Data Science: Matplotlib and Seaborn

CPSC 501: Advanced Programming Techniques
Fall 2022

Jonathan Hudson, Ph.D
Assistant Professor (Teaching)
Department of Computer Science
University of Calgary

Monday, October 3, 2022

UNIVERSITY OF CALGARY

matplotlib

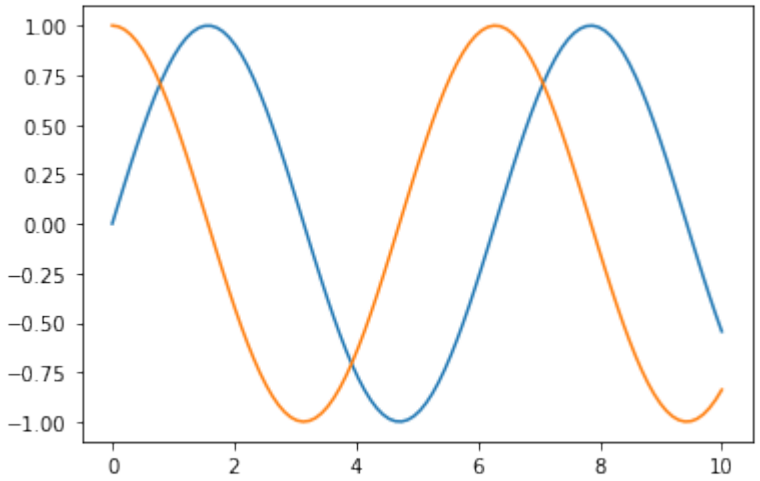
- **matplotlib** (2003) OG chart making inspired by MATLAB
- It received an early boost when it was adopted as the plotting package of choice of the Space Telescope Science Institute (the folks behind the Hubble Telescope), which financially supported Matplotlib's development and greatly expanded its capabilities.
- Big benefit (script chart making)
- Larger base has used it
- Style and usage is dated (one big reason why R and ggplot is popular) is that they are simpler (especially for non-programmers)
- Seaborn often used to look better, but others like ggpy, Holoviews, Altair exist

seaborn

- seaborn improvement on top of matplotlib
- Hides a lot of boilerplate
- Uses panadas dataframes which came post matplotlib
- Good defaults and bunch of presets (like ggplot in R)
- 2.0 matplotlib is response to try and integrate seaborn ideas

Quick matplotlib

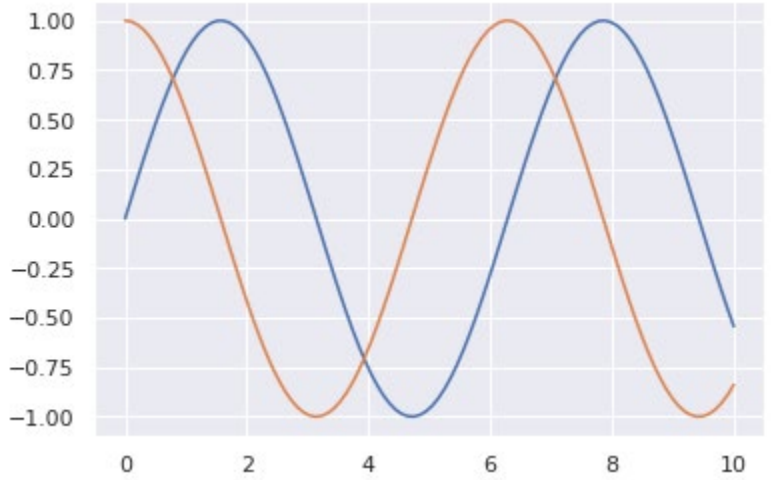
```
import matplotlib.pyplot as plt
import numpy as np


x = np.linspace(0, 10, 100)

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

plt.show()
```



The plt.show() command does a lot under the hood, as it must interact with your system's interactive graphical backend.

Quick seaboarn

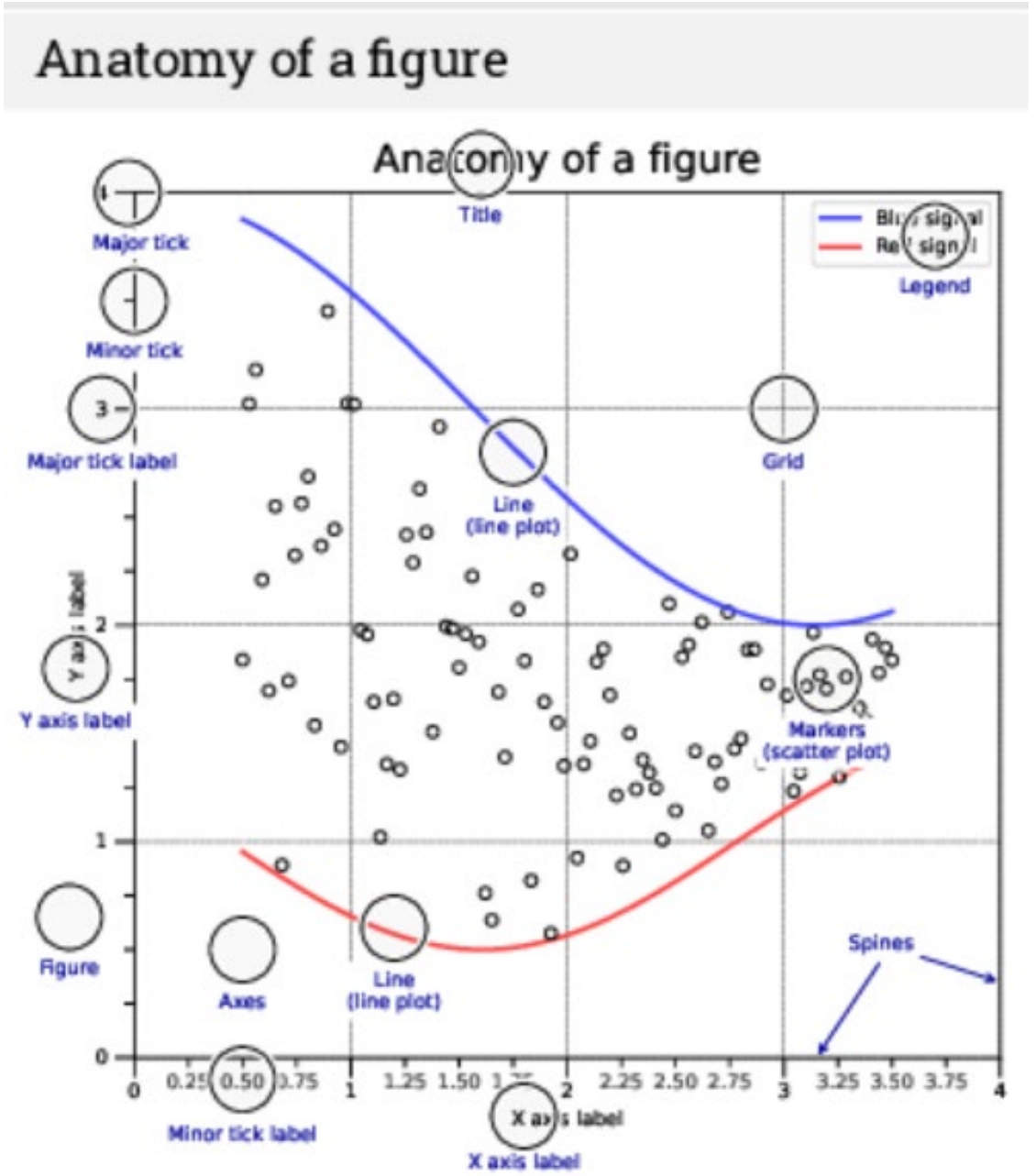
```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
sns.set()
x = np.linspace(0, 10, 100)

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

plt.show()
```



The plt.show() command does a lot under the hood, as it must interact with your system's interactive graphical backend.

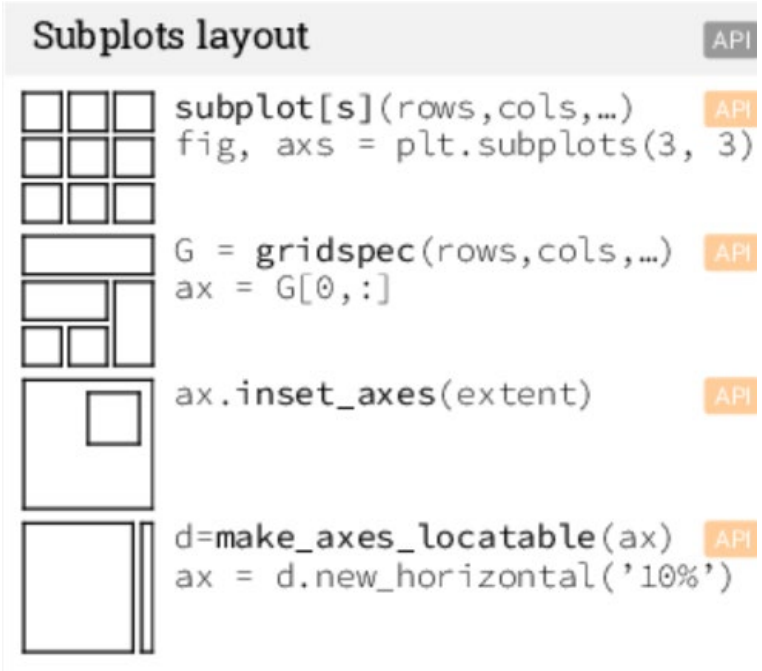
Matplotlib parts

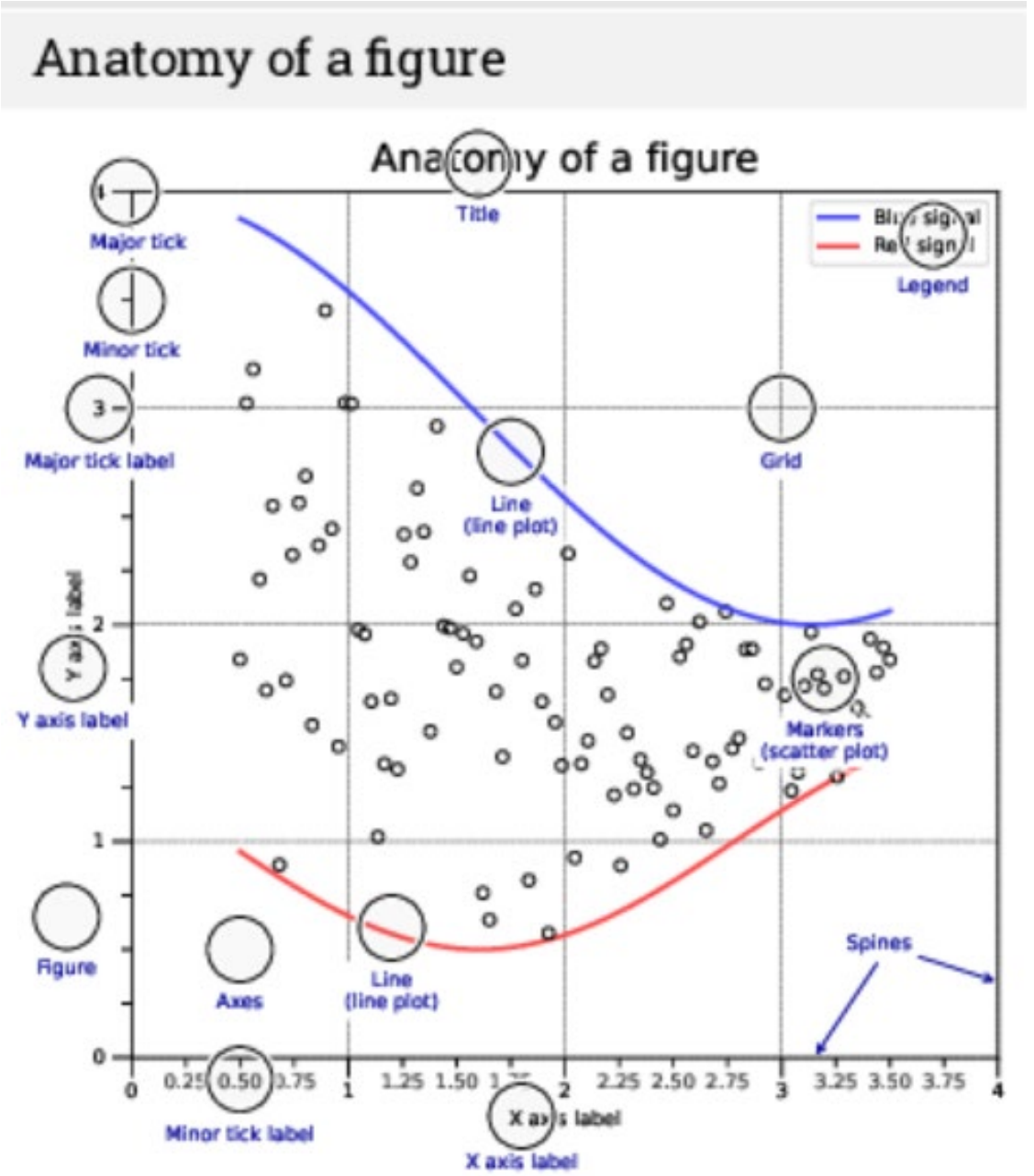
- Matplotlib parts are sensibly named



Matplotlib parts

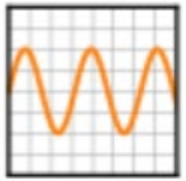
- Matplotlib parts are sensibly named
- Is support for layouts but not near as natural as R





Matplotlib plot types

Basic plots



`plot([X],Y,[fmt],…)` API
X, **Y**, fmt, color, marker, linestyle

`scatter(X,Y,…)` API
X, **Y**, [s]izes, [c]olors, marker, cmap

`bar[h](x,height,…)` API
x, **height**, width, bottom, align, color

`imshow(Z,…)` API
Z, cmap, interpolation, extent, origin

`contour[f]([X],[Y],Z,…)` API
X, Y, **Z**, levels, colors, extent, origin

`pcolormesh([X],[Y],Z,…)` API
X, Y, **Z**, vmin, vmax, cmap

`quiver([X],[Y],U,V,…)` API
X, Y, **U**, **V**, C, units, angles

`pie(X,…)` API
Z, explode, labels, colors, radius

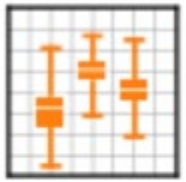
`text(x,y,text,…)` API
x, **y**, **text**, va, ha, size, weight, transform

`fill[_between][x](…)` API
X, Y1, Y2, color, where

Matplotlib plot types

Advanced plots

`step(X,Y,[fmt],…)` API
X, **Y**, fmt, color, marker, where



`boxplot(X,…)` API
X, notch, sym, bootstrap, widths

`errorbar(X,Y,xerr,yerr,…)` API
X, **Y**, xerr, yerr, fmt

`hist(X, bins, …)` API
X, bins, range, density, weights

`violinplot(D,…)` API
D, positions, widths, vert

`barbs([X],[Y], U, V, …)` API
X, Y, **U**, **V**, C, length, pivot, sizes

`eventplot(positions,…)` API
positions, orientation, lineoffsets

`hexbin(X,Y,C,…)` API
X, **Y**, C, gridsize, bins

Matplotlib modify

Scales

API

```
ax.set_[xy]scale(scale,…)
```

linear
any values

log
values > 0

symlog
any values

logit
0 < values < 1

Projections

API

```
subplot(…,projection=p)
```

p='polar'

p='3d'

```
p=Orthographic()
from cartopy.crs import Cartographic
```

API

Lines

API

linestyle or ls

capstyle or dash_capstyle

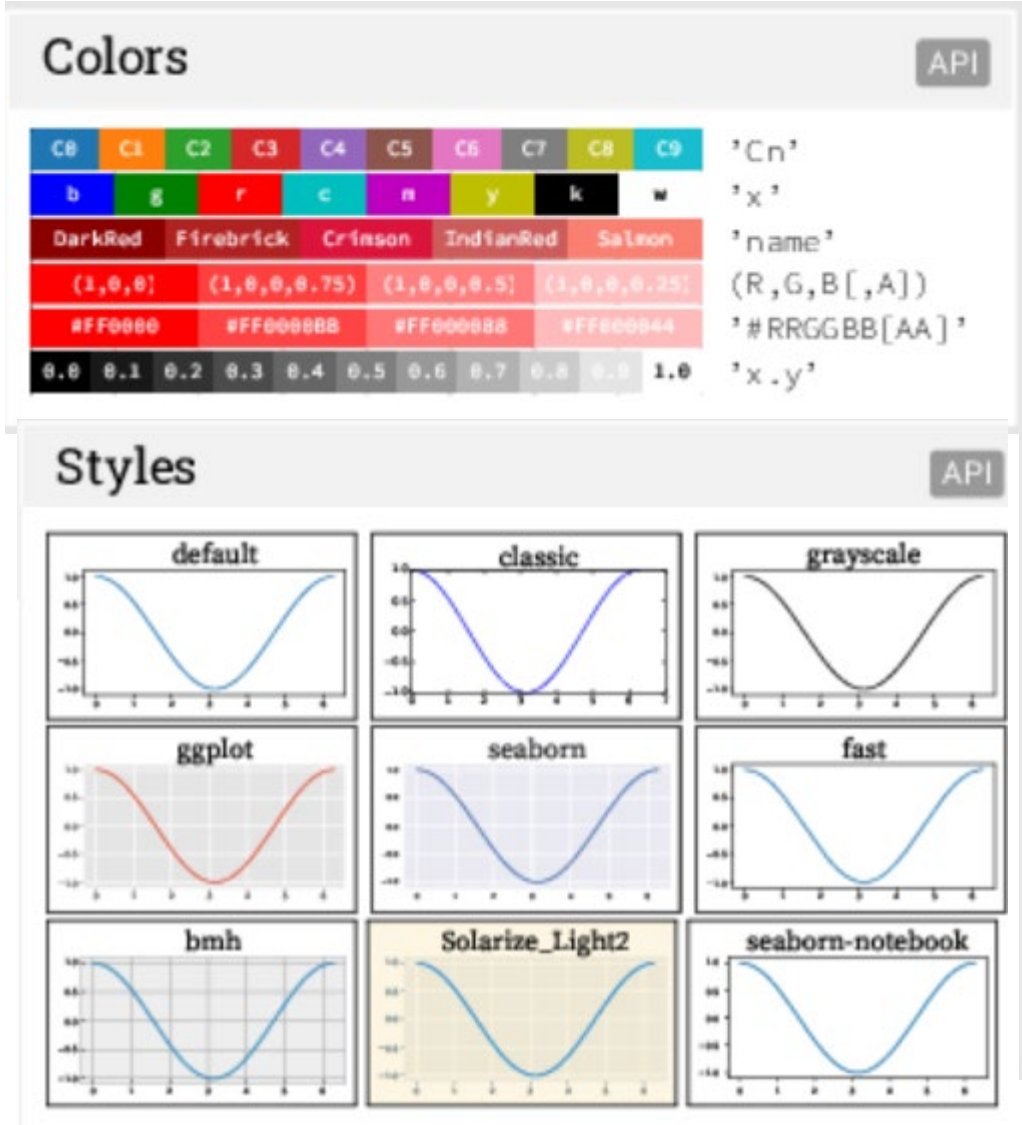
Markers

API

markevery

UNIVERSITY OF CALGARY

Matplotlib colours



Matplotlib other

- Variety of methods to determine ticks
- Or format them
- Labels/annotations can be added
- Even event handling/animation is possible

Event handling API

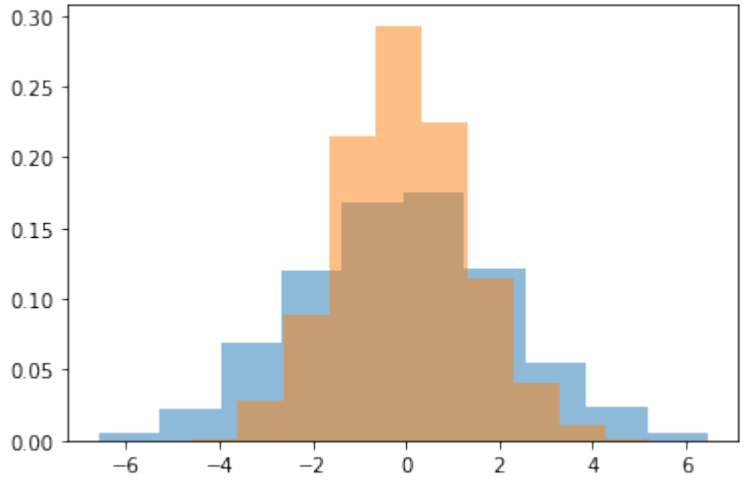
```
fig, ax = plt.subplots()
def on_click(event):
  print(event)
fig.canvas.mpl_connect(
  'button_press_event', on_click)
```

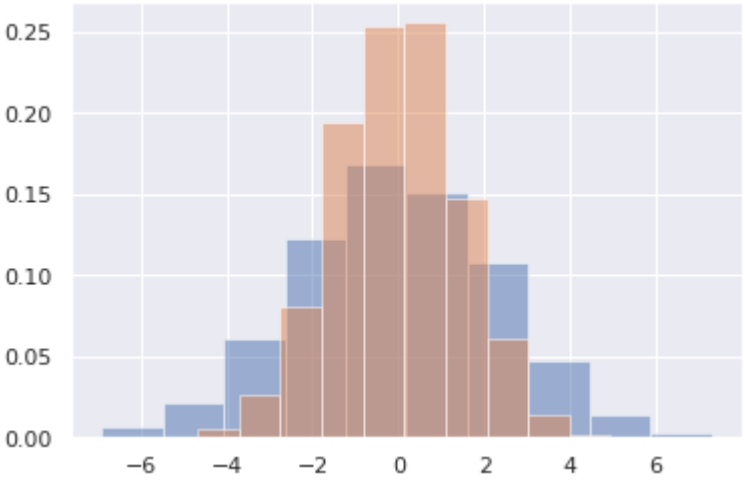
Animation API

```
import matplotlib.animation as mpla

T = np.linspace(0, 2*np.pi, 100)
S = np.sin(T)
line, = plt.plot(T, S)
def animate(i):
    line.set_ydata(np.sin(T+i/50))
anim = mpla.FuncAnimation(
    plt.gcf(), animate, interval=5)
plt.show()
```

Matplotlib other

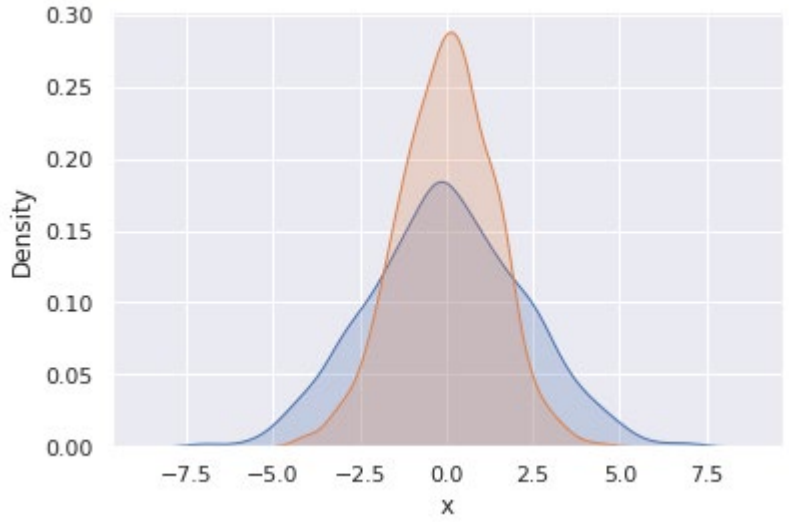
```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
data = np.random.multivariate_normal([0, 0], [[5, 2], [2, 2]], size=2000)
data = pd.DataFrame(data, columns=['x', 'y'])

for col in 'xy':
    plt.hist(data[col], density=True, alpha=0.5)
```



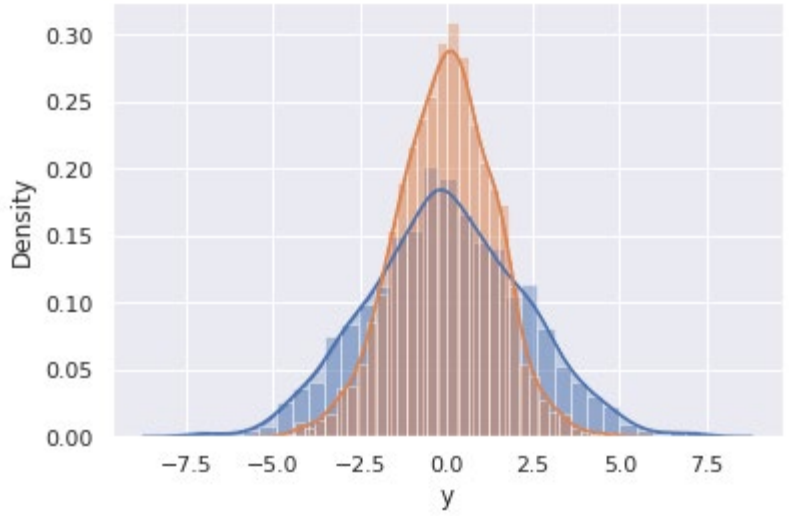


Seaborn

```
for col in 'xy':
    sns.kdeplot(data[col], shade=True)
```

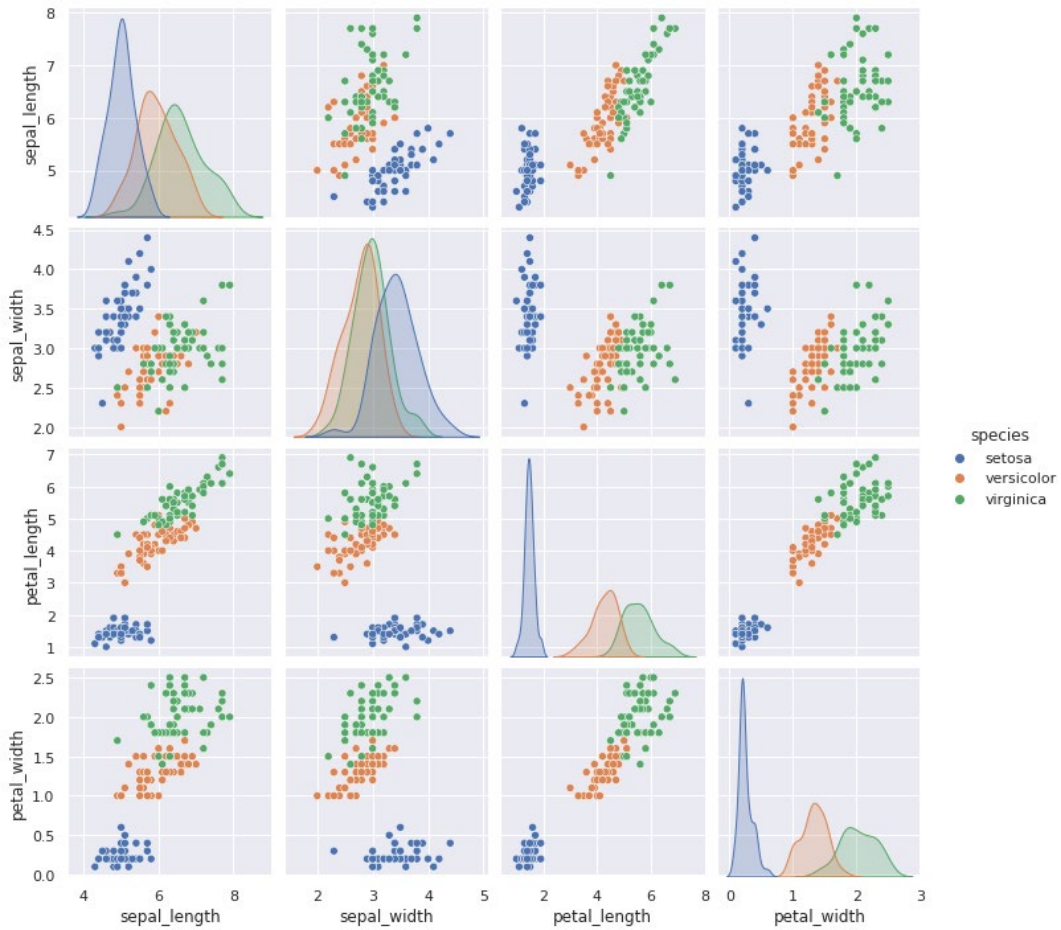



```
#Add in
sns.distplot(data['x'])
sns.distplot(data['y']);
```



Seaborn pair plots

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

iris = sns.load_dataset("iris")
sns.pairplot(iris, hue='species', height=2.5);
```



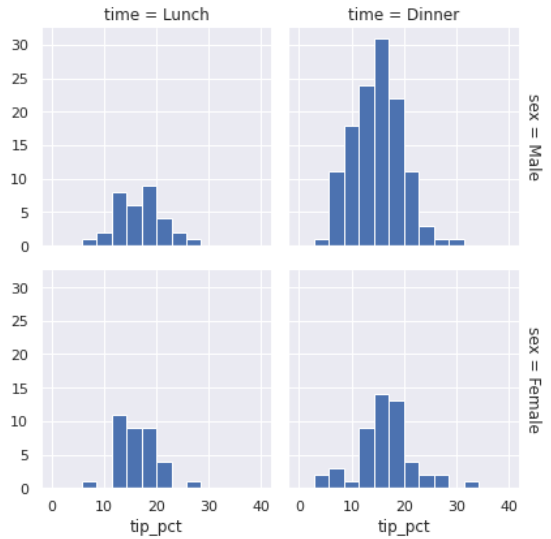
Seaborn others

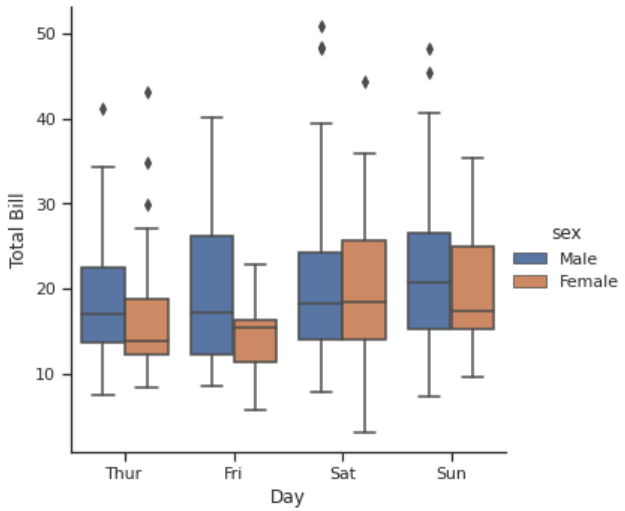
Faceted histograms, sns.FacetGrid

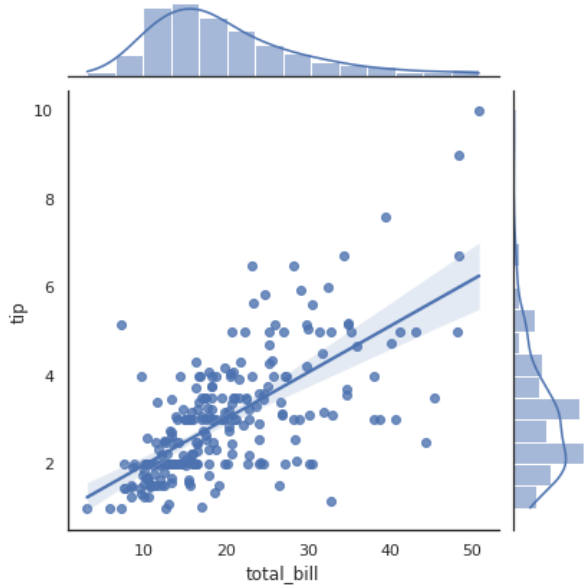
Factor (Category plots) sns.catplot

Joint Distributions sns.jointplot

So much easier (like in R)







UNIVERSITY OF CALGARY

Onward to …
machine learning.

Jonathan Hudson
jwhudson@ucalgary.ca
https://pages.cpsc.ucalgary.ca/~jwhudson/

UNIVERSITY OF CALGARY