Introduction to Data Science

Lecture 9

Data Wrangling, preprocessing, and Visualization

Part II

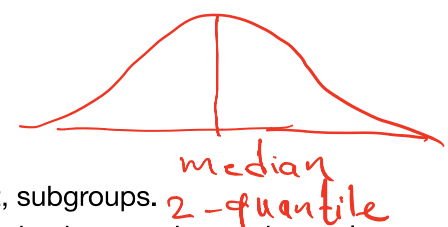
¶

Continued from preprocessing techniques

1. Data Preprocessing
    - dealing with outliers
    - some more features of Pandas
2. Data visualization methods

Dealing with outliers

Quantiles :



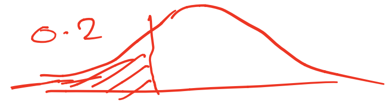
- A quantile is where a sample is divided into equal-sized, adjacent, subgroups.
- Quantiles are cut points dividing the range of a probability distribution into continuous intervals with equal probabilities

- 2-quantile (median) -The median is a quantile; the median is placed in a probability distribution so that exactly half of the data is lower than the median and half of the data is above the median. The median cuts a distribution into two equal areas

Difference between quantiles, quartiles, percentiles and deciles

- Quartiles are also quantiles; they divide the distribution into four equal parts.
- Percentiles are quantiles that divide a distribution into 100 equal parts
- Deciles are quantiles that divide a distribution into 10 equal parts.b

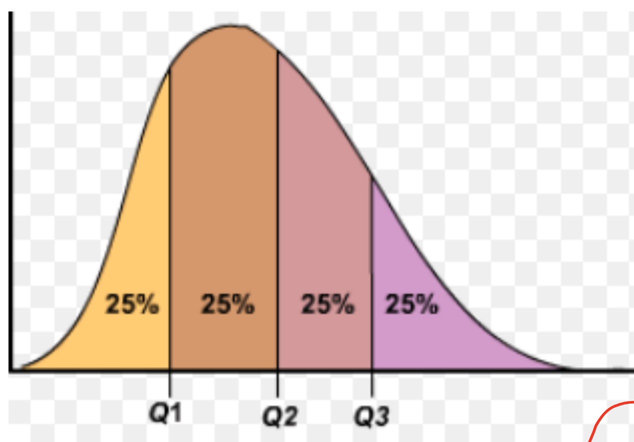
0.2

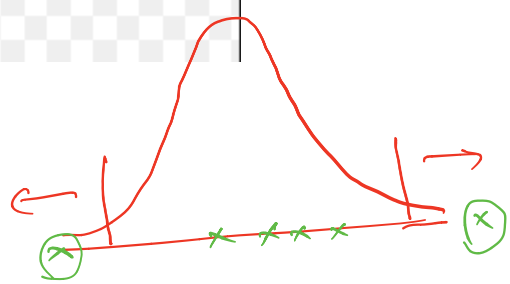


How to Find Quantiles?

- Example: Find the number in the following set of data where 20 percent of values fall below it, and 80 percent fall above: 1 3 5 6 9 11 12 13 19 21 22 32 35 36 45 44 55 68 79 80 81 88 90 91 92 100 112 113 114 120 121 132 145 146 149 150 155 180 189 190
- Step 1: Order the data from smallest to largest. The data in the question is already in ascending order.
- Step 2: Count how many observations you have in your data set. this particular data set has 40 items. N = 40
- Step 3: Convert any percentage to a decimal for "q". We are looking for the number where 20 percent of the values fall below it, so convert that to .2. 0.2
- Step 4: Insert your values into the formula:
  - ith observation = q (n + 1) → ith 0.2(40+1)
  - ith observation = .2 (40 + 1) = $\boxed{8.2}$
- The ith observation is at 8.2, so we round down to 8 (remembering that this formula is an estimate). The 8th number in the set is $\boxed{13}$, which is the number where 20 percent of the values fall below it.

SRC (https://www.statisticshowto.datasciencecentral.com/quantile-definition-find-easy-steps/)





How to find outliers:

- Using the Interquartile Range(IQR)
- Low outliers = Q1 – 1.5(Q3 – Q1) = Q1 – 1.5(IQR)
- High outliers = Q3 + 1.5(Q3 – Q1) = Q3 + 1.5(IQR) Where:
- Q1 = first quartile
- Q3 = third quartile
- IQR = Interquartile range
  - An outlier is defined as being any point of data that lies over 1.5 IQRs below the first quartile (Q1) or above the third quartile (Q3)in a data set.
  - High = (Q3) + 1.5 IQR
  - Low = (Q1) – 1.5 IQR

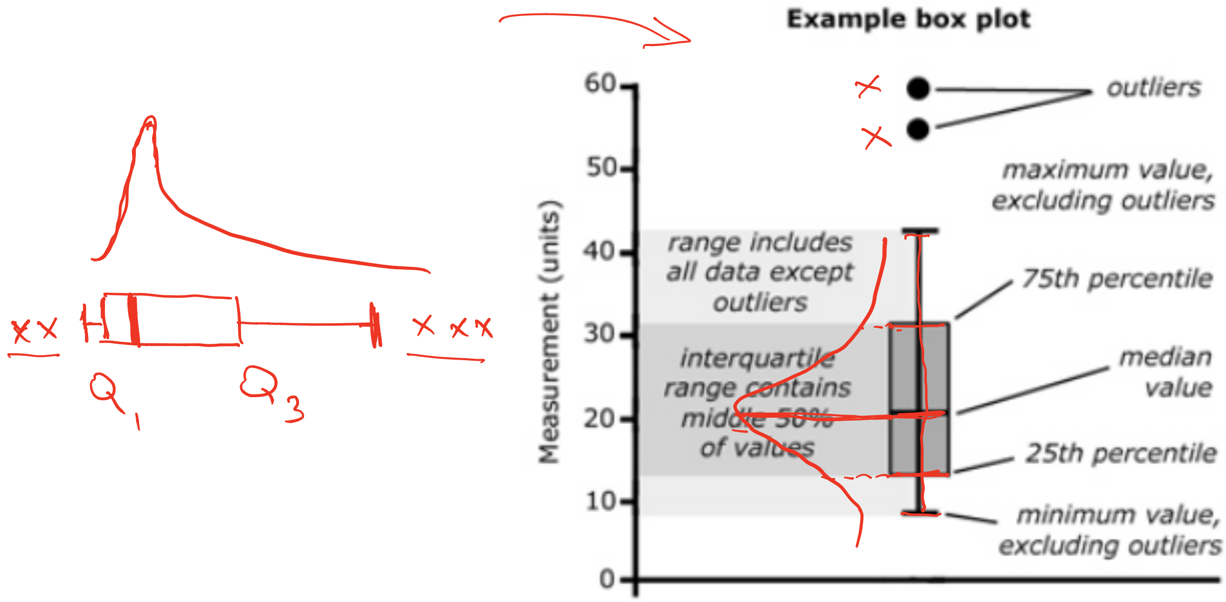
P <<< 1

1.5 → 3

Sample Question: Find the outliers for the following data set: 3, 10, 14, 22, 19, 29, 70, 49, 36, 32.

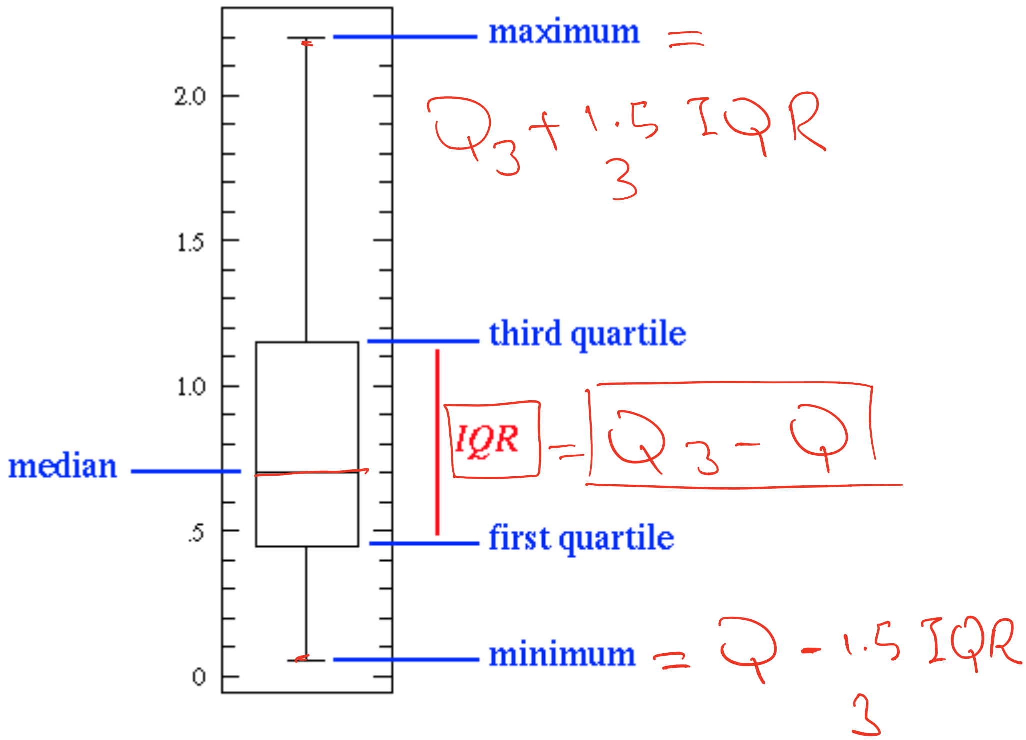
- Step 1: Find the IQR, Q1(25th percentile) and Q3(75th percentile).
  - IQR = 22
  - Q1 = 14
  - Q3 = 36
- Step 2: Multiply the IQR you found in Step 1 by 1.5:
  - IQR *1.5 = 22* 1.5 = 33.
- Step 3: Add the amount you found in Step 2 to Q3 from Step 1:
  - 33 + 36 = 69.
- Step 3: Subtract the amount you found in Step 2 from Q1 from Step 1:
  - 14 – 33 = -19.
  - This is your lower limit. Set this number aside for a moment.

Find outliers using boxplots



- A point that is outside the lower whisker is considered a mild outlier while an extreme outlier is one that is beyond the upper whisker.

How to read a boxplot and find outliers



How to detect and remove outliers using Pandas

In [29]:

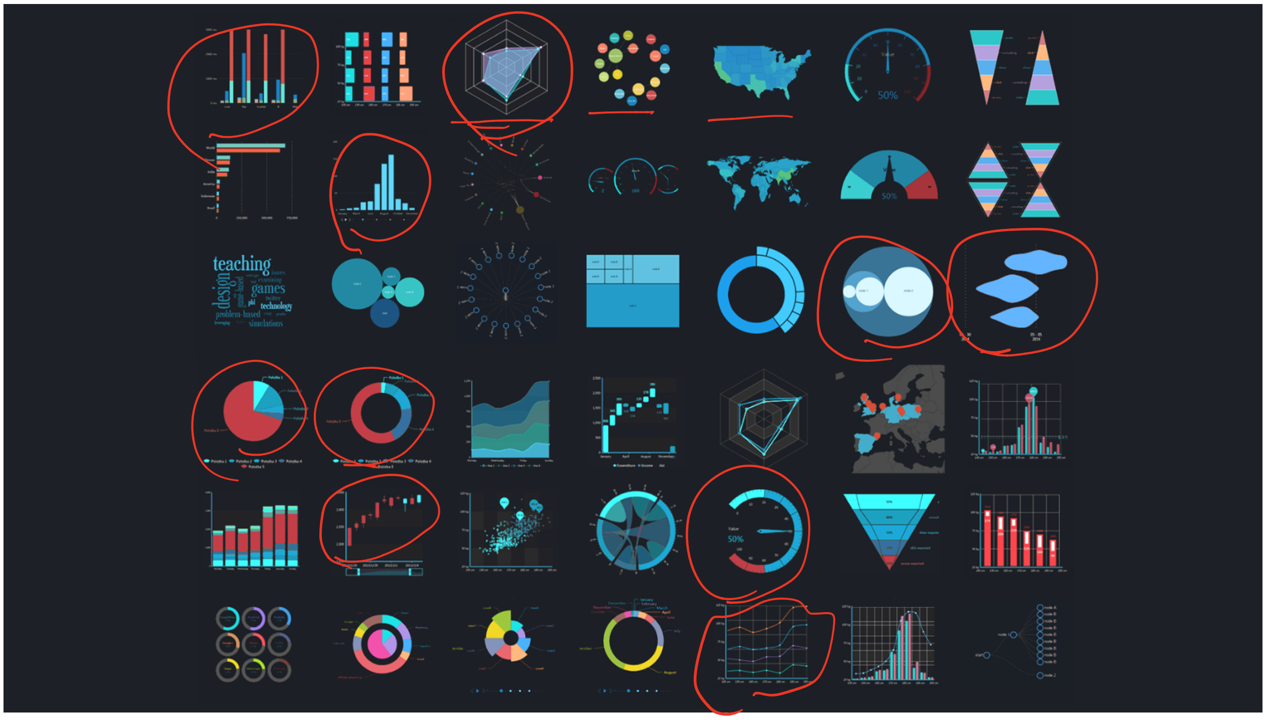
```
np.random.seed(33454)
stepframe = pd.DataFrame({'a': np.random.randint(1, 200, 20),
                          'b': np.random.randint(1, 200, 20),
                          'c': np.random.randint(1, 200, 20)})

stepframe[stepframe > 150] *= 10
print (stepframe)

Q1 = stepframe.quantile(0.25)
Q3 = stepframe.quantile(0.75)
IQR = Q3 - Q1
print('Q1=', Q1)
df = stepframe[~((stepframe < (Q1 - 1.5 * IQR)) |(stepframe > (Q3 + 1.5

print (df)
```

[1, 200]

150x10

```
       a     b     c
0      4  1970    79
1    109    50   124
2   1570    87    41
3    137    60  1990
4     19   138   100
5     86    83   143
6     55    23    58
7     78   145    18
8    132    39    65
9     37   146  1970
10  1640    55  1560
11    32  1540  1570
12   132  1950    44
13    67   148  1880
14    95  1730    52
15   124   102    21
16    93    61    56
17    84    21    25
18    21  1650  1630
19    34    52   126
Q1= a    36.25
b    54.25
c    50.00
Name: 0.25, dtype: float64
      a    b     c
1   109   50   124
3   137   60  1990
4    19  138   100
5    86   83   143
6    55   23    58
7    78  145    18
8   132   39    65
9    37  146  1970
13   67  148  1880
15  124  102    21
16   93   61    56
17   84   21    25
19   34   52   126
```

Data Visualization Methods

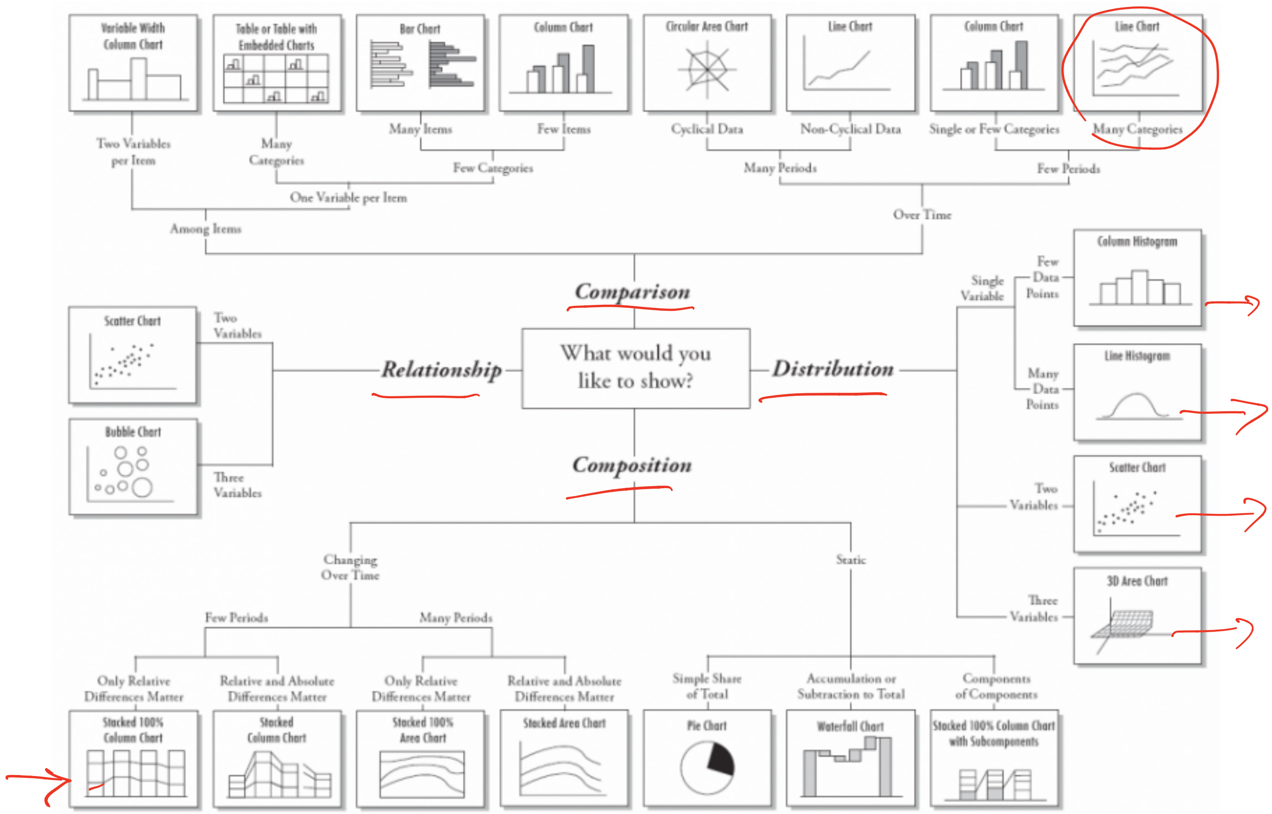


First you need to ask this question what would you like to show

1. Relationship
2. Comparison
3. Composition
4. Distribution

Examaples :

- Bar chart
- Histogram
- Scatter plots
- Network
- Heatmap
- Venn diagram
- Pie chart
- Time Series
- Dendrogram
- Ringchart
- boxplot
- Gantt Chart
- Word Cloud-
- Bubble Cloud/Chart
- Radar/Spider Chart
- Violin Plots



src (https://eazybi.com/blog/data_analysis_and_visualization/)

The Best Python Data Visualization Libraries

SRC (https://www.fusioncharts.com/blog/best-python-data-visualization-libraries/)

1. Matplotlib
2. Many other libraries are built on top of Matplotlib and are designed to work in conjunction with analysis, it being the first Python data visualization library. The versatility of Matplotlib can be used to make many visualization types:-
    - Scatter plots
    - Bar charts and Histograms
    - Line plots
    - Pie charts
    - Stem plots
    - Contour plots
    - Quiver plots
    - Spectrograms
3. seaborn
    - Seaborn is a popular data visualization library that is built on top of Matplotlib. Seaborn’s default styles and color palettes are much more sophisticated than Matplotlib. Beyond that, Seaborn is a higher-level library, meaning it’s easier to generate certain kinds of plots, including heat maps, time series, and violin plots.
4. ggploy

- Ggplot is a python visualization library based on R's ggplot2 and the Grammar of Graphics. It lets you construct plots using high-level grammar without thinking about the implementation details.

5. Bokeh
6. plotly
   - interactive and can access from notebook
7. Pygal
   - like Plotly and Bokeh, offers interactive plots that can be embedded in a web browser.
8. Geoplotlib
9. Missingno
   - good when have missing data
10. Leather
    - designed to work with all data types and produces charts such as SVGs, so that they can be scaled without losing image quality

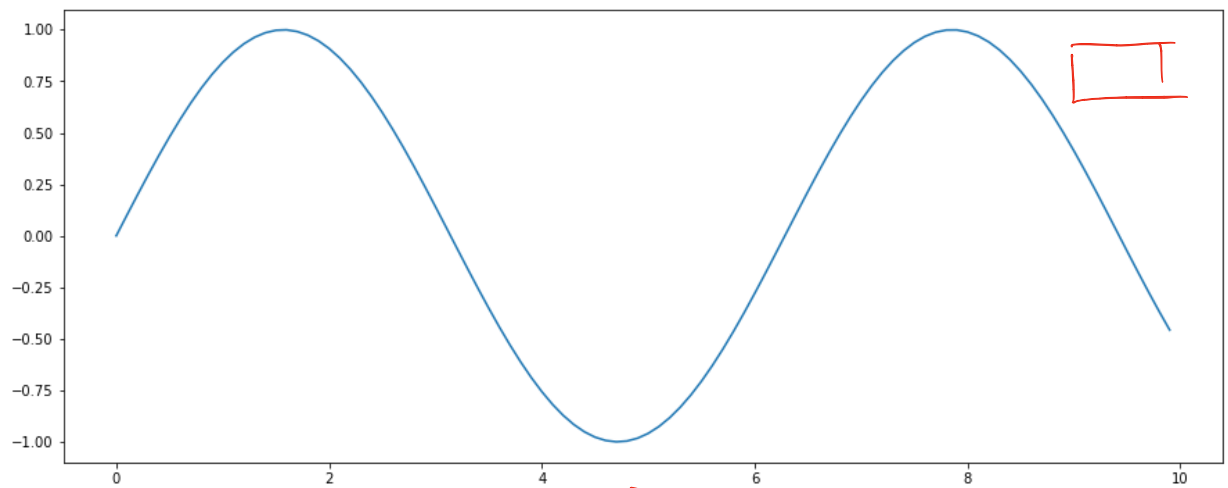
Introduction to Matplotlib

The matplotlib provides a context, one in which one or more plots can be drawn before the image is shown or saved to file. The context can be accessed via functions on pyplot. The context can be imported as follows:

Line plots

In [18]:

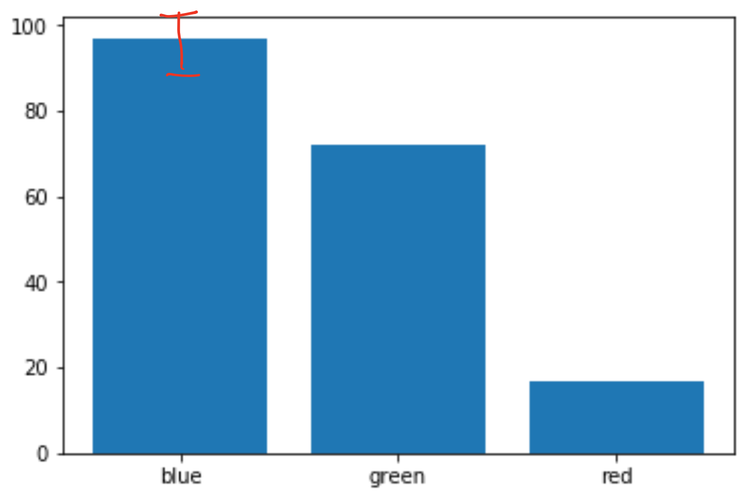
```
#There is some convention to import this context and name it plt;
# for example:


# example of a line plot
from numpy import sin
from matplotlib import pyplot
# consistent interval for x-axis
x = [x*0.1 for x in range(100)]
# function of x for y-axis
y = sin(x)
# create line plot
pyplot.plot(x, y)
# show line plot
pyplot.show()

```



Bar Plots

In [3]:

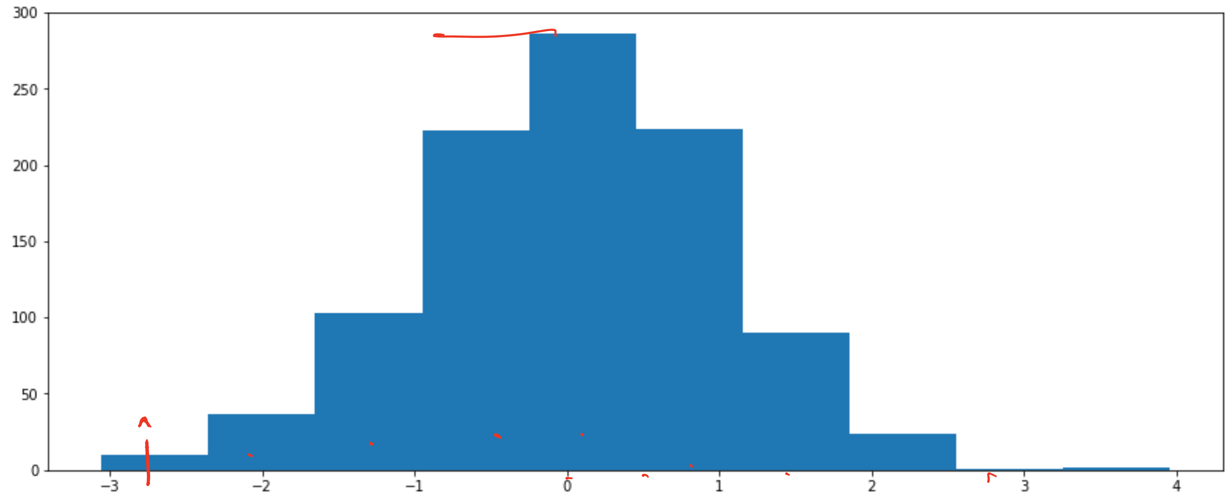
```
# example of a bar chart
from random import seed
from random import randint
from matplotlib import pyplot
# seed the random number generator
seed(1)
# names for categories
x = ['red', 'green', 'blue']
# quantities for each category
y = [randint(0, 100), randint(0, 100), randint(0, 100)]
# create bar chart
pyplot.bar(x, y)
# show line plot
pyplot.show()
```



Histograms

In [20]:

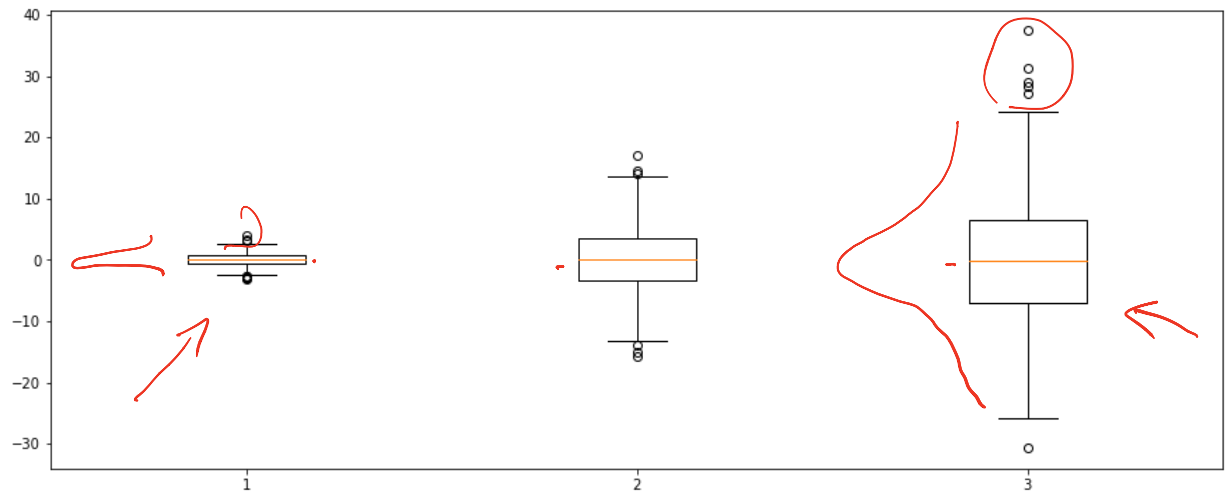
```
# example of a histogram plot
from numpy.random import seed
from numpy.random import randn
from matplotlib import pyplot
# seed the random number generator
seed(1)
# random numbers drawn from a Gaussian distribution
x = randn(1000)
# create histogram plot
pyplot.hist(x)
# show line plot
pyplot.show()
```



Box plots

In [24]:

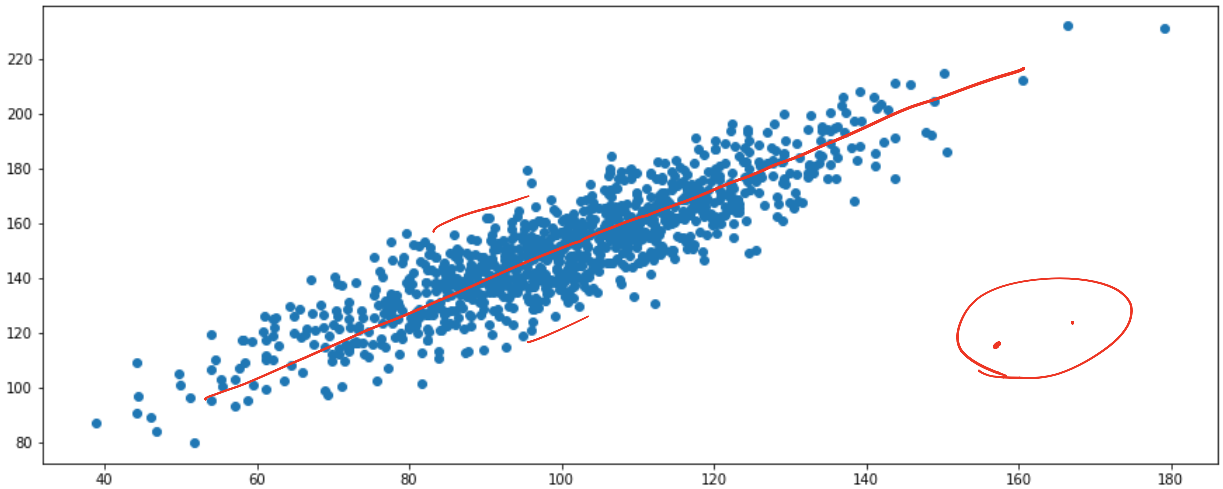
```python
# example of a box and whisker plot
from numpy.random import seed
from numpy.random import randn
from matplotlib import pyplot
# seed the random number generator
seed(1)
# random numbers drawn from a Gaussian distribution
x = [randn(1000), 5 * randn(1000), 10 * randn(1000)]
# create box and whisker plot
pyplot.boxplot(x)
# show line plot
pyplot.show()

```



Scatter plots

In [25]:

```
# example of a scatter plot
from numpy.random import seed
from numpy.random import randn
from matplotlib import pyplot
# seed the random number generator
seed(1)
# first variable
x = 20 * randn(1000) + 100
# second variable
y = x + (10 * randn(1000) + 50)
# create scatter plot
pyplot.scatter(x, y)
# show line plot
pyplot.show()
```



In [38]:

```
16000**2/1000000
```

Out[38]: 256.0

In [7]:

```
import scipy.io
mat = scipy.io.loadmat('CSE391_classificatin_project.mat')
print(mat)
```

```
{'__header__': b'MATLAB 5.0 MAT-file, Platform: MACI64, Created on: Fri D
ec  8 12:48:24 2017', '__version__': '1.0', '__globals__': [], 'X': array
([[10.42699064, 10.84348582, 11.09219389, ..., 11.07023837,
        11.74735081, 10.50401063],
       [ 9.8347583 ,  9.35144039,  9.41664766, ...,  9.48368883,
         9.30421258, 10.26172144],
       [ 8.07588503,  7.14721816,  8.22062065, ...,  7.43532022,
         8.03227775,  8.43868957],
       ...,
       [ 5.41119572,  3.3447859 ,  3.87012795, ...,  4.04829723,
         4.59012292,  3.37591521],
       [ 8.30711472,  9.514704  ,  9.01987062, ...,  9.60936511,
         9.34818881,  3.1990441 ],
       [ 7.83440658,  8.08174439,  7.39368562, ...,  6.75724865,
         7.84346509,  7.42367061]]), 'Y': array([[22.23870362, 24.4442072
8, 23.55802972, ..., 18.89239317,
        21.24972313, 16.77980121],
       [18.76071917, 22.93459431, 21.87983823, ..., 20.30949973,
        20.4313542 , 20.01656347],
       [15.36141305, 15.92780507, 16.13747958, ..., 15.26794821,
        15.44358468, 14.85321791],
       ...,
       [11.17456631, 18.62329949, 14.56388422, ...,  9.31736903,
        14.87404158, 17.81747236],
       [15.5601237 , 20.83243755, 14.18532272, ..., 24.1354336 ,
        21.02309686, 14.06337224],
       [19.99682363, 14.39784469, 21.29521597, ..., 18.67744827,
        18.40020678, 18.94083646]])}
```

In [ ]:

```

```