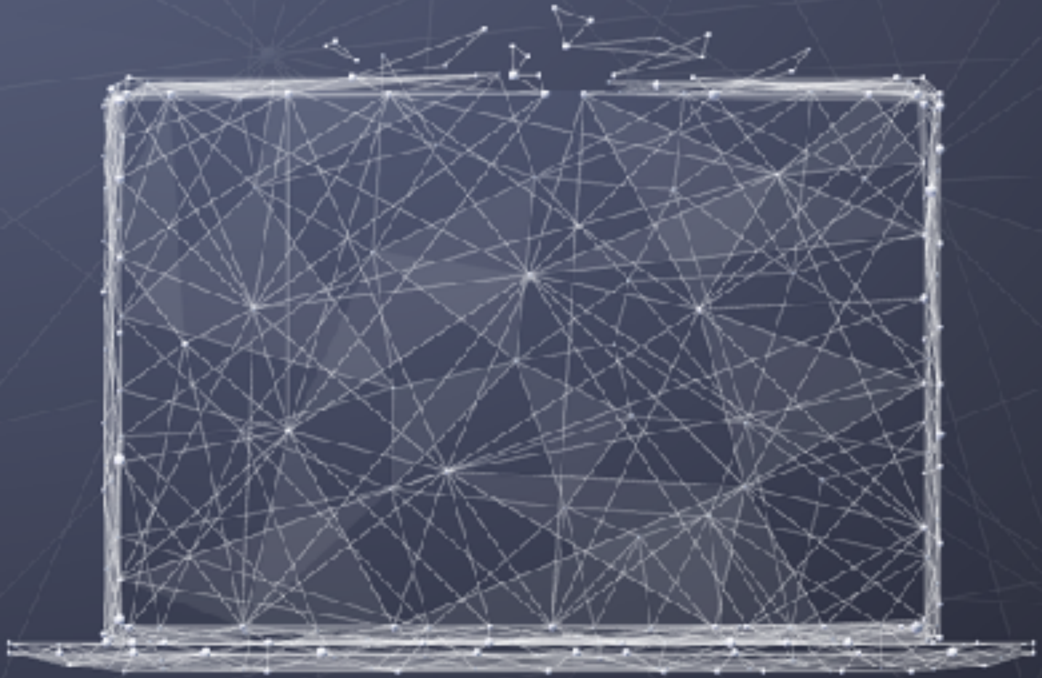
Data Science
Data Engineering I

Visualizing data



PURDUE UNIVERSITY | College of Science

Exploratory data analysis

- Maximize insight into data
- Uncover underlying structure
- Identify important variables
- Detect outliers and anomalies
- Test underlying modeling assumptions
- Generate hypotheses from data

Methods to summarize and visualize

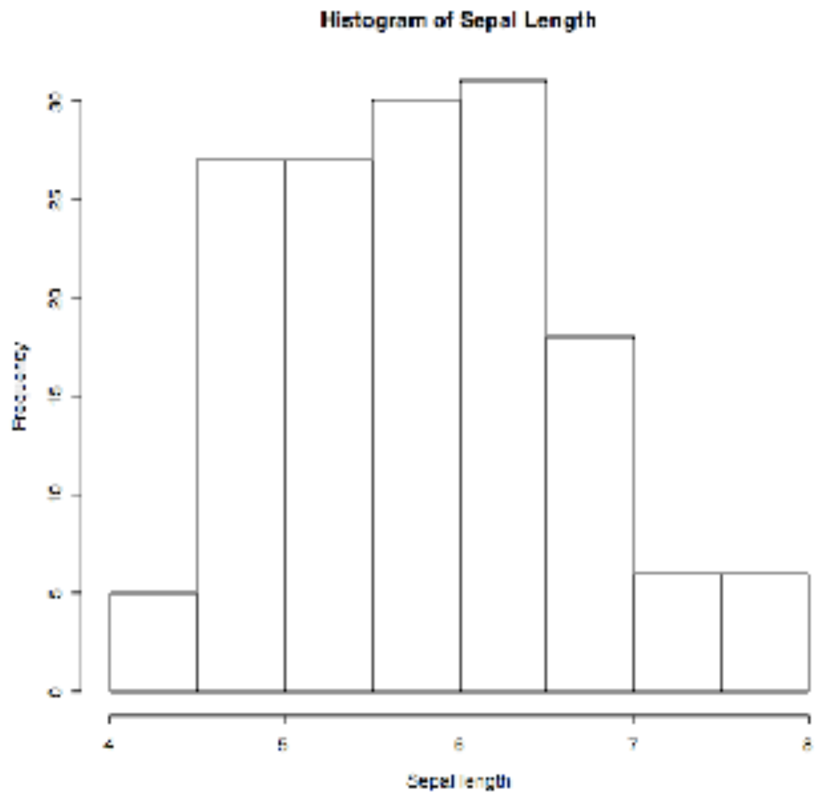
- Low-dimensional data
  - Summarizing data with simple statistics
  - Plotting raw data (1D, 2D, 3D)
- Higher-dimensional data
  - Principal component analysis
  - Multidimensional scaling

Data summarization

- Measures of location
  - Mean, median, quartiles, mode
- Measures of dispersion or variability
  - Variance, standard deviation, range, skew

Histograms (1D)

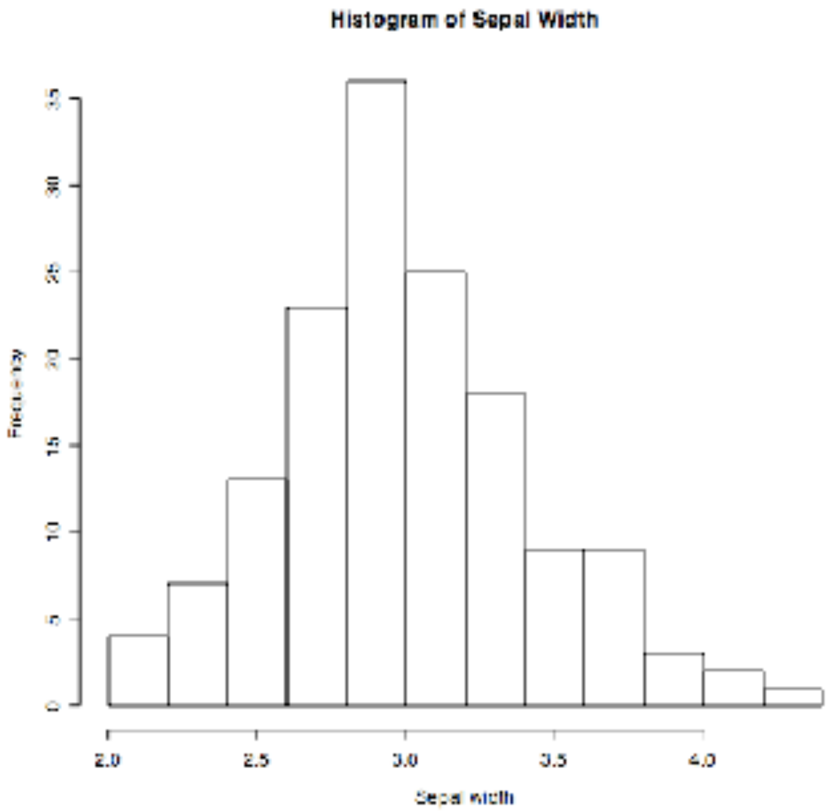
- Most common plot for univariate data
- Split data range into equal-sized bins, count number of data points that fall into each bin
- Graphically shows:
  - Center (location)
  - Spread (scale)
  - Skew
  - Outliers
  - Multiple modes



Example histogram

```
# import python plotting library
import matplotlib.pyplot as plt
plt.hist(data['sepal-width'])
```

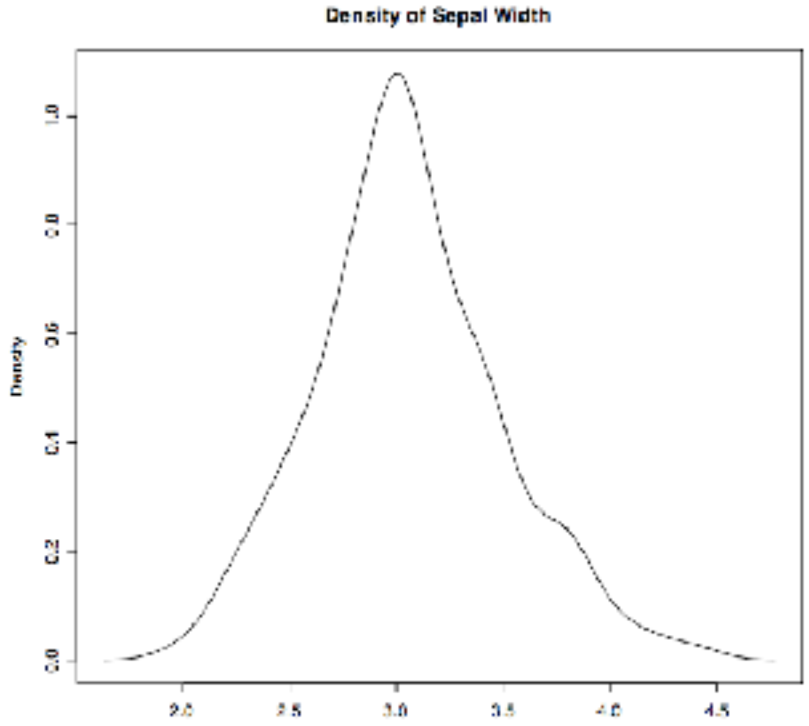
- Useful arguments:
  - bins: number of bins to use, default is equally spaced breaks
  - density: if True, y-axis will reflect probability instead of frequency counts



Histogram limitations

- Histograms can be misleading for small datasets
- Slight changes in the data or binning approach can result in different histograms
- Smoothed density plots may be a better choice

```
# pandas.DataFrame has plot functions too
data['sepal-width'].plot.kde()
```

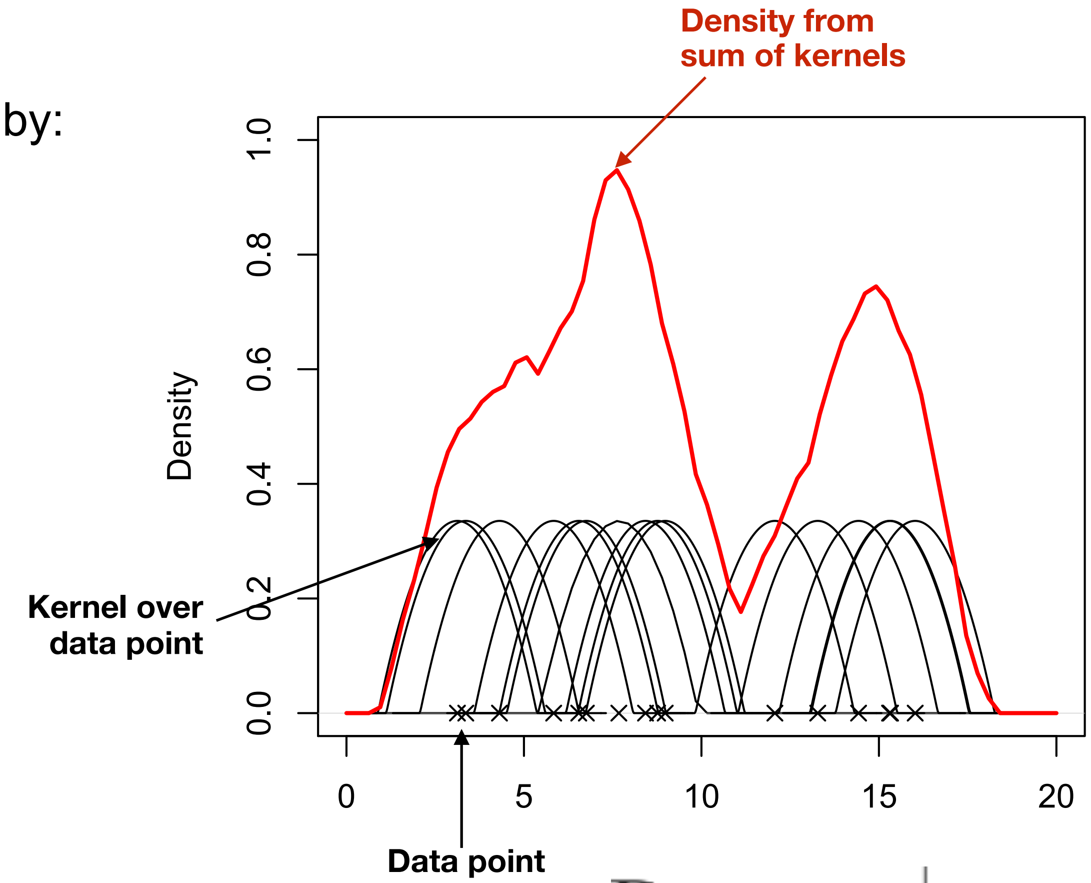


Density plots

- Density function estimates a continuous function from a discrete set of observations by:
  - Using a kernel function to estimate density at each point x,
  - Then pooling the information from neighboring points to estimate density
- Estimated density is:

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K\left(\frac{x - x(i)}{h}\right)$$

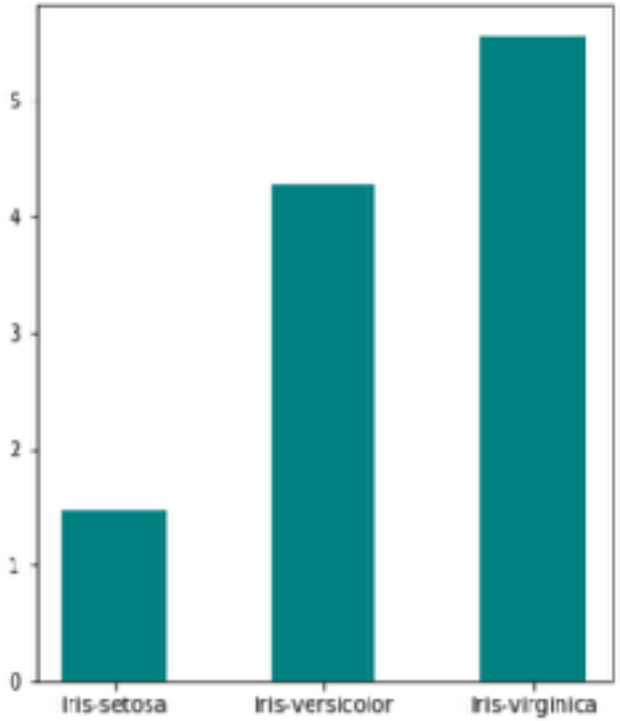
- Parameters: Kernel function K, bandwidth h



PURDUE UNIVERSITY. College of Science

Bar plots

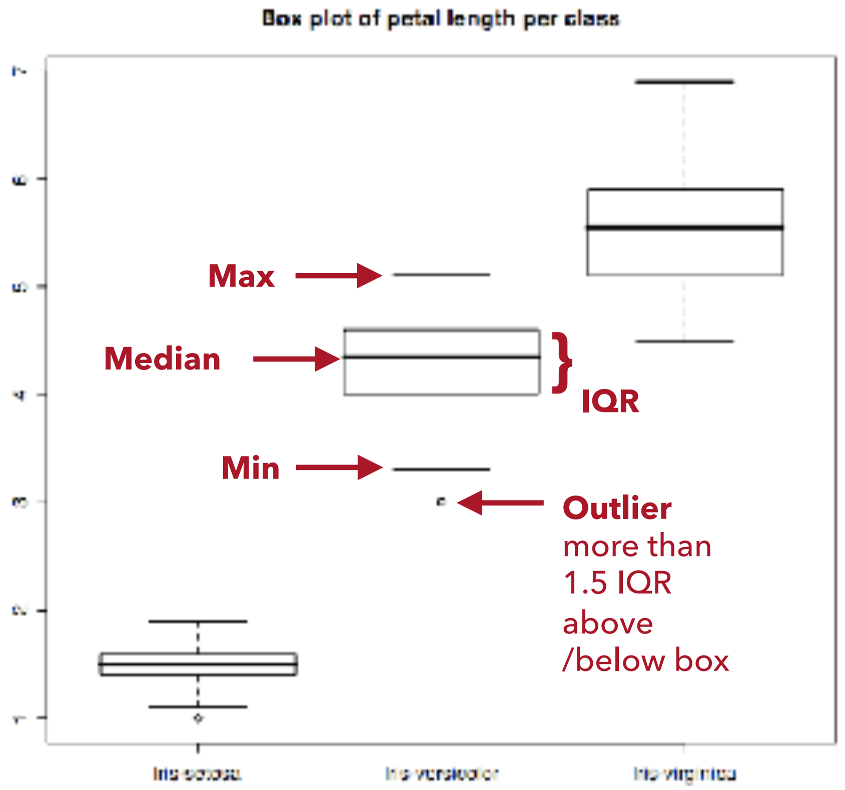
```
x1 = data[data.category=='Iris-setosa'][['petal-length']].mean()[0]
x2 = data[data.category=='Iris-versicolor'][['petal-length']].mean()[0]
x3 = data[data.category=='Iris-virginica'][['petal-length']].mean()[0]
barlabels = ['Iris-setosa','Iris-versicolor','Iris-virginica']
barvals = [x1,x2,x3]
plt.bar(barlabels, barvals)
```



Box plots (2D)

- Display relationship between discrete and continuous variables
- For each discrete value X, calculate quartiles and range of associated Y values

```
# boxplot takes a list of data vectors
# the distribution of values in vector
# is summarized by a box in the plot
plt.boxplot(dataGrps)
```



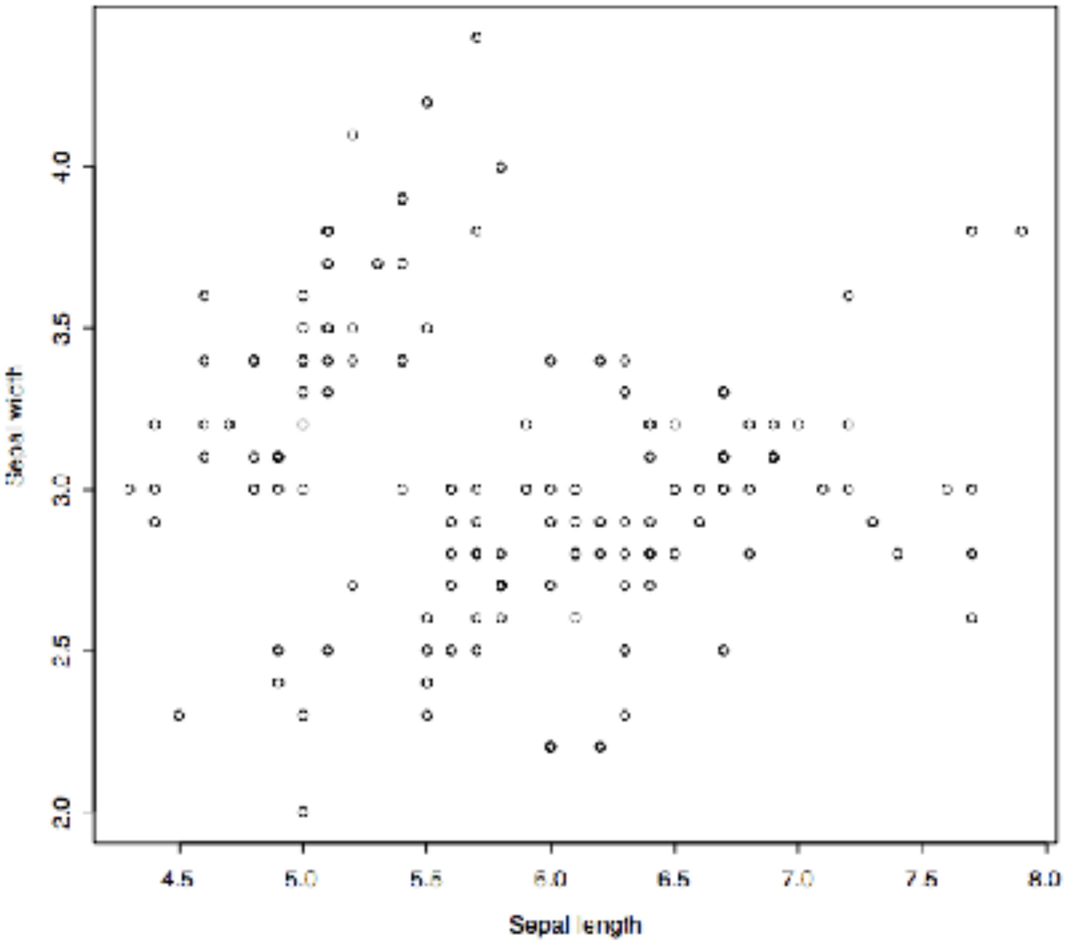
Outlier more than 1.5 IQR above /below box

Scatterplot (2D)

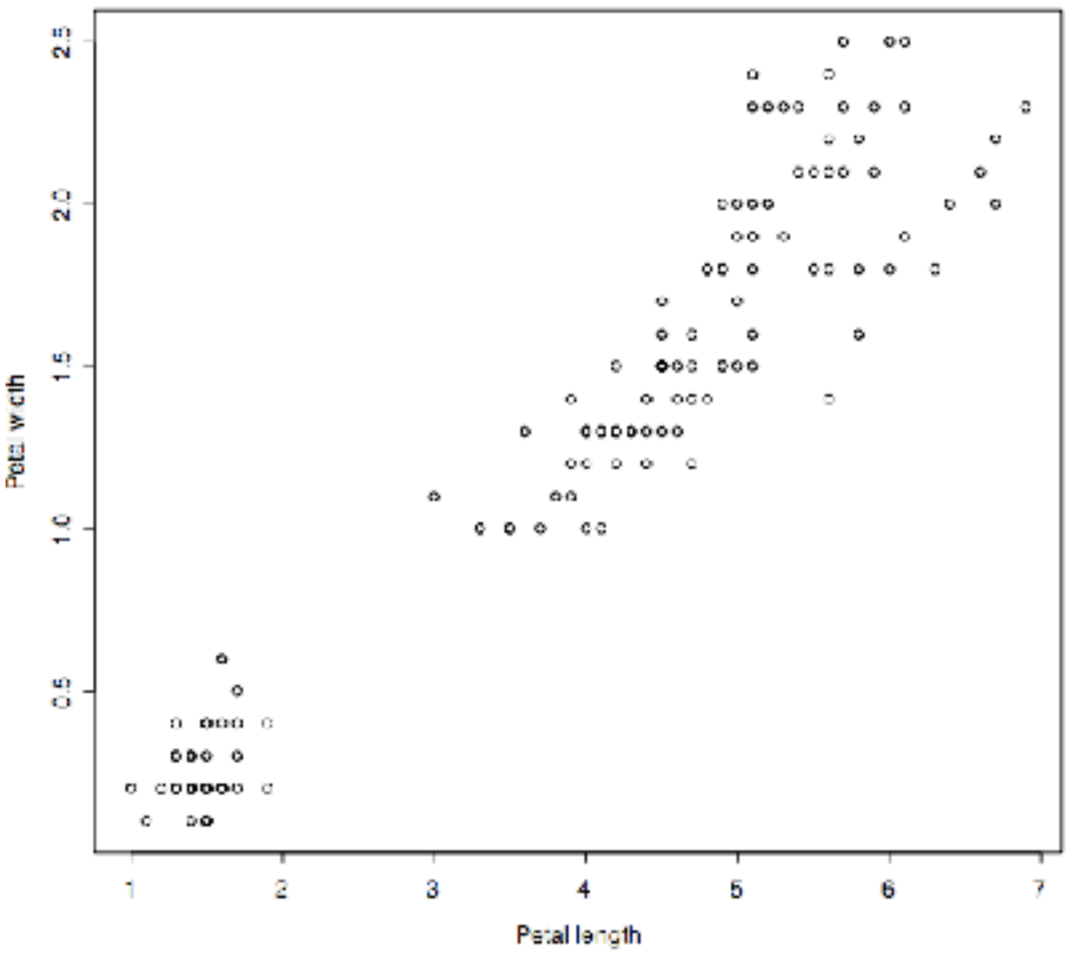
- Most common plot for bivariate data
  - Horizontal X axis: the suspected independent variable
  - Vertical Y axis: the suspected dependent variable
- Graphically shows:
  - If X and Y are related; Linear or non-linear relationship
  - If the variation in Y depends on X
  - Outliers

```
plt.scatter(data[‘sepal-length'],data['sepal-width'])
```

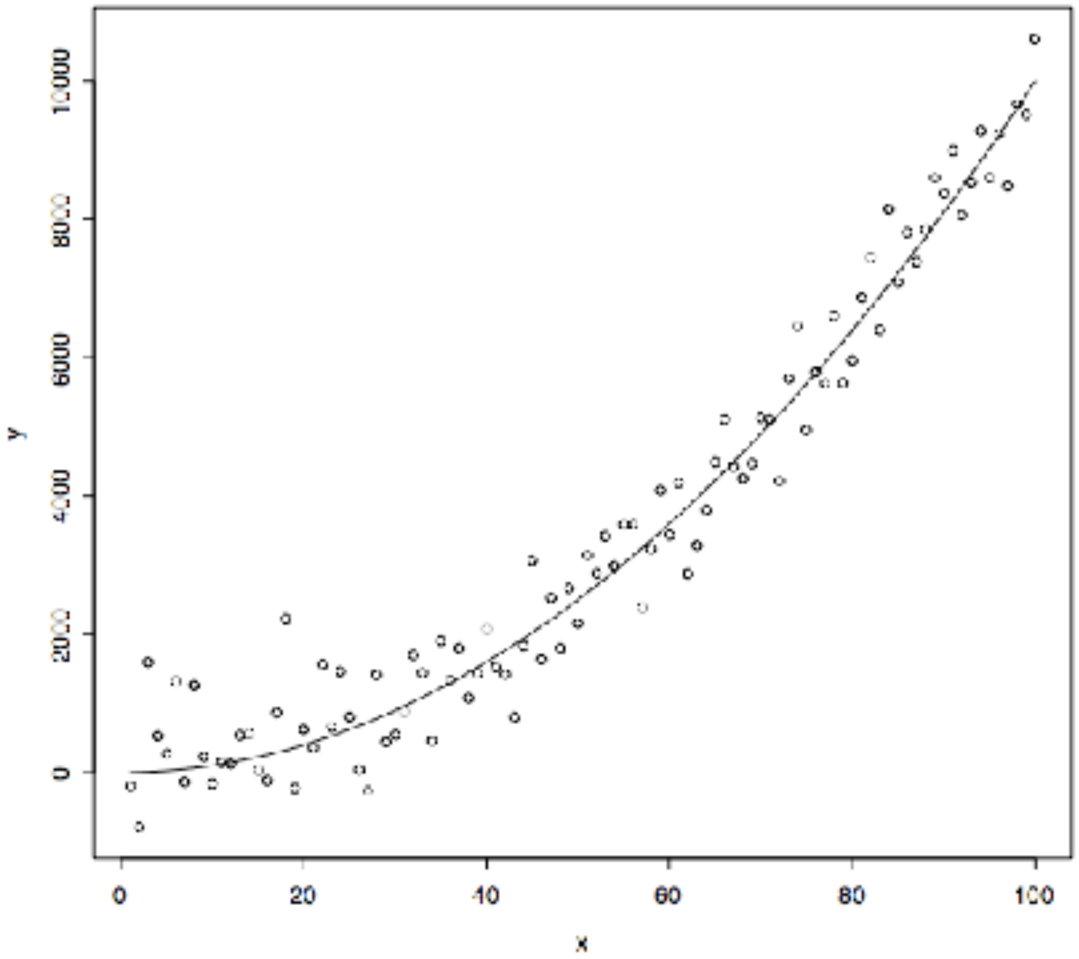
No relationship



Linear relationship



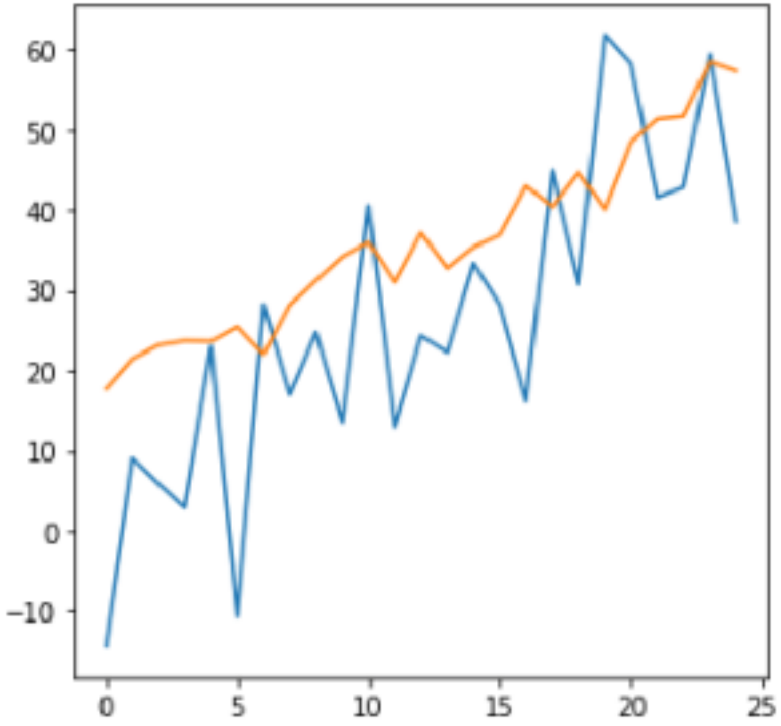
Non-linear relationship



Line graph (2D)

- Plot command can be used to make line graphs as well

```
# create data
import numpy as np
x = []
for i in range(25):
    x.append(i)
y = np.add(np.power(x,1.25),np.random.normal(0,10,25))
y2 = np.add(np.power(x,1.1) + 20,np.random.normal(0,3,25))

# plot line chart
plt.plot(x,y)
# overlay second line on same plot
plt.plot(x,y2)
```

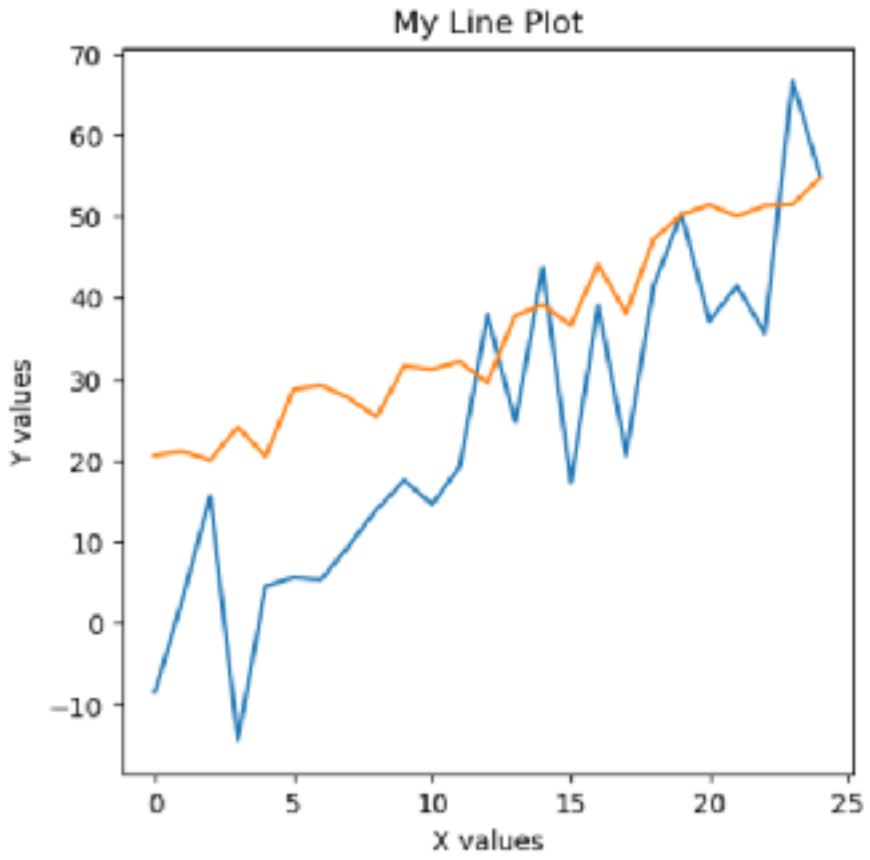


PURDUE UNIVERSITY | College of Science

Formatting plots

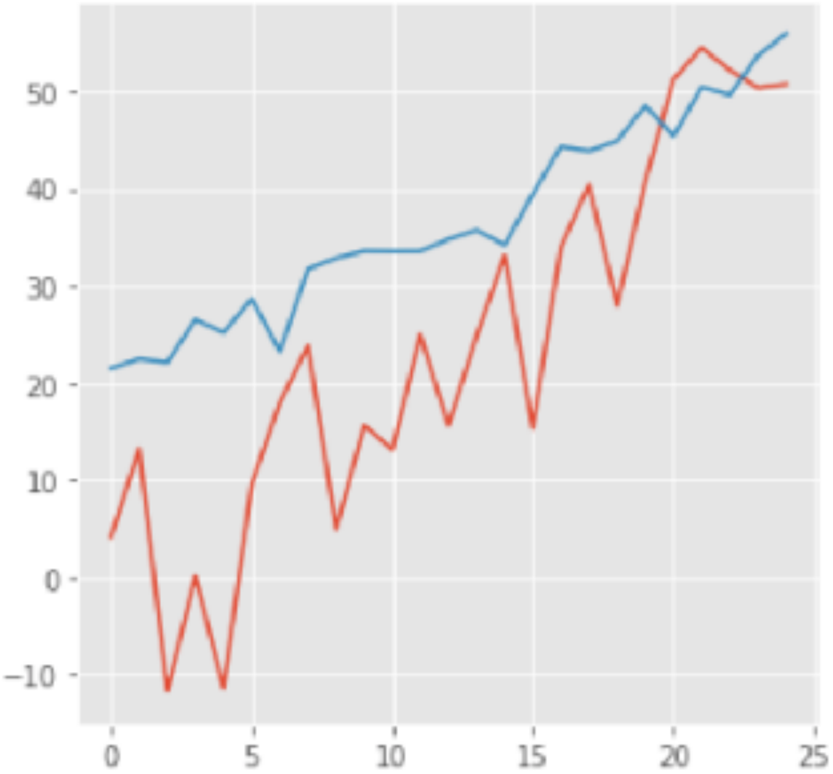
- Subplot command returns figure and axes objects that can be modified

```
fig, ax = plt.subplots()

fig.set_figheight(5)
fig.set_figwidth(5)

ax.set_title("My Line Plot")
ax.set_xlabel("X values")
ax.set_ylabel("Y values")
ax.plot(x,y)
ax.plot(x,y2)
```

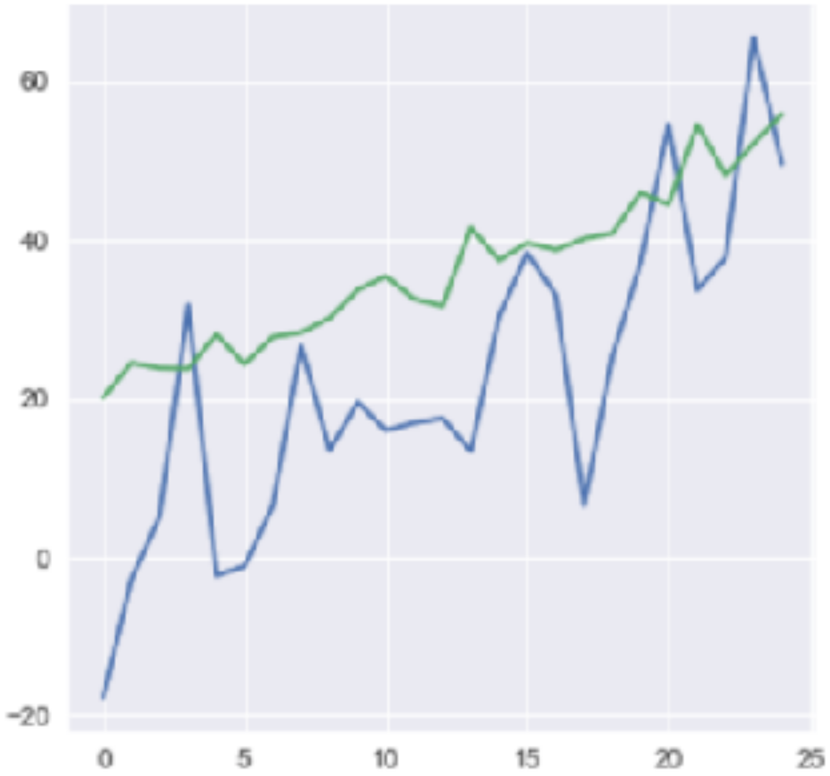


Formatting plots with style sheets

```
import matplotlib.pyplot as plt
plt.style.use(‘ggplot')
```



ggplot



seaborn