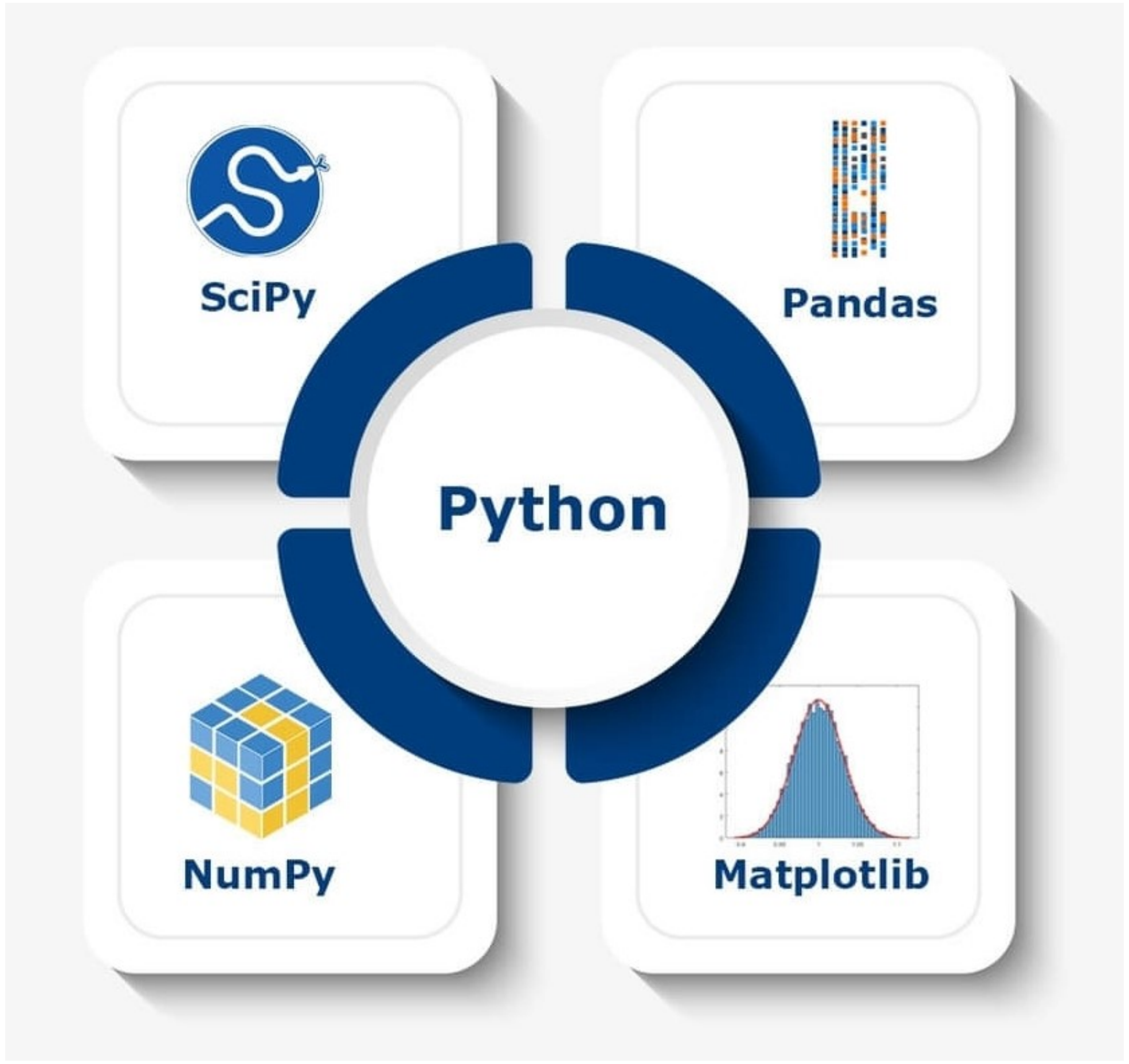
Python for Machine Learning IV

Dr. Junya Michanan

Agenda

- Matplotlib
- Seaborn
- Class Activity
- Homework



What is Matplotlib?

- Matplotlib is a low-level graph plotting library in python that serves as a visualization utility.
- Matplotlib was created by John D. Hunter.
- Matplotlib is opensource and we can use it freely.
- Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

Installation of Matplotlib

- Installation of Matplotlib

```
C:\Users\Your Name>pip install matplotlib
```

- Import Matplotlib

Once Matplotlib is installed, import it in your applications by adding the import module statement:

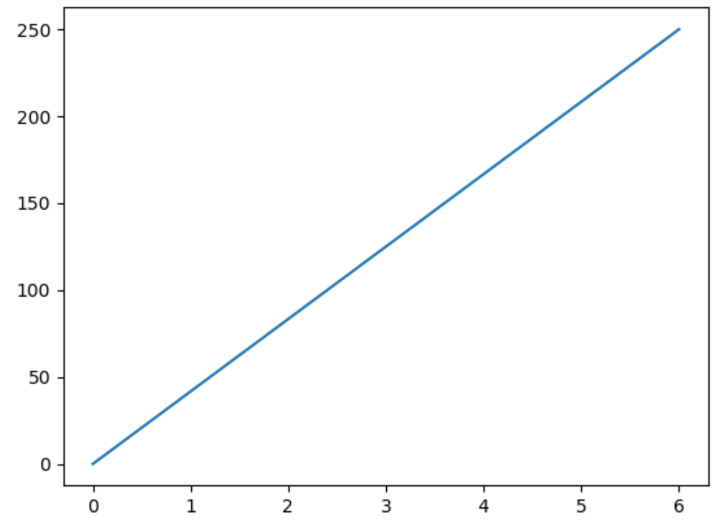
```
import matplotlib
```

Matplotlib Pyplot

- Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias:

Example: Draw a line in a diagram from position (0,0) to position (6,250):

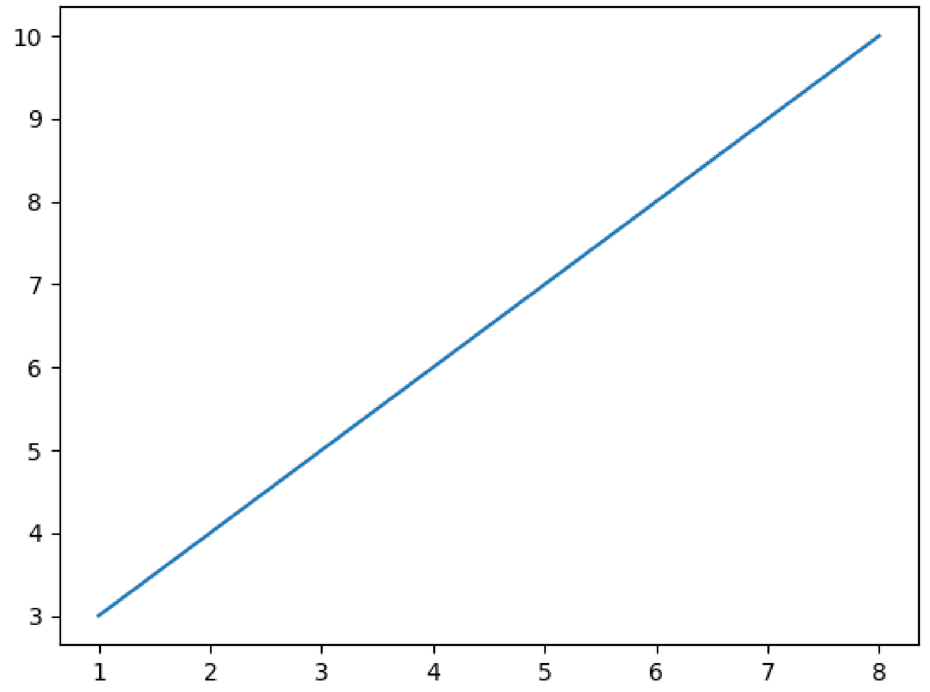
```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([0, 6])
ypoints = np.array([0, 250])

plt.plot(xpoints, ypoints)
plt.show()
```



Matplotlib Plotting

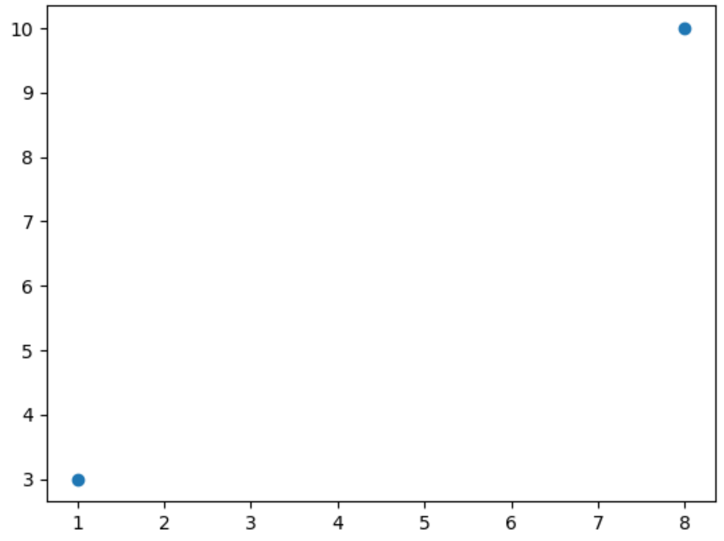
The `plot()` function is used to draw points (markers) in a diagram.
By default, the `plot()` function draws a line from point to point.
The function takes parameters for specifying points in the diagram.
Parameter 1 is an array containing the points on the **x-axis**.
Parameter 2 is an array containing the points on the **y-axis**.
If we need to plot a line from (1, 3) to (8, 10), we have to pass two arrays [1, 8] and [3, 10] to the plot function.

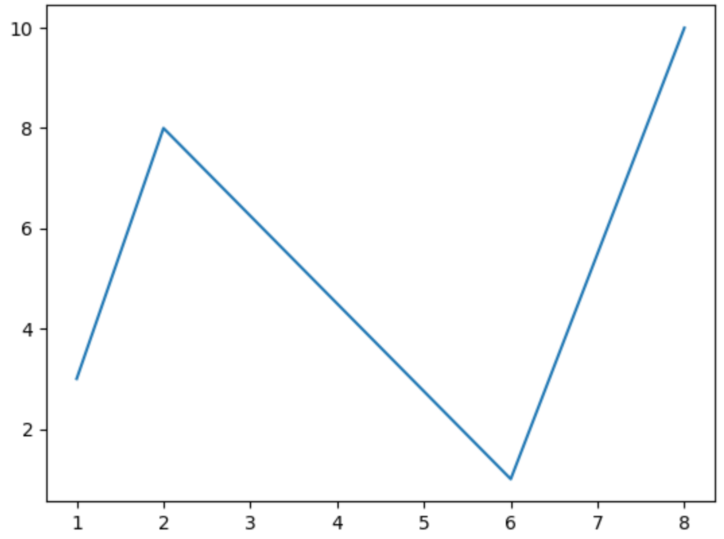
```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([1, 8])
ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints)
plt.show()
```



Matplotlib Plotting

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

xpoints = np.array([1, 8])
ypoints = np.array([3, 10])

#ploting without a line
plt.plot(xpoints, ypoints, 'o’)
plt.show()

#Multiple Points
xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])

plt.plot(xpoints, ypoints)
plt.show()
```

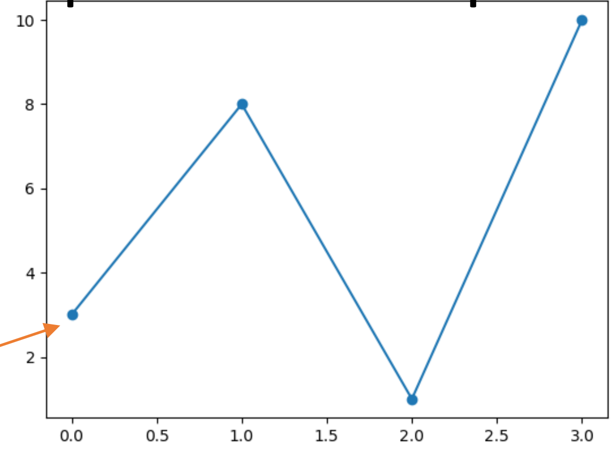


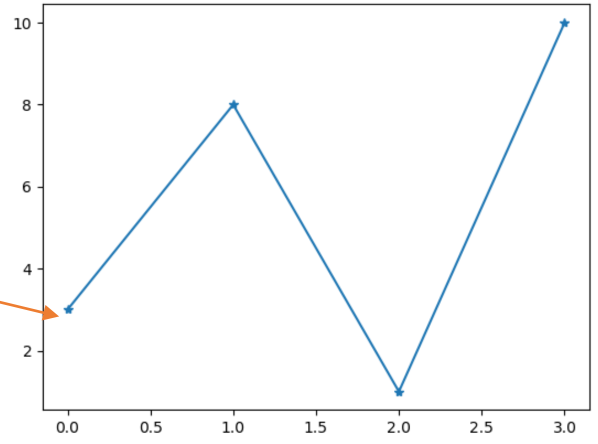


Matplotlib Markers

- You can use the keyword argument marker to emphasize each point with a specified marker:

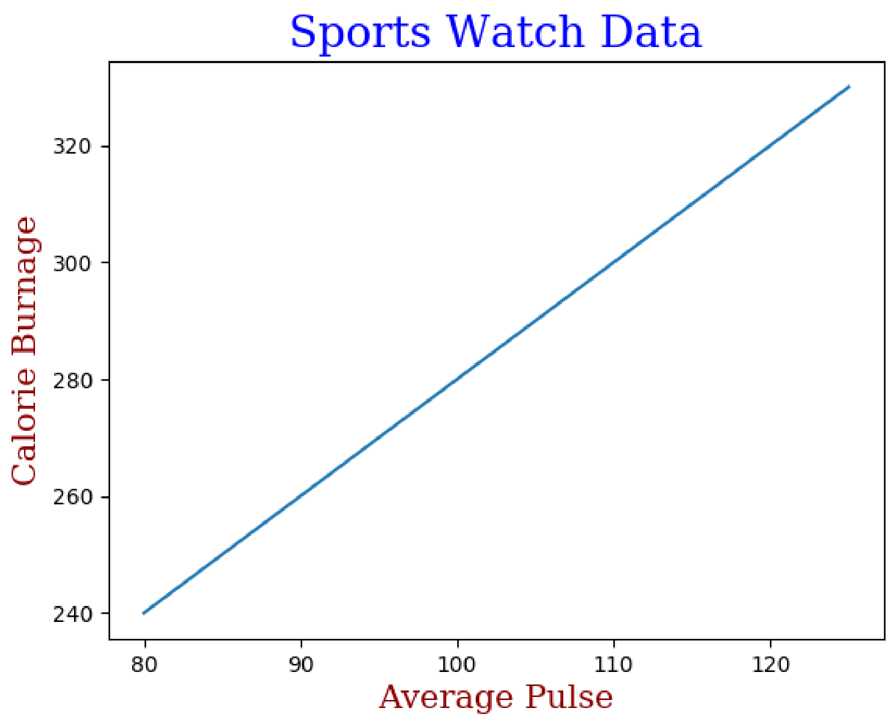
```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o')
plt.show()

plt.plot(ypoints, marker = '*’)
plt.show()
```





Matplotlib Labels and Title

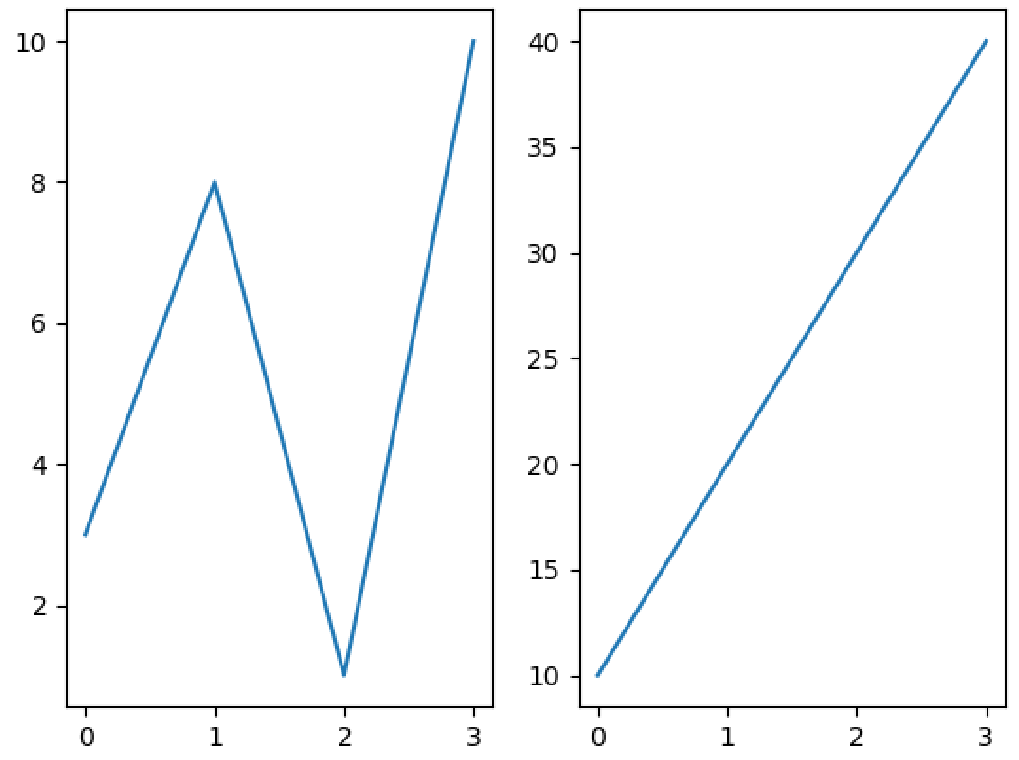
```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

font1 = {'family':'serif','color':'blue','size':20}
font2 = {'family':'serif','color':'darkred','size':15}

plt.title("Sports Watch Data", fontdict = font1)
plt.xlabel("Average Pulse", fontdict = font2)
plt.ylabel("Calorie Burnage", fontdict = font2)

plt.plot(x, y)
plt.show()
```



Matplotlib Subplots

- With the subplots() function you can draw multiple plots in one figure:

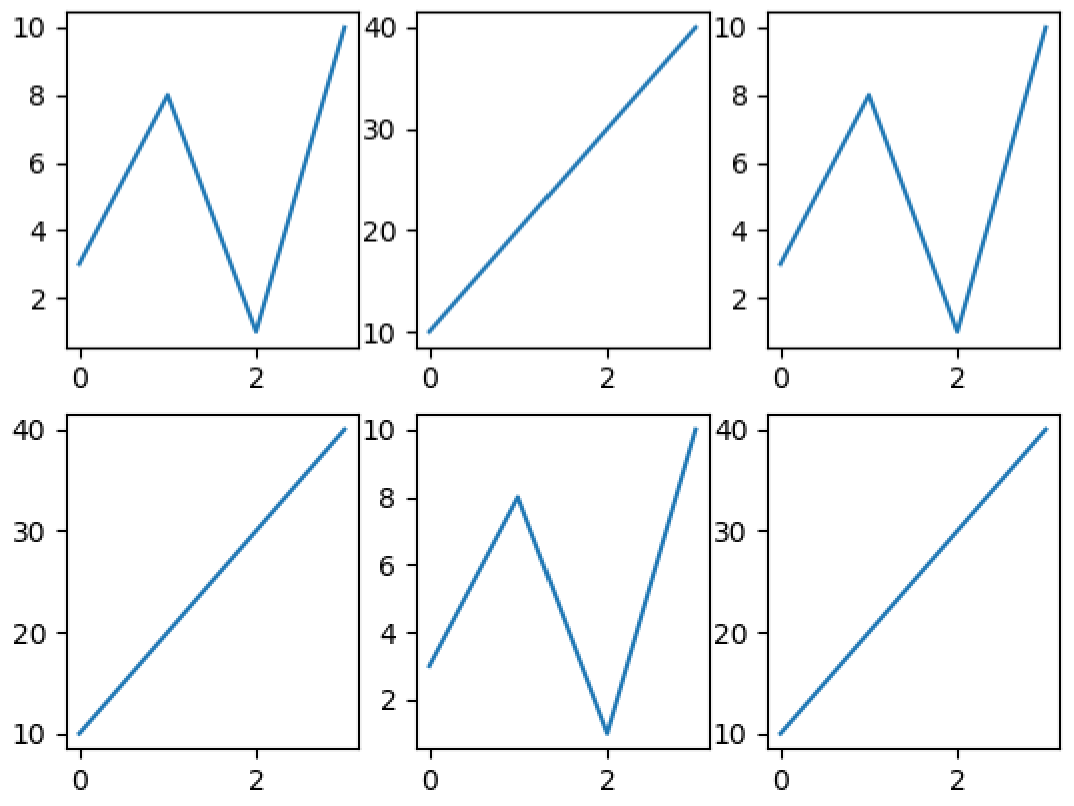
```
import matplotlib.pyplot as plt
import numpy as np

#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

#the figure has 1 row, 2 columns, and this plot is the first plot.
plt.subplot(1, 2, 1)
plt.plot(x,y)

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

#the figure has 1 row, 2 columns, and this plot is the second plot.
plt.subplot(1, 2, 2)
plt.plot(x,y)

plt.show()
```



Matplotlib Subplots

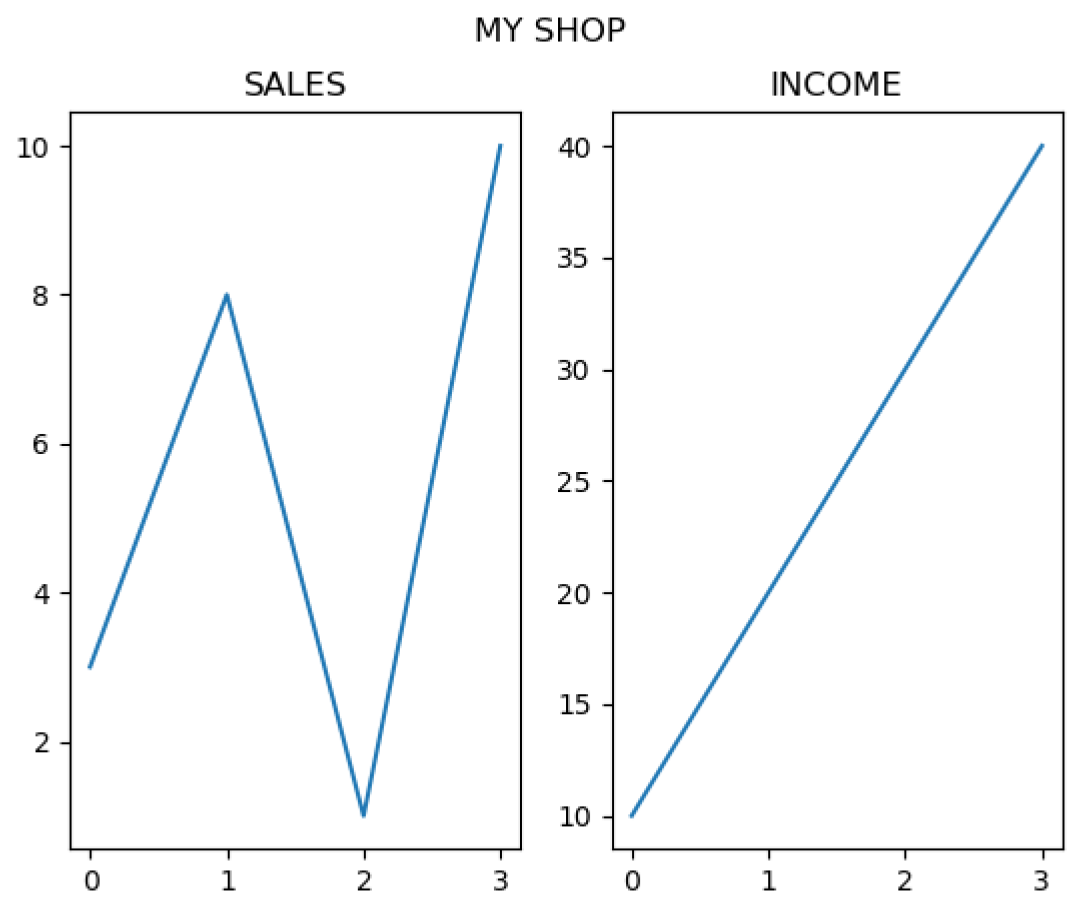
You can draw as many plots you like on one figure, just descibe the number of rows, columns, and the index of the plot.

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

x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(2, 3, 1)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(2, 3, 2)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(2, 3, 3)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(2, 3, 4)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(2, 3, 5)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(2, 3, 6)
plt.plot(x,y)

plt.show()
```



Matplotlib Subplots

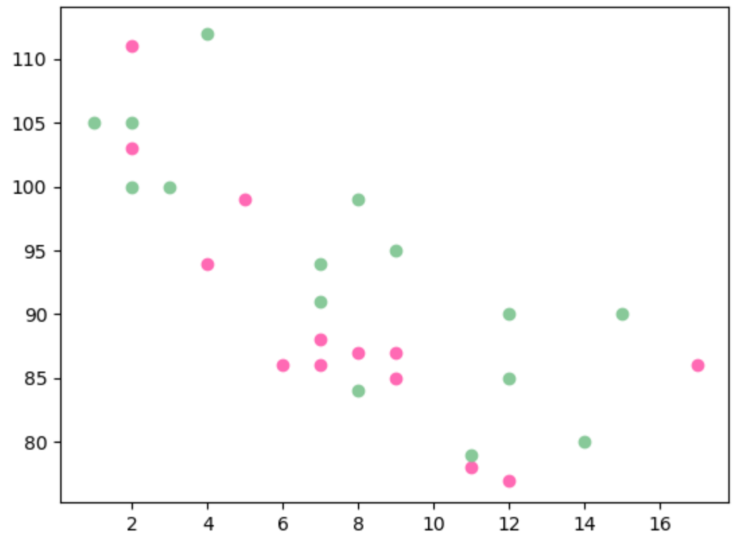
```
import matplotlib.pyplot as plt
import numpy as np

#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(1, 2, 1)
plt.plot(x,y)
plt.title("SALES")

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(1, 2, 2)
plt.plot(x,y)
plt.title("INCOME")

plt.suptitle("MY SHOP")
plt.show()
```



Matplotlib Scatter

- With Pyplot, you can use the scatter() function to draw a scatter plot.
- The scatter() function plots one dot for each observation. It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis:

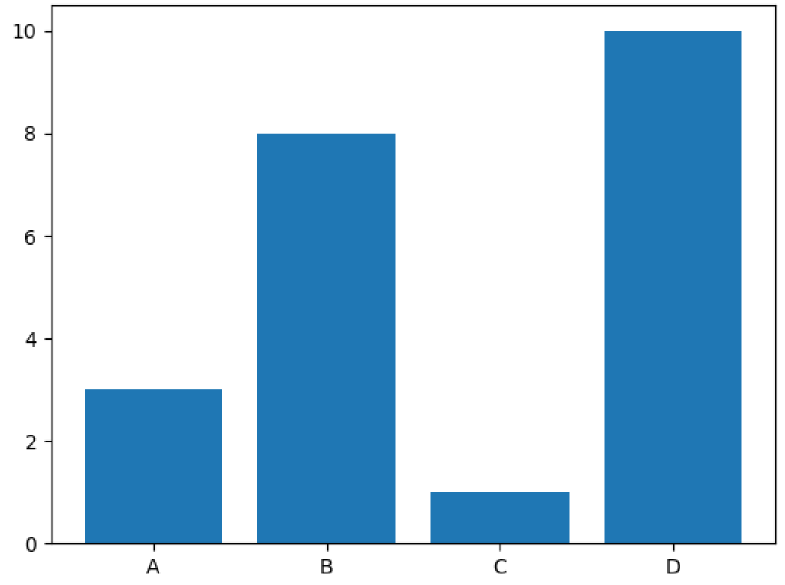
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x, y, color = 'hotpink')

x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
y =
np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
plt.scatter(x, y, color = '#88c999')

plt.show()
```



Matplotlib Bars

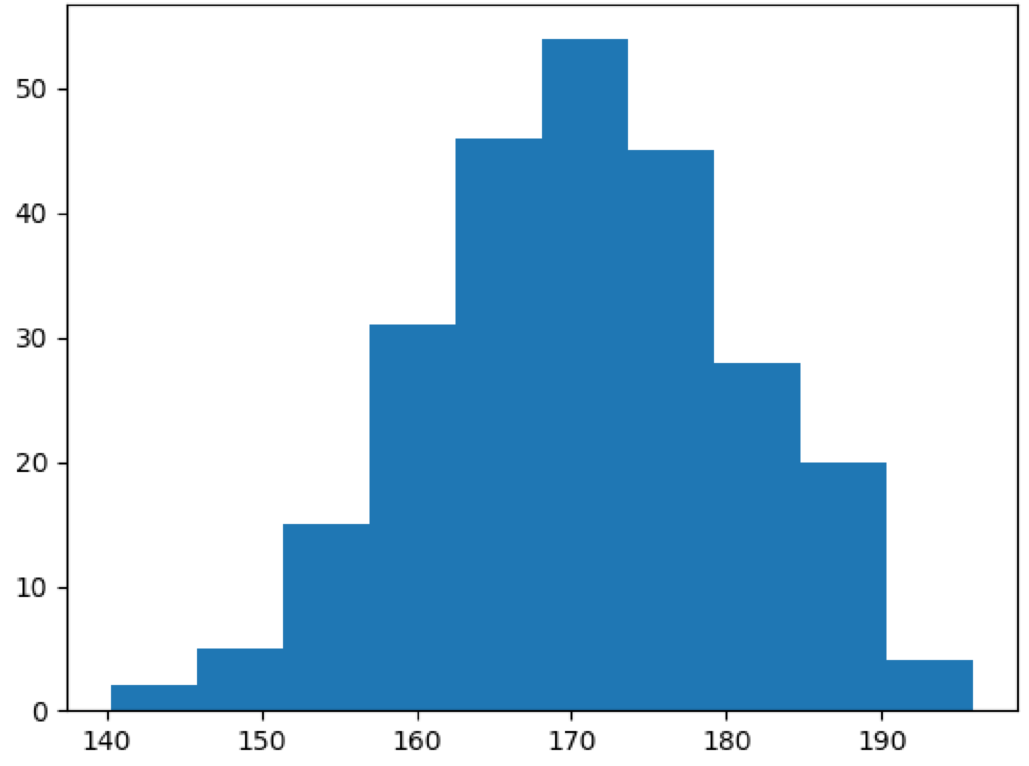
- With Pyplot, you can use the bar() function to draw bar graphs:

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

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x,y)
plt.show()
```



Matplotlib Histograms

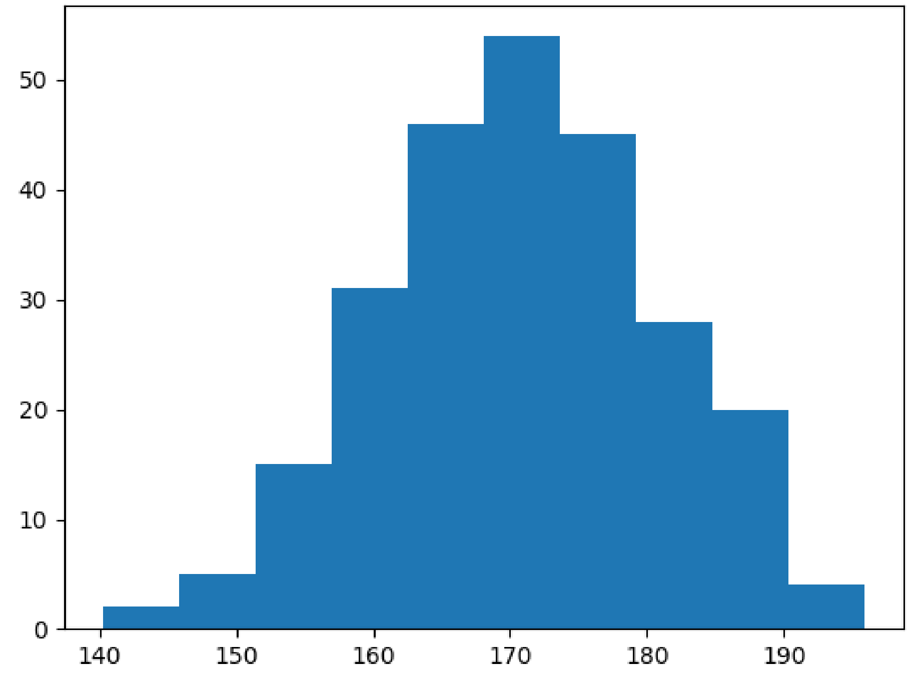
- A histogram is a graph showing frequency distributions.
- It is a graph showing the number of observations within each given interval.
- Example: Say you ask for the height of 250 people, you might end up with a histogram like this:



You can read from the histogram that there are approximately:
2 people from 140 to 145cm
5 people from 145 to 150cm
15 people from 151 to 156cm
31 people from 157 to 162cm
46 people from 163 to 168cm
53 people from 168 to 173cm
45 people from 173 to 178cm
28 people from 179 to 184cm
21 people from 185 to 190cm
4 people from 190 to 195cm

Matplotlib Histograms

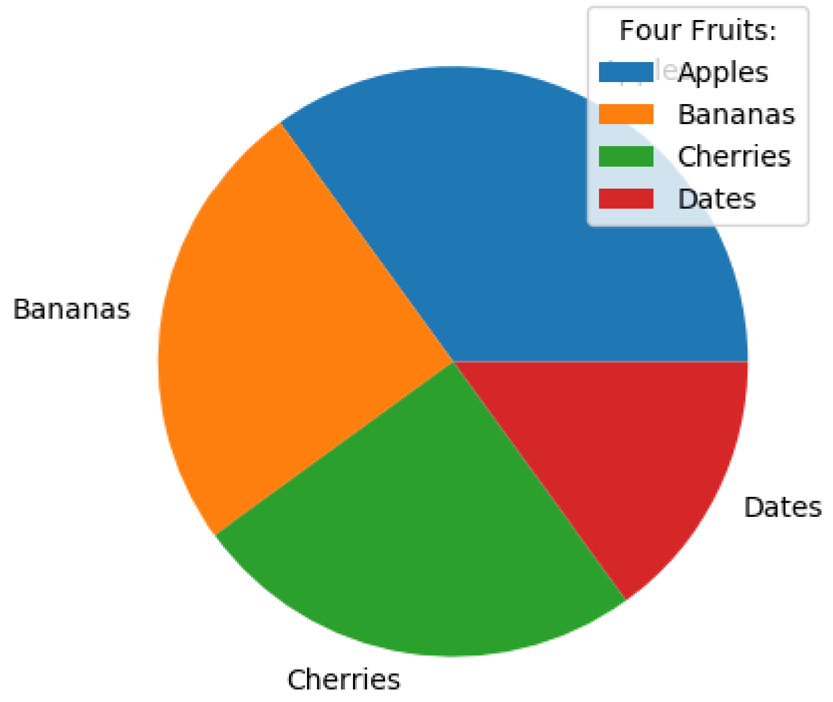
```python
import matplotlib.pyplot as plt
import numpy as np

x = np.random.normal(170, 10, 250)

plt.hist(x)
plt.show()
```



Matplotlib Pie Charts

- With Pyplot, you can use the pie() function to draw pie charts:

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

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

plt.pie(y, labels = mylabels)
plt.legend(title = "Four Fruits:")
plt.show()
```



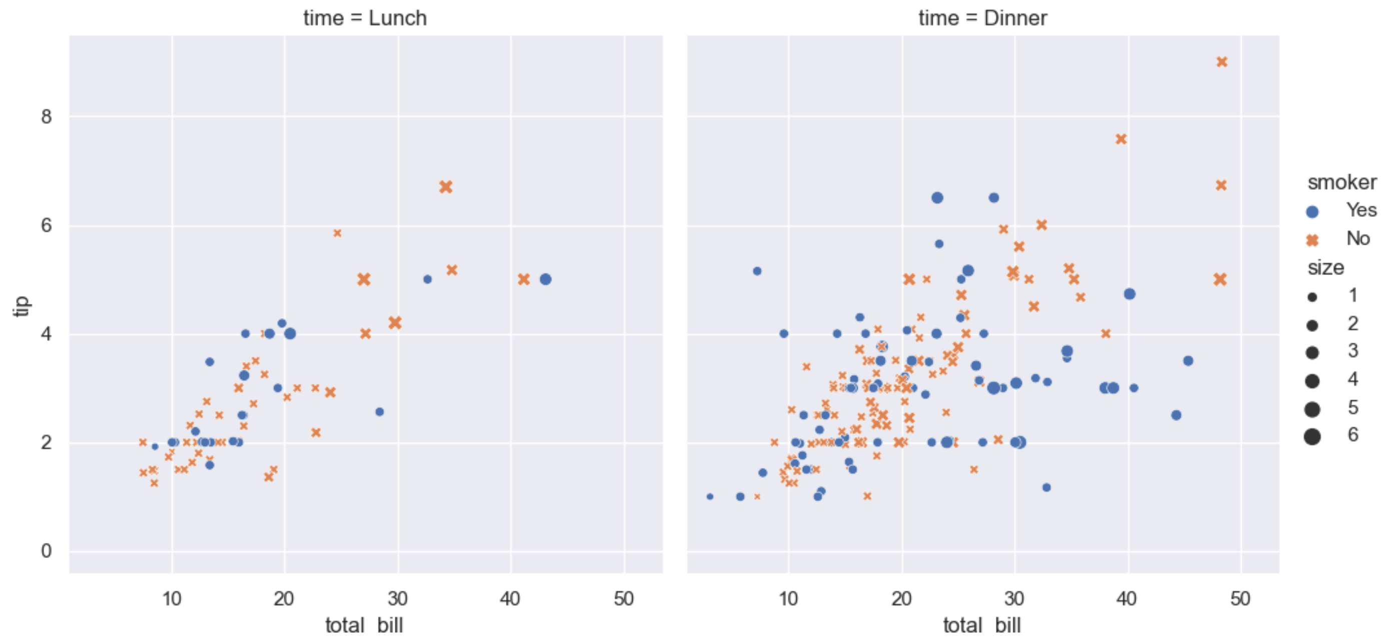
Introduction to Seaborn

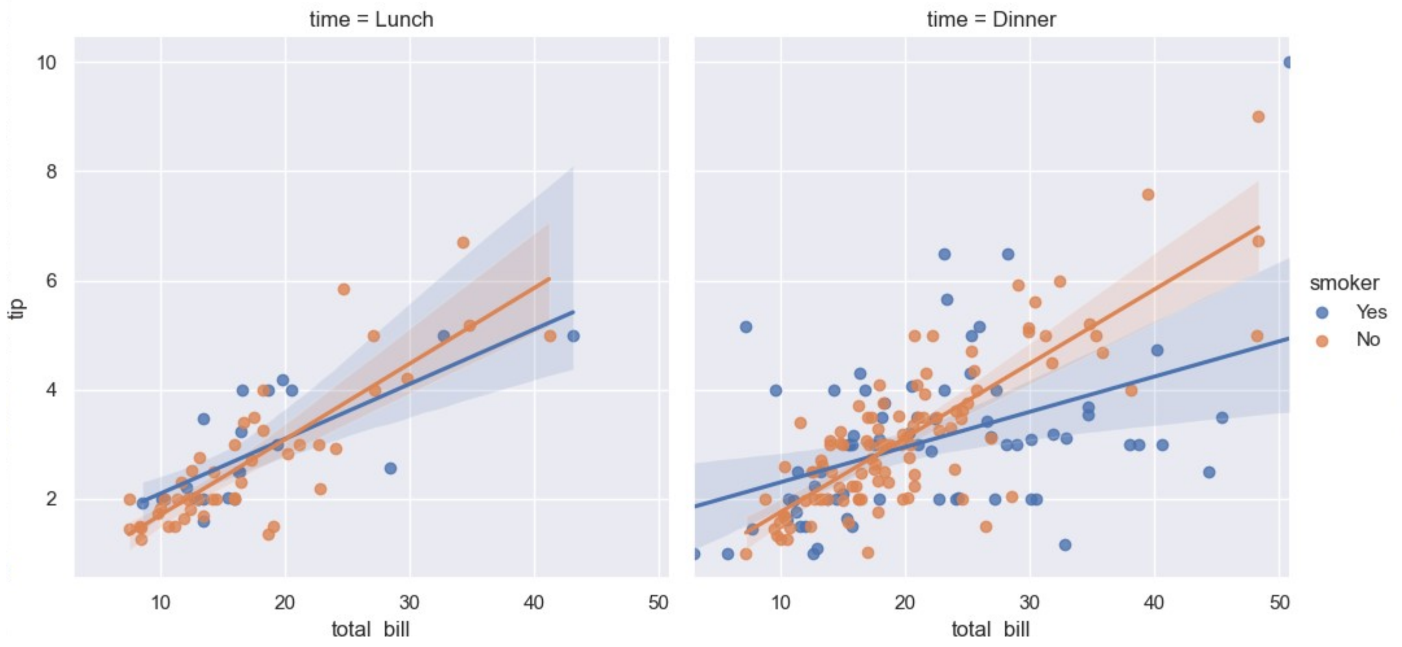
- Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.
- Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

seaborn

Seaborn

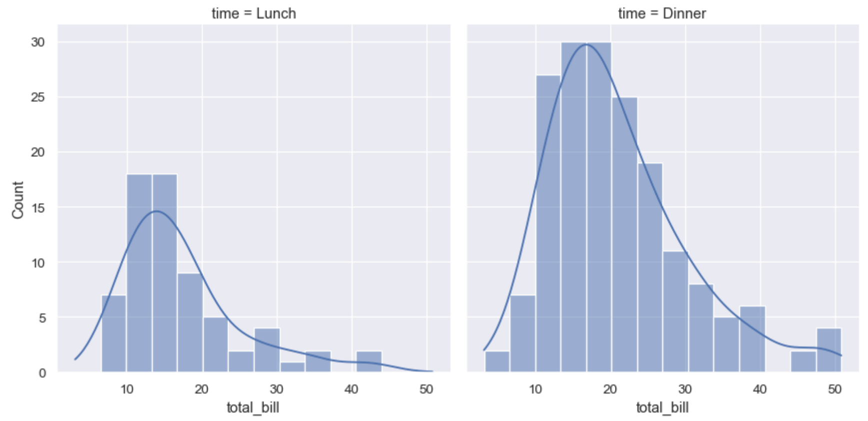
```
import matplotlib.pyplot as plt
# Import seaborn
import seaborn as sns

# Apply the default theme
sns.set_theme()

# Load an example dataset
tips = sns.load_dataset("tips")

# Create a visualization
sns.relplot(
    data=tips,
    x="total_bill", y="tip", col="time",
    hue="smoker", style="smoker", size="size",
)
plt.show()

sns.lmplot(
    data=tips,
    x="total_bill",
    y="tip", col="time", hue="smoker")
plt.show()
```





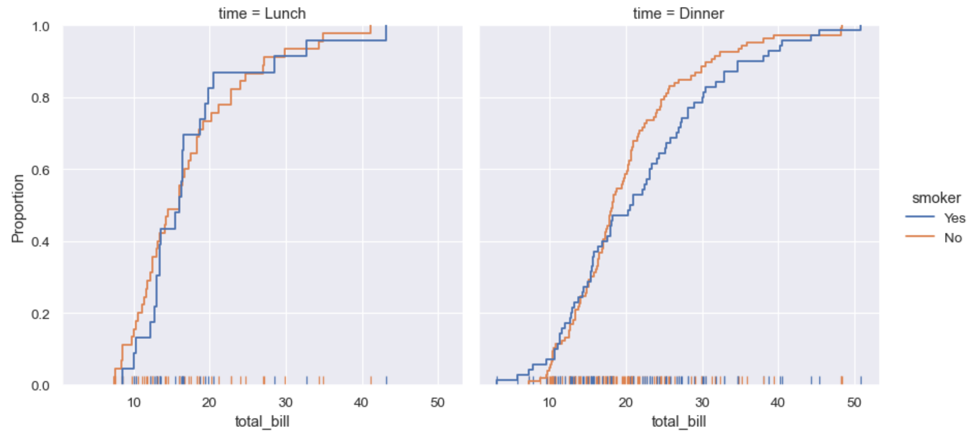
Seaborn →Learn more: http://seaborn.pydata.org/introduction.html

sns.displot(data=tips, x="total_bill", col="time", kde=True)



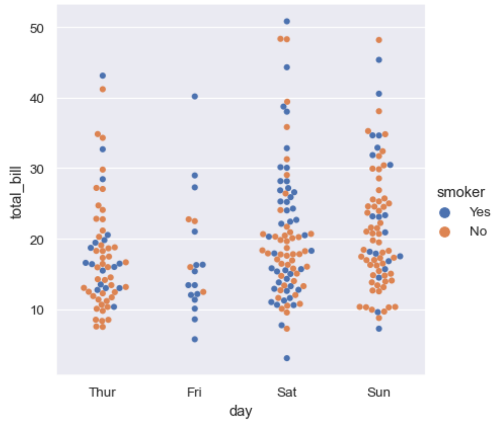
Histograms and computationally-intensive approaches like kernel density estimation

sns.displot(data=tips, kind="ecdf", x="total_bill", col="time", hue="smoker", rug=True)



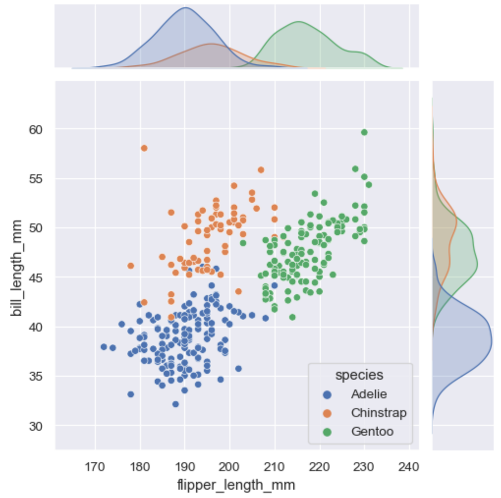
Calculating and plotting the empirical cumulative distribution function of the data

sns.catplot(data=tips, kind="swarm", x="day", y="total_bill", hue="smoker")



Drawing a “swarm” plot: a scatter plot that adjusts the positions of the points along the categorical axis

penguins = sns.load_dataset("penguins")
sns.jointplot(data=penguins, x="flipper_length_mm", y="bill_length_mm", hue="species")



jointplot() plots the joint distribution between two variables along with each variable’s marginal distribution.

Class 5’s Activity - Code Learning (10 points)

- Gather all the codes into .py or .ipynb files
- Run the codes and make sure they execute successfully
- Capture the images of the codes and the running results
- Submit the files and screen shots to E-Learning class assignment

Seaborn Exercises – 10 Points

- Finish the code in http://seaborn.pydata.org/introduction.html
- Gather all the code with comments into .py or .ipynb files
- Run the codes and make sure they execute successfully
- Capture the images of the codes and the running results
- Submit the files and screen shots to E-Learning class assignment