Kmit

SONET

Data Science

(Level-I)

Python: Descriptive Statistics

Statistics

- Statistics plays very important role in Data Science
- All the required algorithms for analysis are available in statistics
- Data representation in graphical and numeric is handled in statistics

Statistics

- Why Statistics?
- Definition
- Types of Statistics
- Key-Terms

Why Statistics?

- Everyone is using statistics unknowingly.
- Ex: to buy a mobile… we may compare…

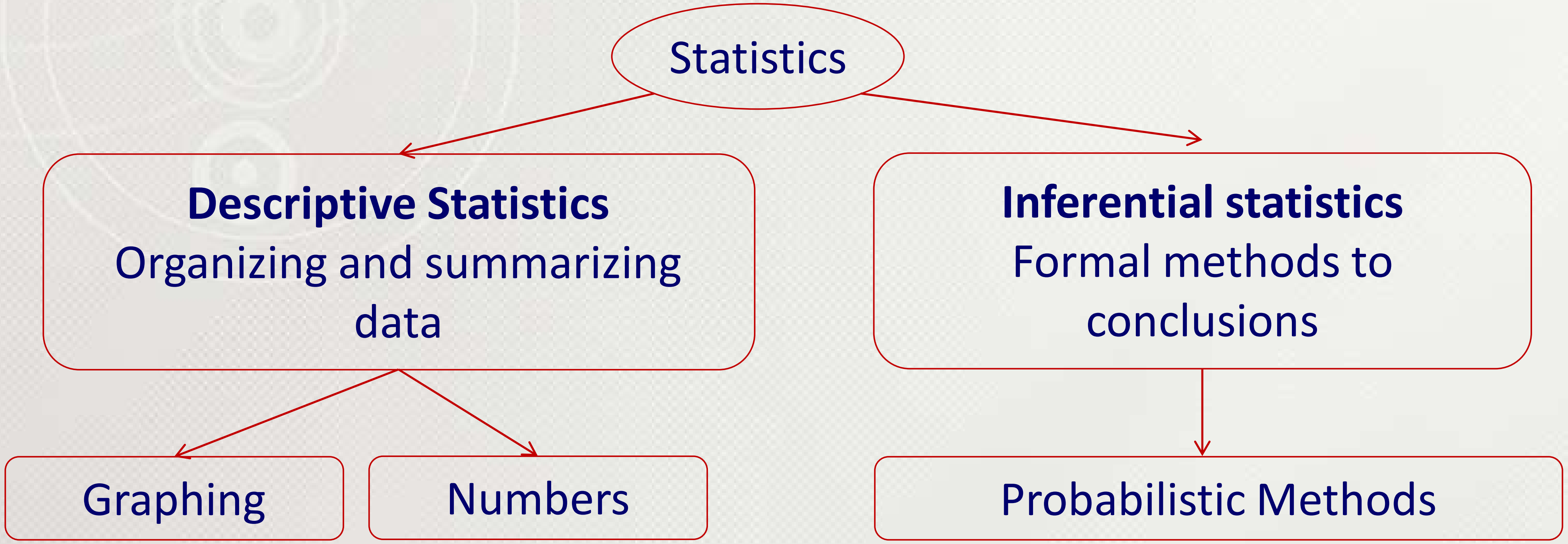
  cost Performance Warranty Etc…
- We can find statistics in (almost everywhere):

  News paper Television Sports Stock Market

- Sources provide sample information
  - We can understand and take decision
- It is the body of methods for making **wise decisions** in the face of uncertainty on the basis of numerical data and calculated risks.

Statistics

- The science of Numbers that deals with the collection, analysis, interpretation and presentation of data.



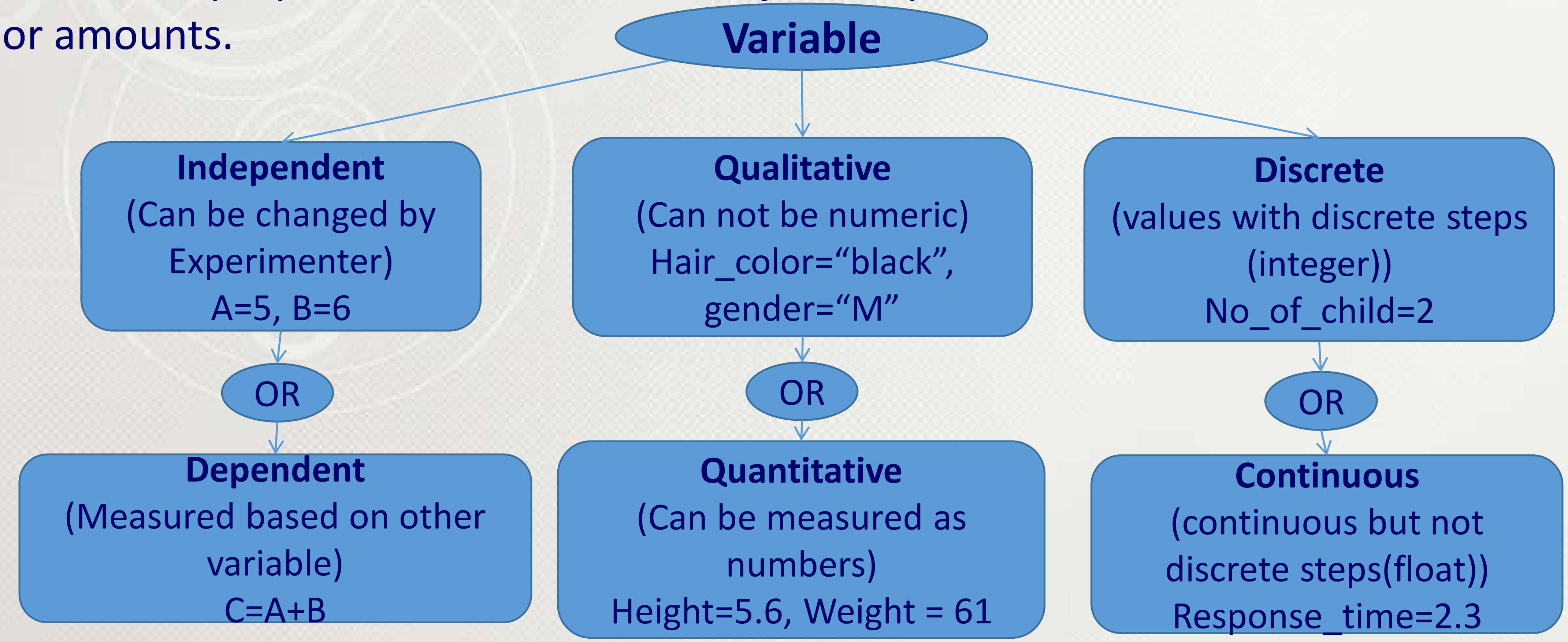
Descriptive Statistics

- Deals with Organizing and summarizing data.
- used when we know all of the values in order to describe a phenomenon

- There are 3 steps
    - Describing Data
    - Summarize / Analyze
    - Visualize

Descriptive statistics

- Variables : properties of some event, object, or person that can take on different values or amounts.



Variable
Independent
(Can be changed by Experimenter)
A=5, B=6
OR
Dependent
(Measured based on other variable)
C=A+B
Qualitative
(Can not be numeric)
Hair_color="black", gender="M"
OR
Quantitative
(Can be measured as numbers)
Height=5.6, Weight = 61
Discrete
(values with discrete steps (integer))
No_of_child=2
OR
Continuous
(continuous but not discrete steps(float))
Response_time=2.3

- Used for conducting analysis on one variable at a time or univariate Analysis

Kmit

SONET

Machine Learning

Descriptive statistics

The variable can be considered as data set.

X={54,56,34,44,34,65}

Common Approaches to analyze data set:

Central Tendency

"in the middle" or "popular" value in data.

Mean

Median

Mode

"most frequent observation"

most common height

most common income

most common size of shoe

Variability

How "far" away from the mean or median

how "spread out" are the data

Range

Variance

Quartile

Standard Deviation

Central Tendency

Mean: add all values and then divide by the total number of values.

Marks scored by a student in 5 subjects : 56, 31, 56, 8 and 32

Mean = $$\frac{56+31+56+8+32}{5} = \frac{183}{5} = 36.60$$

Mean $$\bar{X} = \frac{X_1 + X_2 + \ldots + X_{N-1} + X_N}{N}$$ = > $$\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i$$

Calculating the mean in Python :

```
>>> from __future__ import division
>>> (56+31+56+8+32)/5
36.6
>>> x = [56, 31, 56, 8, 32]
>>> print sum(x)/len(x)
36.6
```

Numpy provides a builtin function mean()

```
>>> import numpy as np
>>> x = np.array([56, 31, 56, 8, 32])
>>> print np.mean(x)
36.6
```

Central Tendency

- **Median:** The median of a set of observations is just the middle value(50 percentile).

- Make all the elements in ascending or descending order
- Marks scored by a student in 5 subjects : 56, 31, 56, 8 and 32

Ascending Order:
8, 31, 32, 56, 56

- Median = 32

- Scott took 8 math tests in one marking period. What is the median test score? Scores =[89, 73, 84, 91, 87, 77, 94, 67]
- Median = scores[3]+scores[4]/2 = 134.5
- Note: Median does not depend on every value in your data set.

Numpy provides a function median():

```
>>>import numpy as np
>>>x = np.array([56, 31, 56, 8, 32])
>>>print np.median(x)
32.0
```

Calculating the median in Python :

```
def median(v):
   """finds the 'middle-most' value of v"""
   n = len(v)
   sorted_v = sorted(v)
   midpoint = n // 2

   if n % 2 == 1:
      # if odd, return the middle value
      return sorted_v[midpoint]
   else:
      # if even, return the avg of the middle values
      lo = midpoint - 1
      hi = midpoint
      return (sorted_v[lo] + sorted_v[hi]) / 2
```

```
#main

x = [56, 31, 56, 8, 32]
print median(x)

32
```

Central Tendency

Mean Vs Median:

- If data are nominal scale (Label/String), you probably shouldn’t be using either the mean or the median.
- If data are ordinal scale(values in order), you’re more likely to want to use the median than the mean.
- For interval(integer)and ratio(float) scale data, either one is generally acceptable.

Central Tendency

Mean Vs Median:

Choosing median or mean plays an important role in analysis

Example: Income of 3 employees is given : Ramu-50,000 krishna-65,000 Sastry-60,000

Subbu added with 10,00,00,000

```
>>>income = [50000,65000,60000]
>>> print mean(income)
58333.3333333
>>> print median(income)
60000
```

```
>>>incme_new = [50000,65000,60000,100000000]
>>>print mean(incme_new)
25043750.0
>>>print median(incme_new)
62500.0
```

Note: For extreme-valued observations called outliers, mean is not suitable. Because the result would be misleading.

Scenario-1

- Last year, a fast food outlet in a beachside city paid 3 kitchen hands $16000 per year, 2 supervisors $22000 and the owner $85000.
- The mean salary at this business was $29500 and the median was $19000 (the mean of the two centre observations in an ordered list). The mode was $16000.
- Explain why the mean is higher than the median.

Mode

- **Mode** – the value that occurs most often.
- In a given dataset we can have more than one mode.
- To find the mode, first put the numbers in order, then count how many of each number. A number that appears most often is the mode.

weights = [100, 49, 41, 40, 25,78, 57, 89, 45,34,12,34, 45, 56,67, 87, 89, 90,81, 73]

Mode = [34, 45, 89]

```
# Mode function in Python
def mode(x):
    """returns a list, might be more than one mode"""
    cnts = Counter(x)
    max_cnt = max(cnts.values())
    return [x_i for x_i,cnt in cnts.iteritems()
            if cnt == max_cnt]
```

```
print "Mode =",mode(weights)
```

Quantile

- A generalization of the median is the Quantile.
- represents the values less than which a certain percentile (quartile, decile, etc.) of the data lies.
- The median represents the value less than which 50% of the data lies.

scores = [100, 49, 41, 40, 25,78, 57, 89, 45,34, 12,34, 45, 56, 67, 87, 89, 90, 81, 73]

```
def quantile(x,p):
    """returns the pth-percentile value in x"""
    pi = int(p * len(x))
    return sorted(x)[pi]
```

```
print quantile(scores, 0.10) #decile
print quantile(scores, 0.25) #quartile
print quantile(scores, 0.75) #75%
```

Scenario-2

- Write a program to answer the following by using methods of CentralTendency class.
- The weights, in kilograms, of the 8 members of Rams House tug of war team at a school sports are: 75, 73, 77, 76, 84, 76, 77, 78.
- Calculate the mean weight of the team.
- The 8 members of Syam House tug of war team have a mean weight of 64 kilograms.
- Which team do you think will win a tug of war between Ram’s House and Syam’s House? Give a reason for your answer.

Scenario - 3

- Create a Module called Statistics with CentralTendency class having methods such as mean, median, mode, and quantile.
- Consider 5 families with incomes of $20,500, $25,000, $35,000, $60,000 and $750,000.
- Find the average income of data given in the above example using suitable methods of CentralTendency class.
- Which measure among the Mean and Median is relevant? Why?

Measures of Dispersion or Variability

- Dispersion – measures or summarize the amount of spread or variation in the distribution of values in a variable.
- how values are spread out within a distribution.
- There are a number of different measures:
    - Range
    - Variance
    - Standard Deviation

Range

Range = **highest Value - lowest value** in the data set.

- Ex: Consider Weekly scores of 10 students.

  Weekly_scores=[180, 220, 280, 320, 280, 180, 350, 280, 330, 220]

  range = 350-180 = 170.

- Range function

  def data_range(x):

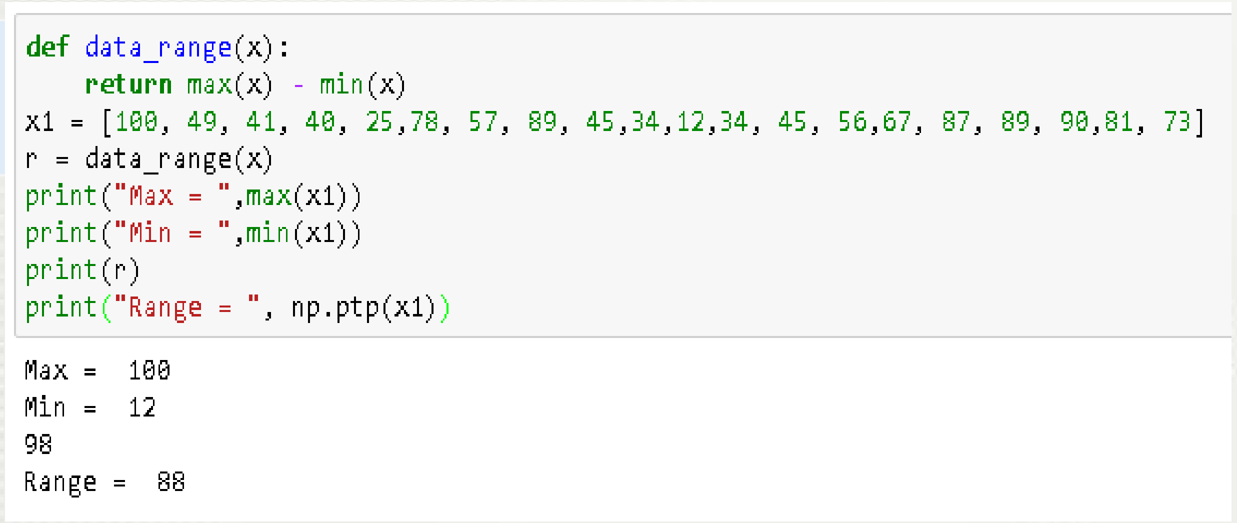
  return max(x) - min(x)

  r = data_range(scores)

  print r

  170

```
>>> data = [2,2,2,2,2,2,2,2]
>>>print data_range(data)
0
```



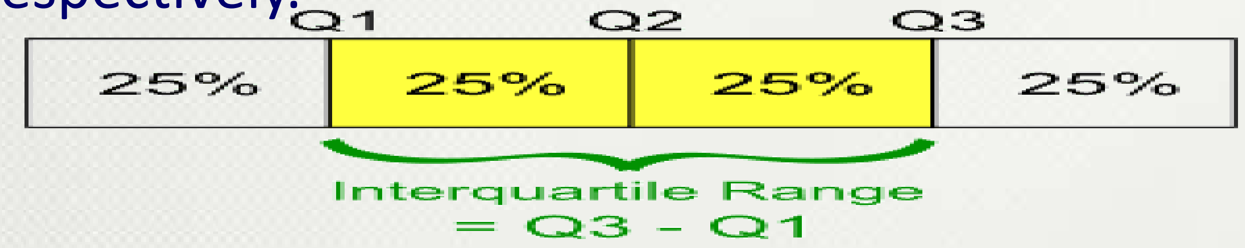
```
def data_range(x):
    return max(x) - min(x)
x1 = [100, 49, 41, 40, 25,78, 57, 89, 45,34,12,34, 45, 56,67, 87, 89, 90,81, 73]
r = data_range(x)
print("Max = ",max(x1))
print("Min = ",min(x1))
print(r)
print("Range = ", np.ptp(x1))
```

```
Max =  100
Min =  12
98
Range =  88
```

- **if r = 0, all elements are same. If r is large, the data is more spread out. It does not depend on whole dataset.**

Interquartile Range

- The **interquartile range** (**IQR**) is a measure of variability, based on dividing a data set into quartiles. Quartiles divide a rank-ordered data set into four equal parts. The values that divide each part are called the first, second, third, & fourth quartiles; and they are denoted by Q1, Q2, and Q3, respectively.

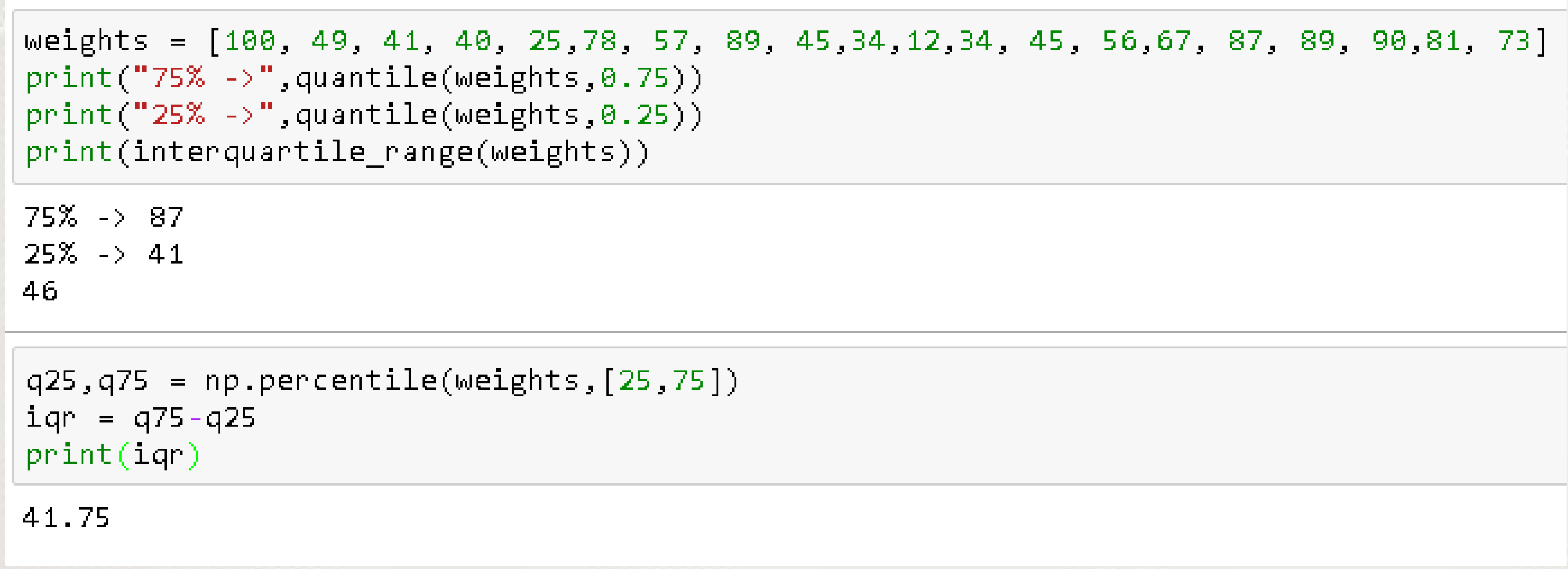


- Is a **measure of where the "middle fifty" is in a data set**. It is where the bulk of the values lie.
- It Calculates difference between 25th quantile and 75th quantile.

Interquartile Range

```
#interquartile range = quantile(x,75)-quantile(x,25)
def interquartile_range(x):
    return quantile(x,0.75)-quantile(x,0.25)
weights = [100, 49, 41, 40, 25,78, 57, 89, 45,34,12,34, 45, 56,67, 87, 89, 90,81, 73]
>>>print quantile(weights,0.25)
41
>>>print quantile(weights,0.50)
57
>>>print quantile(weights,0.75)
87
>>>print median(weights)
56.5
```

```
>>>print interquartile_range (weights)
46
```



```
weights = [100, 49, 41, 40, 25,78, 57, 89, 45,34,12,34, 45, 56,67, 87, 89, 90,81, 73]
print("75% ->",quantile(weights,0.75))
print("25% ->",quantile(weights,0.25))
print(interquartile_range(weights))
```

```
75% -> 87
25% -> 41
46
```

```
q25,q75 = np.percentile(weights,[25,75])
iqr = q75-q25
print(iqr)
```

```
41.75
```

Variability

- **Mean Absolute Deviation:** It is the average distance (deviation) of all the elements in the dataset from the mean of the same dataset.

$$MAD = \frac{1}{n}\sum_{i=1}^{n} |x_i - \bar{x}|$$

- Observations: 2,3,4,5,6,7,8,9
  - Mean =5.5
  - Mean-observations = 3.5,-2.5,-1.5,-0.5,0.5,1.5,2.5,3.5
  - |mean-observations| = 3.5 2.5 1.5 0.5 0.5 1.5 2.5 3.5
  - Mean absolute deviation = (3.5+2.5+1.5+0.5+0.5+1.5+2.5+3.5)/8 =2

With numpy

```
>>> data = np.array([2,3,4,5,6,7,8,9])
>>> print mean(data)-data              output = [ 3.5  2.5  1.5  0.5 -0.5 -1.5 -2.5 -3.5]
>>> print abs(mean(data)-data)         output = [ 3.5  2.5  1.5  0.5  0.5  1.5  2.5  3.5]
>>> print mean(abs(mean(data)-data))   output =  2.0
```

Variability
Mean Absolute Deviation

In python:

```
>>> data = [2,3,4,5,6,7,8,9]
>>> def de_mean(x):
    """translate x by subtracting its mean"""
    x_bar = mean(x)
    return [x_i - x_bar for x_i in x]

>>> print de_mean(data)
[-3.5, -2.5, -1.5, -0.5, 0.5, 1.5, 2.5, 3.5]
>>> print map(abs,de_mean(data))
[3.5, 2.5, 1.5, 0.5, 0.5, 1.5, 2.5, 3.5]
>>> print mean(map(abs,de_mean(data)))
2.0
```

Variance

- **Variance :** determines how close the scores in the distribution are to the middle of the distribution.
- The average squared difference of the scores from the mean.

For complete data set $$\sigma^2 = \frac{\Sigma(X-\mu)^2}{N}$$ **For Sample:** $$s^2 = \frac{\Sigma(X-M)^2}{N-1}$$

  - Observations: 2,3,4,5,6,7,8,9
  - Mean =5.5
  - $(|\text{mean-observations}|)^2$ = 12.25 6.25 2.25 0.25 0.25 2.25 6.25 12.25
  - variance = (12.25 +6.25+2.25+0.25+0.25+2.25+6.25+12.25)/8 =5.25

- Range Vs Variance:
  - range looks at the extremes (**biggest value-smallest value)**
  - variance **looks at all the data points. It determines distribution of data around mean**.
  - Variance - **average** (**squared deviations** from the **Mean)**

Variance

- Procedure:
  - Find out the Mean
  - for each element: subtract the Mean, then square the result (the *squared difference*).
  - Find the average (squared differences).

```
# assumes x has at least two elements
def variance(x):
   n = len(x)
   deviations = de_mean(x)
   return sum_of_squares(deviations)/(n-1)

>>>dataset=[2,3,4,5,6,7,8,9]
>>> print variance(scores)
>>> 6.0
```

- **Variance = 0 , when all the elements are identical**

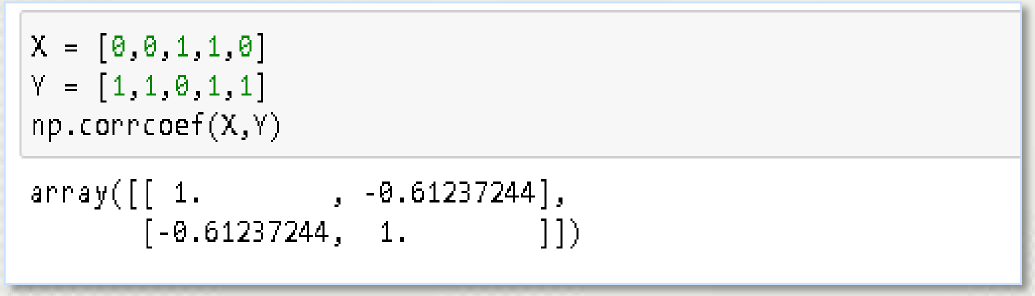
Standard Deviation

- Standard Deviation – the square root of Variance.
- used to find the outliers.
- useful measure of variability when the distribution is normal

The "Population Standard Deviation“:	$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$
The "**Sample** Standard Deviation**":**	$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$

Correlation

- **Correlation** – show whether and how strongly pairs of variables are related.
  - For Example**:** height, weight are related; taller people tend to be heavier than shorter people.
  - The **relationship isn't perfect**. People of the same height vary in weight,
  - consider two people you know where the shorter one is heavier than the taller one. The average weight of people 5'5'' is less than the average weight of people 5'6'', etc.
  - **Correlation can tell you just how much of the variation in peoples' weights is related to their heights.**



```
X = [0,0,1,1,0]
Y = [1,1,0,1,1]
np.corrcoef(X,Y)
```

```
array([[ 1.        , -0.61237244],
       [-0.61237244,  1.        ]])
```

Correlation

- **Correlation works for quantifiable data** (Numerical Data).
- It cannot be used for purely categorical data, such as gender, brands purchased, or favourite colour.
- Correlation is used to understand the relationship between variables such as:
  - Whether the relationship is +ve or –ve
  - the strength of the relationship.
- **Positive correlation**: is a relationship between two variables where if one variable increases, the other one also increases and vice-versa
  - Eg: family size , family expenditure will increase or decrease together.

Correlation

- **Negative Correlation**: there is an inverse relationship between two variables - when one variable decreases, the other increases and vice-versa.
  - **Eg: negative correlation**, between **price** and **demand** for goods and services. As the price of goods and services increases, the quantity demanded falls.
- **Coefficient of Correlation (r):** Statistical correlation is measured is known as "**coefficient of correlation (r)**". Its numerical value ranges from +1.0 to -1.0. It gives us **the strength of relationship.**

Correlation

- In general, $r > 0$ – positive relationship
- $r<0$ – negative relationship
- $r = 0$ – no relationship (meaning the variables are independent and not related)
- when $r = +1.0$ – describes a perfect +ve correlation
- when $r = -1.0$ – describes a perfect -ve correlation
- closer the coefficients are to +1.0 and -1.0, greater is the strength of the relationship between the variables.

Covariance

- **Covariance** - indicates how two variables are related ie, is **how two variables are deviated from their means**.
- A positive covariance - variables are positively related,
- A negative covariance - variables are inversely related.

$$\mathrm{COV}(x,y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{n-1}$$

x = independent variable, *y* = dependent variable
n = number of data points in the sample
x’, y’ = the mean of *x, y respectively*

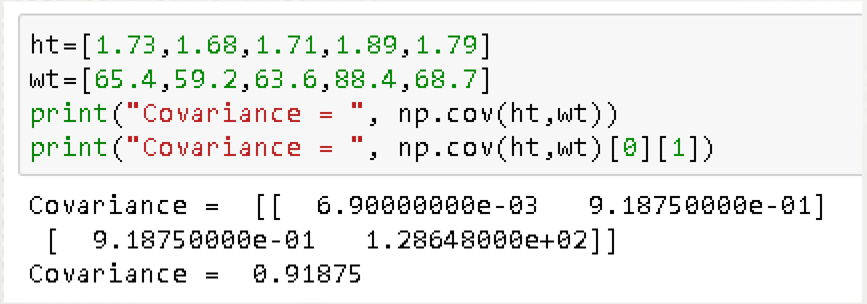
- A **large +ve covariance** means – **x tends to be large, when y is large, and small when y is small**.
- A **large -ve covariance** means – **x tends to be large, when y is small, and small when y is large**

Covariance

```
def de_mean(x):
   """ to calculate x – x' """
   x_bar = mean(x)
   return [i - x_bar for i in x]

def covariance(x,y):
    n=len(x)
    return dot(de_mean(x),de_mean(y))/(n-1)
```

```
>>>ht=[1.73,1.68,1.71,1.89,1.79]
>>>wt=[65.4,59.2,63.6,88.4,68.7]
>>>print covariance(ht, wt)
0.9187
```



```
ht=[1.73,1.68,1.71,1.89,1.79]
wt=[65.4,59.2,63.6,88.4,68.7]
print("Covariance = ", np.cov(ht,wt))
print("Covariance = ", np.cov(ht,wt)[0][1])
```

```
Covariance =  [[  6.90000000e-03   9.18750000e-01]
 [  9.18750000e-01   1.28648000e+02]]
Covariance =  0.91875
```

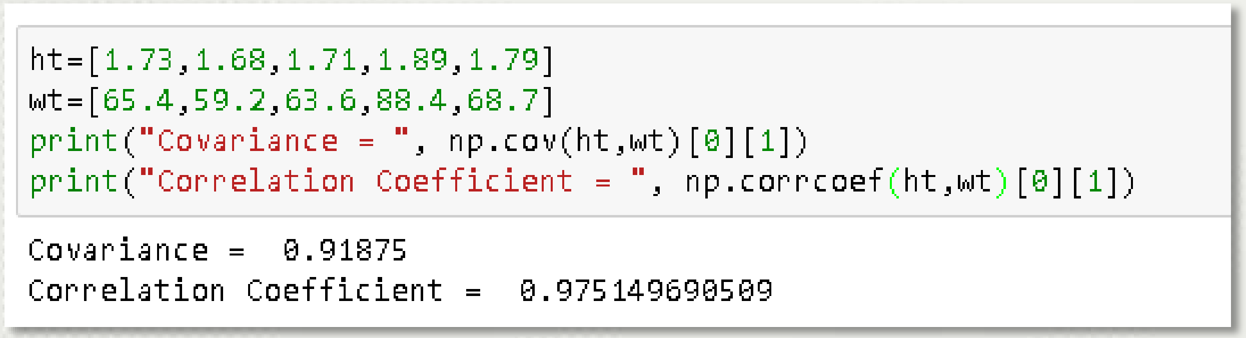
- **This example is showing large +ve covariance** meaning – ht **tends to be large, when** wt **is large, and small when** wt **is small**.

Correlation

•#correlation – divides out the standard deviations of both variables x,y.

```
def correlation(x,y):
    sd_x = std_deviation(x)
    sd_y = std_deviation(y)
    if sd_x > 0 and sd_y > 0:
        return covariance(x,y) / sd_x / sd_y
    else:
        return 0

>>> print correlation(ht,wt)
0.975149690509
```



```
ht=[1.73,1.68,1.71,1.89,1.79]
wt=[65.4,59.2,63.6,88.4,68.7]
print("Covariance = ", np.cov(ht,wt)[0][1])
print("Correlation Coefficient = ", np.corrcoef(ht,wt)[0][1])
```

```
Covariance =  0.91875
Correlation Coefficient =  0.975149690509
```

Correlation

- While 'r' (correlation coefficient) is a powerful tool, it has to be handled with care.
- mostly correlation coefficients only **measure linear relationship**. It is also possible that there is strong **non linear relationship** between the variables, r is close to 0 or even 0. In such a case, a scatter diagram can roughly indicate the existence or otherwise of a non linear relationship.
- Must be careful in interpreting the value of 'r'. For Ex, one could compute 'r' between the size of shoe and intelligence of individuals, heights and income. Which is non sense.
- 'r' should not be used to say anything about cause and effect relationship.

Conclusion

Discussed about ...

- Files – Reading – Writing

Next Session .....

Data Pre-Processing

THANK YOU