→ Descriptive Analytics

1. Data summarization
2. Visualization
3. Probability

→ Types of Data w.r.t statistics

1. Continuous data
   - There is possibility of decimal values
   - Always be NUMERIC
   - E.g. distance b/ KHGR-CSMT is 85.45metres
2. Discrete data
   - There is no possibility of decimal
   - Only whole numbers allowed, usually counting
   - Always be NUMERIC
   - E.g. count of students in class is 84
3. Attribute data
   - Characteristic data / attributes
   - E.g. gender of Ryan is ‘Male’

Descriptive Statistics

- helps to describe, show and summarize data in a meaningful way.
- describing features of a data set by generating summaries
- state facts and proven outcomes from a population
- describe a situation
- achieved with the help of charts, graphs, tables, etc.

NumPy

```
import numpy as np
import pandas as pd
# importing NumPy & pandas
```

```
grades = np.array(['A', 'C', 'D', 'A', 'A', 'B', 'C', 'A', 'B', 'B', 'B', 'A', 'C', 'B', 'A', 'A', 'B', 'C'])
# creating an ndarray of grades
```

```
len(grades)
# printing the count of grades
```

```
18
```

np.unique(ndarray)

```
np.unique(grades)
# printing nd array of only unique values from existing ndarray
```

```
array(['A', 'B', 'C', 'D'], dtype='<U1')
```

np.unique(ndarray, return_counts=True)

```
np.unique(grades, return_counts=True)
# returns a tuple of two ndarrays, one with unique values and
# other with their respective count
# unique values are attribute data
# count of unique values is discrete data
```

```
(array(['A', 'B', 'C', 'D'], dtype='<U1'), array([7, 6, 4, 1]))
```

```
grd_counts = np.unique(grades, return_counts=True)
# returns a tuple of two ndarrays, one with unique values and
# other with their respective count
```

```
grd_counts[0]
# printing unique values from tuple grd_counts
```

```
array(['A', 'B', 'C', 'D'], dtype='<U1')
```

```
grd_counts[1]
# printing count of unique values from tuple grd_counts
```

```
array([7, 6, 4, 1])
```

Proportion

- Proportion = count / total number of items

```
grd_props = grd_counts[1] / len(grades)
grd_props
# calculate grade proportion
```

```
array([0.38888889, 0.33333333, 0.22222222, 0.05555556])
```

```
grd_props = np.round(grd_props, 3)
grd_props
# rounding off grade proportion
```

```
array([0.389, 0.333, 0.222, 0.056])
```

Proportion Percentage

- proportion percentage = Proportion * 100
- proportion percentage = (count/total Number of items)*100

```
grd_pct = grd_props*100
grd_pct
# calculate grade percentage from grade proportion
```

```
array([38.9, 33.3, 22.2,  5.6])
```

np.cumsum()

- returns cummulative sum
- keeps summing up values index-wise

```
np.cumsum(grd_pct)
# calculate summulative sum, means to add grade percentages cummulativly
# to tell
```

```
array([ 38.9,  72.2,  94.4, 100. ])
```

Sorting

```
x = np.array([45, 78, 23, 56, 12])
# creading an ndarray
```

```
np.argsort(x)
# returns index in order of sorted numbers
```

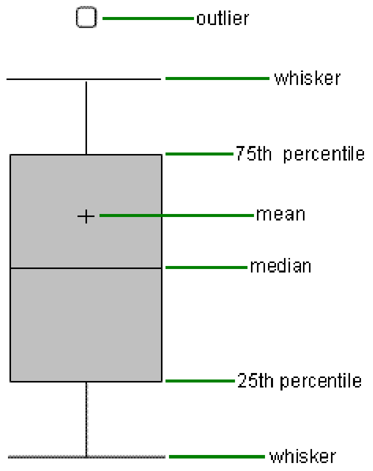
```
array([4, 2, 0, 3, 1])
```

```
sorted(x)
# returns list in sorted order
```

```
[12, 23, 45, 56, 78]
```

Box Plot

- method for descriptive statistics
- also known as box and whisker plot
- includes two parts, a box and a set of whiskers
- consists of five number summary of data set
    1. Minimum (0th percentile) - lower whisker
    2. first quartile (25th percentile)
    3. Median (50th percentile) - horizontal line in the middle of box
    4. third quartile (75th percentile)
    5. Maximum (100th percentile) - upper whisker
- we draw a box from the first quartile to the third quartile
- A vertical line goes through the box at the median, and



IQR

- Inter-Quartile Range
- consists of the central 50% of the data
    - `IQR = Q3 - Q1`

    ```
    IQR = third quartiile - first quartile
    IQR = 75th percentile - 25th percentile
    ```

Whiskers

- two whiskers represent the maximum & minimum
    - Lower whisker - minimum / 0th Percentile
    - Upper Whisker - maximum / 100th Percentile

Fences

- used to identify outliers

1. lower inner fence: `Q1 - 1.5*IQ`

2. upper inner fence: `Q3 + 1.5*IQ`

sometimes we consider 3IQR too for fences

3. lower outer fence: `Q1 - 3*IQ`
4. upper outer fence: `Q3 + 3*IQ`

Outliers

- a data point that is located outside the fences of the box plot
- represented by dot, a small circle, a star, etc.

Three types of Descriptive summaries

1. Distribution (Probability Distribution)
2. Central Tendency
3. Variability

- Central Tendency
    1. mean
    2. median
    3. mode
- Variation or Dispersion
    1. range
    2. inter-quartile range
    3. variance
    4. standard deviation
    5. Shape
        1. skewness
        2. kurtosis
- Probability Distributions
    1. Discrete Probability Distributions
        1. Binomial Distribution
        2. Poisson Distribution

        3. Hypergeometric Distribution
    2. Continuous Probability Distributions
        1. Exponential Probability Distributions
        2. Normal Probability Distributions
        3. Student's t Probability Distributions

Central Tendency

- represent the center point or typical value of a dataset
- descriptive "summary" of a data set

1. Mean
2. Median
3. Mode

Mean / Average x̄ or μ

- Mean or Average : sum of all elements / count of elements
- `Mean or Average : (x1 + x2 + x3 + ... + xn) / n`

1. Sample Mean / Average (x̄)
    - `x̄` : Sample Mean of variable `x`
    - `Σxn` : sum of `n` values
    - `n` : number of values in sample
    - `x̄ = Σxn / n`
2. Population Mean / Average (μ)
    - `μ` : Population Mean / Average
    - `∑XN` : sum of `N` values
    - `N` : number of values in the population
    - `μ = ∑XN / N`

Note:

- Inferential statistics use Hypothesis tests such as t-tests, ANOVA tests to determine whether Sample data is different than Population mean

np.mean(ndarray)

```
mylist = [3, 6, 5, 4, 7, 6, 8, 9, 1, 10, 13, 4, 7, 11, 4, 7, 9]
# create a list of values
mylist
# printing the list
```

```
[3, 6, 5, 4, 7, 6, 8, 9, 1, 10, 13, 4, 7, 11, 4, 7, 9]
```

```
sum(mylist)/len(mylist)
# manually calculating mean
```

```
6.705882352941177
```

```
np.mean(mylist)
# Claculates Population Mean or Average of the the ndarray / list
```

```
6.705882352941177
```

Weighted Mean

- 10 have value as 7
- 15 have value as 8
- 23 have value as 19
- 8 have value as 10

- Assign a weight to each value in the dataset

- frequency is the weight

```
- x1 = 07     w1 = 10

- x2 = 08     w1 = 15

- x3 = 09     w1 = 23

- x4 = 10     w1 = 08
```

  - `Weighted Mean = ∑xw / ∑w`
  - `Weighted Mean = (x1w1 + x2w2 +x3w3 + x4w4) / (w1 + w2 + w3 + w4)`

```
values = np.array([7, 8, 9, 10])
# creating ndarray of values / weights to assign weight
```

```
freq = np.array([10, 15, 23, 8])
# creating ndarray of frequencies
freq
```

```
array([10, 15, 23,  8])
```

```
freq*values
# calculating products of weights & samples
```

```
array([ 70, 120, 207,  80])
```

```
freq*values/sum(freq)
# average or mean of each weight
```

```
array([1.25      , 2.14285714, 3.69642857, 1.42857143])
```

```
sum(freq*values/sum(freq))
# weighted mean / average
```

```
8.517857142857142
```

Median

- natural measure of central tendency
- splits the dataset in half
- middle value in sorted list of data
- first sort the data in increasing order, then median is the middle value
  - for odd sample size, `Median = middle value`

    ```
                  Median = middle value
                  Median = ((n+1)/2)th term
    ```

    - e.g. `sample size, n=7`

      ```
      Median = ((7+1)/2)th term
      Median = (8/2)th term
      ```

```
Median = 4th term
```

- for even sample size: `Median = Average of two Middle values`

```
        Median = Average of two Middle values
        Median = ((n/2)th term + ((n/2)+1)th term)/2
```

  - e.g. `sample size, n=8`

```
Median = ((8/2)th term + ((8/2)+1)th term)/2
Median = (4th term + (4+1)th term)/2
Median = (4th term + 5th term)/2
```

```
mylist = [3, 6, 5, 4, 7, 6, 8, 9, 1, 10, 13, 4, 7, 11, 4, 7, 9]
# create a list of values
mylist
# printing the list
```

```
[3, 6, 5, 4, 7, 6, 8, 9, 1, 10, 13, 4, 7, 11, 4, 7, 9]
```

```
len(mylist)
```

```
17
```

```
sorted(mylist)[8]
# median: middle value in sorted list
# 17+1/2 = 18/2 = 9th value = at 8th index
```

```
7
```

np.median(ndarray or list)

```
np.median(mylist)
# median using Numpy median method for the array / list
```

```
7.0
```

Mean vs. Median ... when to use mean and when median

- mean might skew your average value drastically, while median does not skew the middle value

1. mean should be used when
   - when data distribution is symmetric
   - data doesnot have any extreme values
2. for symmetric distribution, both mean & median are almost same
3. for skewed distribution, median better represents the central tendency
4. median is a robust statistic, mean is sensitive to outliers and skewed distributions
5. use median only for Skewed distribution, Continuous data, Ordinal data

Note

- if data is symmetric, then mean = median
- example of symmetric distribution is normal distribution curve/ bell curve

```
sal = [6, 7, 6, 7, 6, 7, 8, 8, 6, 7, 6, 8, 8, 7, 7,8, 6, 7, 8, 6]
# list with symmetric/normal distribution
```

```
np.mean(sal)
# Mean / Average
```

```
6.95
```

```
np.median(sal)
# Median

# note that, for symmetric data, as there are no extreme values
# so mean & median are almost same
```

```
7.0
```

```
new_sal = [6, 7, 6, 7, 6, 7, 8, 8, 6, 7, 6, 8, 8, 7, 7,8, 6, 7, 8, 6, 54, 50, 52]
# list with extreme values, does not follow symmetric distribution
```

```
np.mean(new_sal)
# as there are extreme values, so mean / average might be misleading
```

```
12.826086956521738
```

```
np.median(new_sal)
#  median shows better reality for middle value
# note that, as there are extreme values, mean has changed drastically
# but median is not impacted
```

```
7.0
```

Mode

- most frequent value
- two value can be mode if both hold the highest frequency
- if no value is repeating, (all values have frequency=1), then data does not have a mode
- use mode for categorical , ordinal and discrete data only

Note

- for symmetrical data, mean, median and mode are same

```
from collections import Counter
```

```
mylist = [3, 6, 5, 4, 7, 6, 8, 9, 1, 10, 13, 4, 7, 11, 4, 7, 9]
# create a list of values
mylist
# printing the list
```

```
[3, 6, 5, 4, 7, 6, 8, 9, 1, 10, 13, 4, 7, 11, 4, 7, 9]
```

```
Counter(mylist).most_common()
# returns unique values and their espective count
# Mode: element with most number of occurences
# mode is 4, 7 with frequecy 3
```

```
[(4, 3),
 (7, 3),
 (6, 2),
 (9, 2),
 (3, 1),
 (5, 1),
 (8, 1),
 (1, 1),
 (10, 1),
```

```
 (13, 1),
 (11, 1)]
```

Variation or Dispersion

1. Range
2. Inter-Quartile Range
3. Variance
4. Standard Deviation

Range

- easiest to calulate, simplest measure of dispersion
- denotes the spread of observations
- difference between maximum (0th Percentile) and minimum (100th percentile)
- `range(X) = max(x) - min(x)`

Drawbacks of range

1. it uses only two elements of multiple elements, which does not describe intermediate values, making it unreliable if intermediate values change
2. calculation of range can be affected by presence of extreme values
3. does tell anything about variablility of other data

```
mylist = [3, 6, 5, 4, 7, 6, 8, 9, 1, 10, 13, 4, 7, 11, 4, 7, 9]
# create a list of values
mylist
# printing the list
```

```
[3, 6, 5, 4, 7, 6, 8, 9, 1, 10, 13, 4, 7, 11, 4, 7, 9]
```

```
max(mylist) - min(mylist)
# range: max - min
# printing the range for list 'mylist'
```

```
12
```

```
x = [1, 4, 4, 7, 5, 8, 9, 3, 10]
max(x)-min(x)
# printing the range for list 'x'
```

```
9
```

```
y = [1, 4, 4, 4, 4, 4, 4, 4, 10]
max(y)-min(y)
# printing the range for list 'y'
```

```
9
```

Inter-Quartile Range (IQR)

- Inter-Quartile Range
- consists of the central 50% of the data
    - `IQR = Q3 - Q1`

    ```
    IQR = third quartile - first quartile
    IQR = 75th percentile - 25th percentile
    ```

- cannot calculate IQR directly in python, first calulate 75th percentile & 25th percentile using `np.percentile(ndarray, percentile)`, and then find the difference

```
mylist = [3, 6, 5, 4, 7, 6, 8, 9, 1, 10, 13, 4, 7, 11, 4, 7, 9]
# create a list of values
mylist
# printing the list
```

```
[3, 6, 5, 4, 7, 6, 8, 9, 1, 10, 13, 4, 7, 11, 4, 7, 9]
```

np.percentile(ndarray, percentile)

```
np.percentile(mylist, 75)
# calculates 75th percentile
```

```
9.0
```

```
np.percentile(mylist, 75) - np.percentile(mylist, 25)
# IQR = 75th percentile - 25th percentile
# calculates IQR
```

```
5.0
```

```
x = [1, 4, 4, 7, 5, 8, 9, 3, 10]
# asymmetric data - has extreme values/ skewed distribution
y = [1, 4, 4, 4, 4, 4, 4, 4, 10]
# symmetric data - has normal distribution
```

```
np.percentile(x, 75) - np.percentile(x, 25)
# for asymmetric data, there will be some value of IQR
```

```
4.0
```

```
np.percentile(y, 75) - np.percentile(y, 25)
# for symmetric data, IQR is zero
# it has same value on both sides of median, so IQR = zero
```

```
0.0
```

Deviation, D

- calculated by taking the differences between each number in the data set and the mean
- deviation of each observation (X) w.r.t. Sample Mean (x̄), which can be +ve or -ve
- sum of all devations is zero
- `Deviation, D = Observation - Sample Mean`
- `Deviation, D = Xn - x̄`

Variance S^2 or σ^2

- measures how far each number in the set is from the mean
- measurement of the spread between numbers in a data set
- square of deviation for each observation
- unit is also squared, as it is squared to remove negative values

1. Sample Variance, S^2
    - `Sample Variance, S^2 = (Observation - Sample Mean)^2/Number Of Values`

- `Sample Variance, S^2 = Σ(Xn - x̄)^2 / n-1`

2. Population Variance, σ^2

   - `Population Variance, σ^2 = (Observation - Population Mean)^2/Number Of Values`
   - `Population Variance, σ^2 = (Xn - μ)^2 / N`

np.var(ndarray, ddof=0)

```
x = [4, 6, 7, 9, 10, 3, 5, 8, 3, 7]
```

```
np.var(x, ddof=0)
# when ddof=0, calculates Population Variance
```

```
5.36
```

```
np.var(x, ddof=1)
# when ddof=0, calculates Sample Variance
```

```
5.955555555555556
```

```
np.var(x, ddof=0)**0.5
# population standard deviation is square root of population variance
```

```
2.3151673805580453
```

Standard Deviation, σ or S.D.

- square root of variance
- unit is same as observation data, as it is square root of variance
- represents the average distance from the mean
- S.D. reflects the extent of variation in data
- As variation increases, value of S.D also increases

1. Sample S.D.

   - `Sample S.D., σ = sqrt(sum(Observation - Sample Mean)**2/Number Of Values)`
   - `Sample S.D., σ = sqrt(Σ(Xn - x̄)**2 / n-1)`

2. Population S.D.

   - `Population S.D., σ = sqrt(sum(Observation - Population Mean)**2/Number Of Values)`

    - `Population S.D., σ = sqrt(Σ(Xn - μ)**2 / N)`

- `Sample S.D.` > `Population S.D.` because there is uncertainity in sample S.D. as dof is more
- `degreee of freedom (dof) = Sample Size (N) - 1`

np.std(ndarray, ddof=0)

```
np.std(x, ddof=0)
# when ddof=0, calculates Population standard deviation
# ddof : delta degree of freedom
```

```
2.3151673805580453
```

```
np.std(x)
# when ddof is not specified, ddof=0, calculates Population standard deviation
```

```
2.3151673805580453
```

```
np.std(x, ddof=1)
# sample standard deviation
# when ddof=1, calculates Sample standard deviation
```

```
2.440400695696417
```

```
x = [4, 7, 6, 8, 9, 2, 3, 5, 4, 7]
```

```
np.std(x, ddof=0)
# population standard deviation
```

```
2.1563858652847827
```

```
np.std(x, ddof=1)
# sample standard deviation
```

```
2.273030282830976
```

Shape

1. skewness - for location of peak
2. kurtosis - for nature of peak

Skewness, Skew[X]

- measure of the lack of symmetry

1. Symmetric
    - A distribution, or data set, is symmetric if it looks the same to the left and right of the center point
    - for symmetric curve
    - normal distribution
    - `Skew[X] = 0`
    - e.g. bell curve / normal distribution
2. peak at left
    - when peak is at left, it is Right skewed
    - positively skewed, `Skew[X] > 0 (+ve)`
    - e.g. salary distribution of employees, seniors have more salary but have less frequency,
3. peak at right
    - when peak is at right, it is Left skewed
    - negatively skewed, `Skew[X] < 0 (-ve)`
    - e.g. age distribution of members of parliament, young people are very

Note:

- Using skewness, we can predict if calculating mean would be risky or not, because mean for data which is not normally distributed, would be incorrect

```
import scipy
from scipy import stats
```

```
x = [1,1,2,2,2,2,2,2,2,2,2,2,3,3,3,4,4,5,6]
y = [1,2,2,3,3,4,4,4,5,5,5,5,5,5,5,5,5,6,6,7]
z = [1,1,2,2,2,3,3,3,3,4,4,4,4,4,4,4,4,4,4,4,4,4,5,5,5,5,6,6,6,7,7]
```

```
scipy.stats.skew(x)
# Positive skewness means Right Skewed or peak at left
```

```
1.1839325019418967
```

```
scipy.stats.skew(y)
# Negative skewness means Left Skewed or peak at right
```

```
-0.6877802002110032
```

```
scipy.stats.skew(z)
# Zero skewness means Normal Distribution
```

```
0.0
```

Kurtosis, K

- is a unitless measure that describes sharpness of the peak
- describes range of data, how much of variation is there in data
- degree to which data values are concentrated around the mean
- measure of whether the data are heavy-tailed or light-tailed relative to a normal distribution

1. mesokurtic
    - for normal distribution curve or bell curve
    - `kurtosis = 0`
2. leptokurtic if curve is sharper than normal distribution curve
    - heavy tails, `kurtosis > 0 (+ve)`
    - more peaked as compared to normal distribution curve
    - peak is prominent, so many outliers
3. platykurtic
    - light tails, `kurtosis < 0 (-ve)`
    - if curve is flat than normal distribution curve
    - less peaked than normal distribution curve
    - peak is less prominent, so fewer outliers

Note:

- higher the Kurtosis (+ve or -ve), means some elements have very high frequency

```
x = [1,1,1,2,2,2,3,3,3,3,4,4,4,4,4,5,5,5,5,6,6,6,7,7]
y = [1,1,2,2,2,3,3,3,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,5,5,5,5,6,6,6,7,7]
```

```
scipy.stats.kurtosis(y)
# positive kurtosis means sharp curve
```

```
0.70242214532872
```

```
scipy.stats.kurtosis(x)
# negative kurtosis means flat curve
```

```
-0.9451498127839182
```

Laptop Project

Refer: use data.xlsx file

- Q1. How many laptops did I sell overall? How many of those where Dell Laptops(absolute and percentage)?
- Q2. What was my total revenue? How much of that can be contributed to Dell(absolute and percentage)?
- Q4. How is the revenue distributed over laptop categories?
- Q5. What is the best selling continent, country and city?