgrid.FacetGrid at 0x7f8ec93bc390>
```



Regression (Lowess or LOESS)

There are two libraries for performing regression:

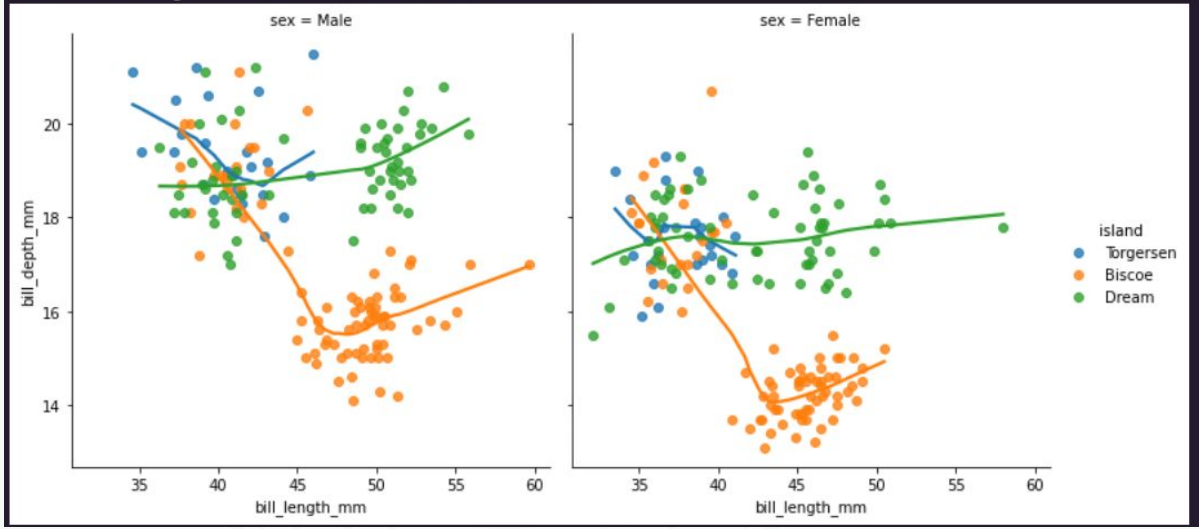
- `regplot`
- `lmplot`

See the Seaborn documentation for the details of how these differ.

Feature	regplot	lmplot
Function type	Lower-level	Higher-level (uses regplot internally)
Data input	Arrays or DataFrame keys	Requires DataFrame and column names
Figure creation	Uses existing axes or creates new one	Always creates new figure with FacetGrid
Faceting support	No	Yes
Scatter plot control	More control	Less direct control
Additional variables	Not directly supported	Can use through faceting or hue
Statistical model	Single regression line	Can fit separate lines for data subsets
Use case	Quick visualizations, more plot control	Exploring relationships across data subsets, faceting
Customization	More flexible for individual plot elements	Better for multi-plot layouts and categorical comparisons
Performance	Generally faster for simple plots	May be slower due to additional features
Axis sharing	Manual (through matplotlib)	Automatic across facets

```
5 sns.lmplot(y='bill_depth_mm', x='bill_length_mm', hue='island', col='sex', lowess=True, data=df_p)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f8ec92a2910>
```



Equations for Lowess (Locally Weighted Scatterplot Smoothing) will come later in the semester....

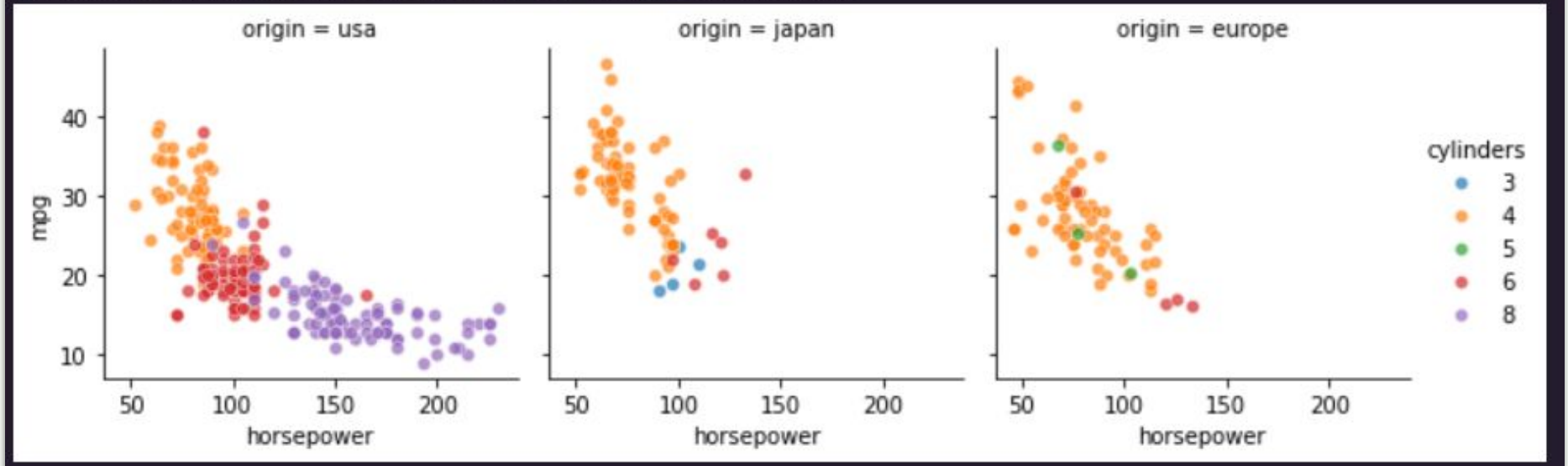
FacetGrid: Deeper Insights!

```
1 g = sns.FacetGrid(df_mpg, col="origin", hue="cylinders")
2 g.map(sns.scatterplot, "horsepower", "mpg", alpha=.7)
3 g.add_legend()
```

creates an object g

uses methods on g

```
<seaborn.axisgrid.FacetGrid at 0x7f8ecc82d0d0>
```

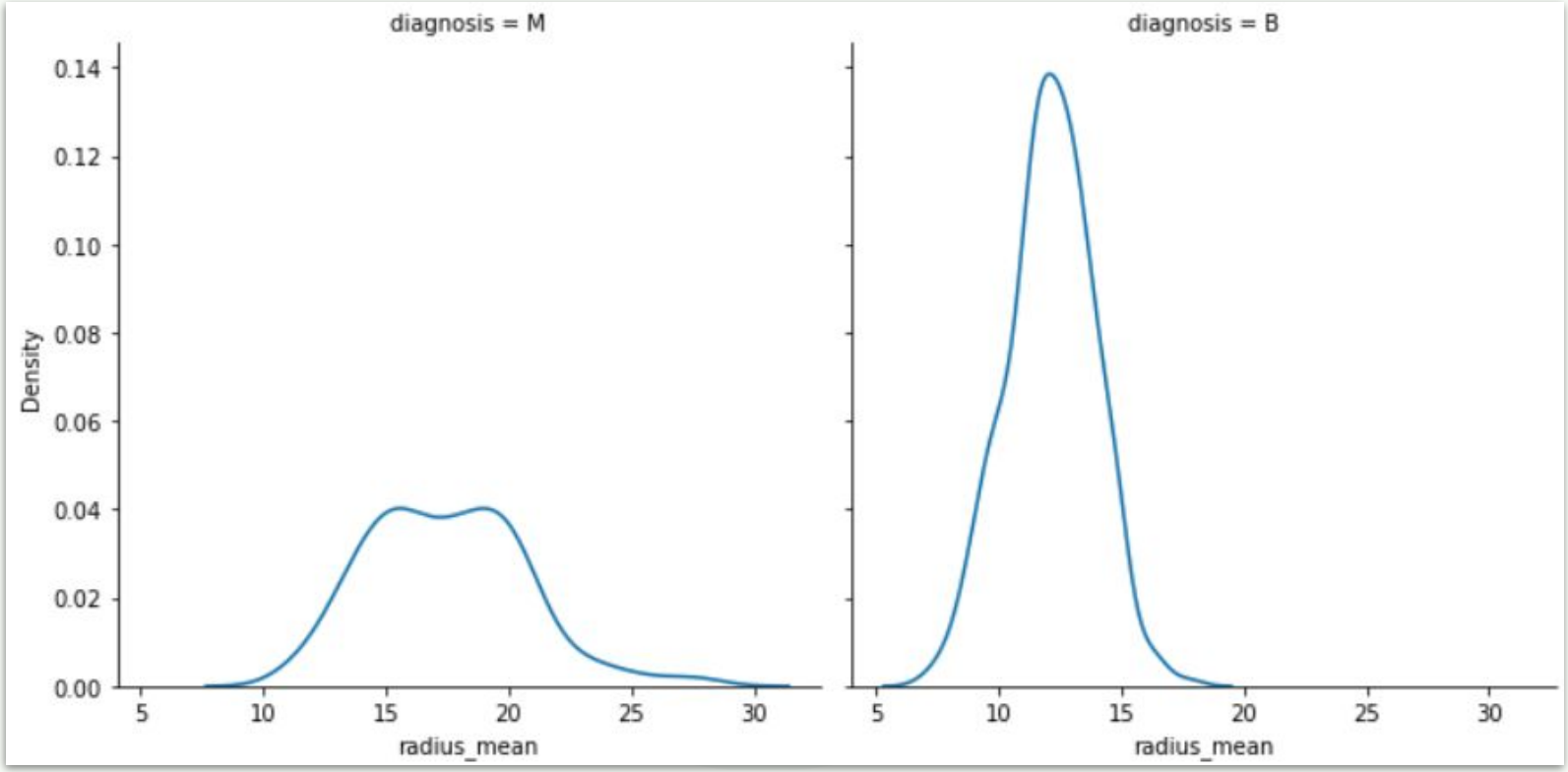


ICA on Thursday

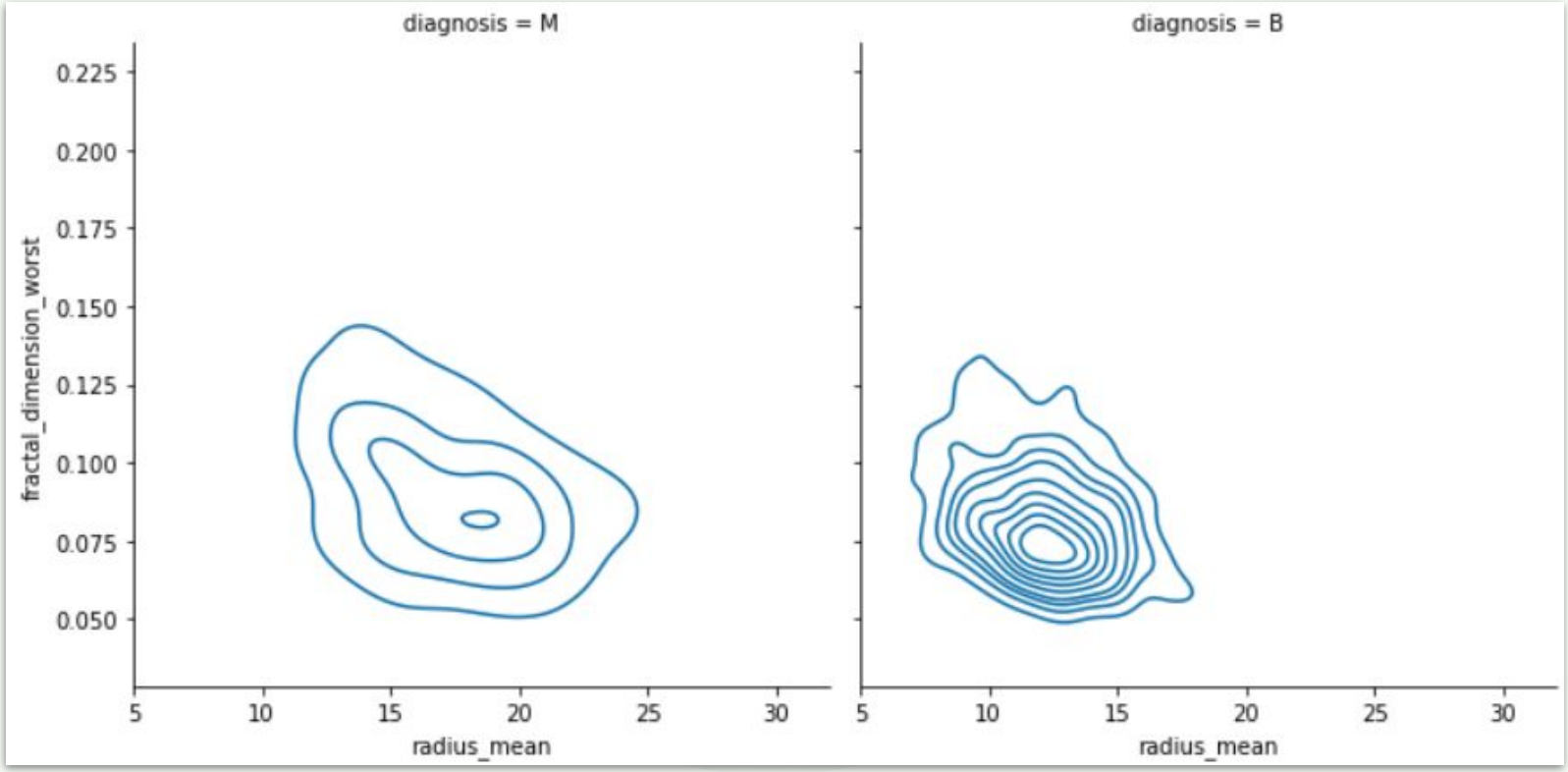
1. Seaborn
    a. get a new dataset (WI cancer) and read it in (`.read_csv`)
    b. perform basic IDA and EDA
    c. from the EDA design 3-5 informative Seaborn plots
    d. **design** a web application

2. Streamlit
    a. input information into streamlit
    b. **build** the web application into streamlit

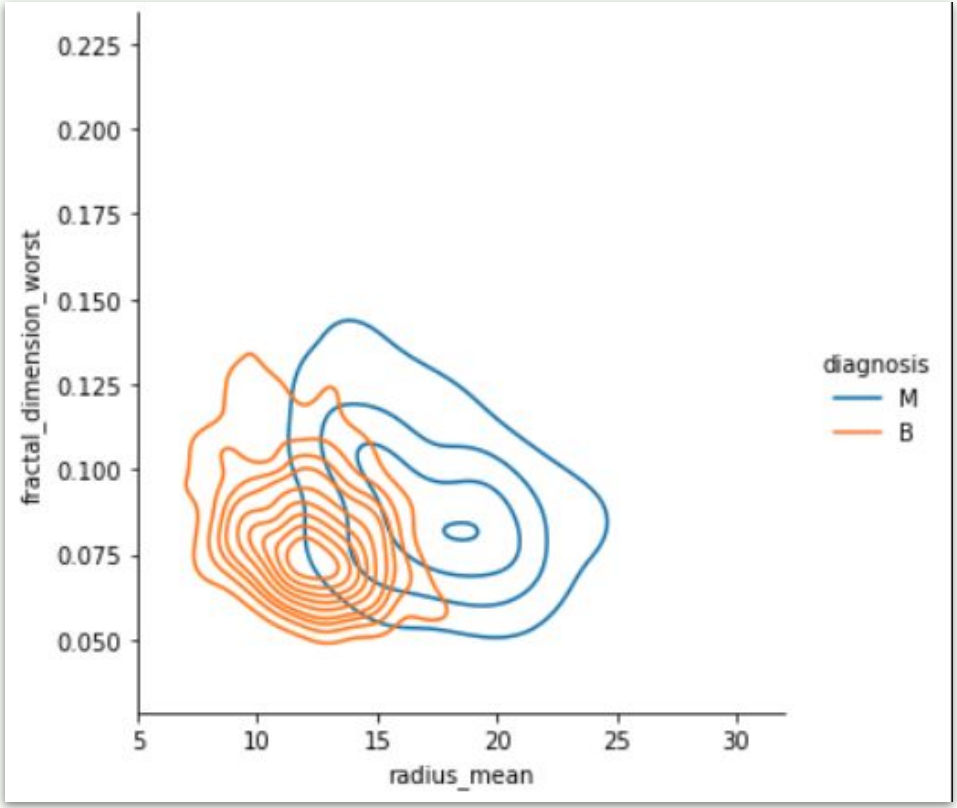
Examples


diagnosis = M
diagnosis = B
Density
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
5
10
15
20
25
30
radius_mean

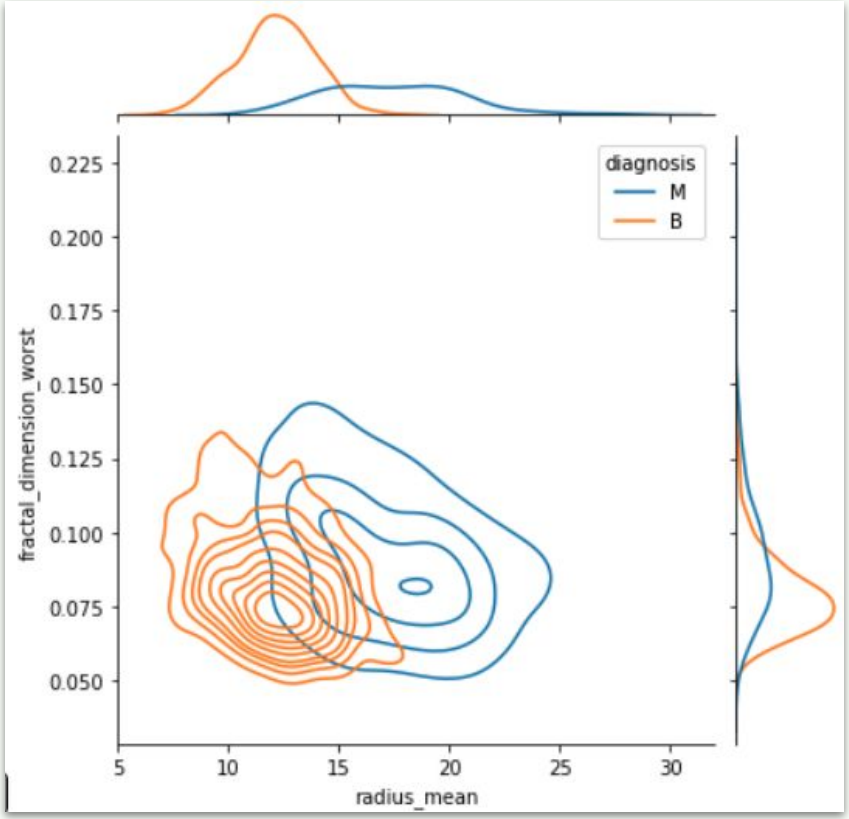
Radius Doesn’t Capture The Full Story



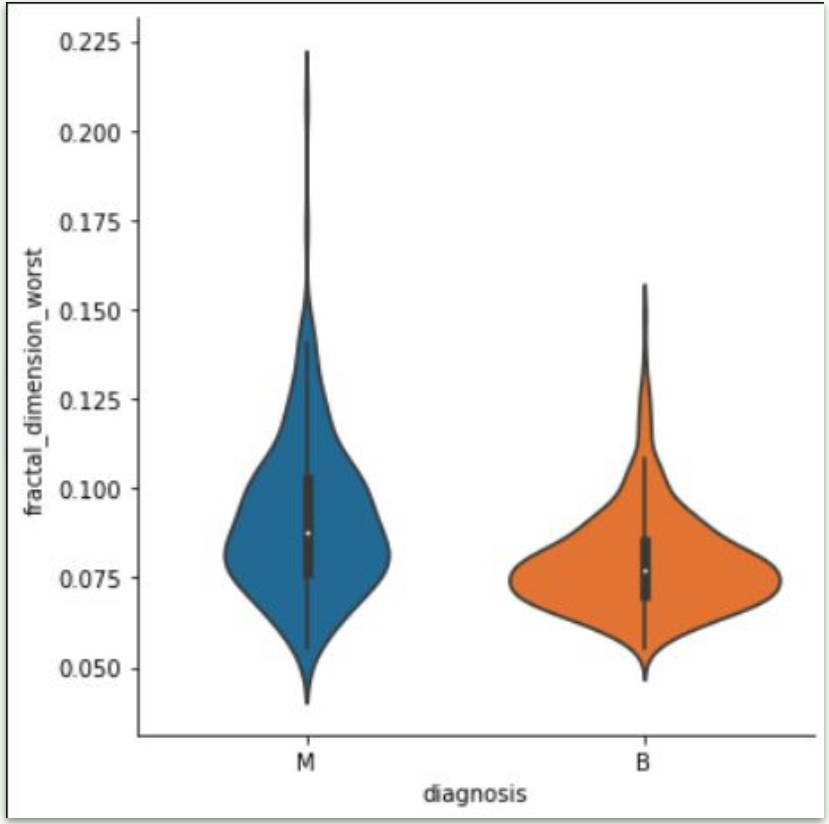
We Get More With One Plot



Probability Distributions Reveal Additional Information



I Should Have Used a Violin Plot!



See You Thursday!