# 2. Statistics

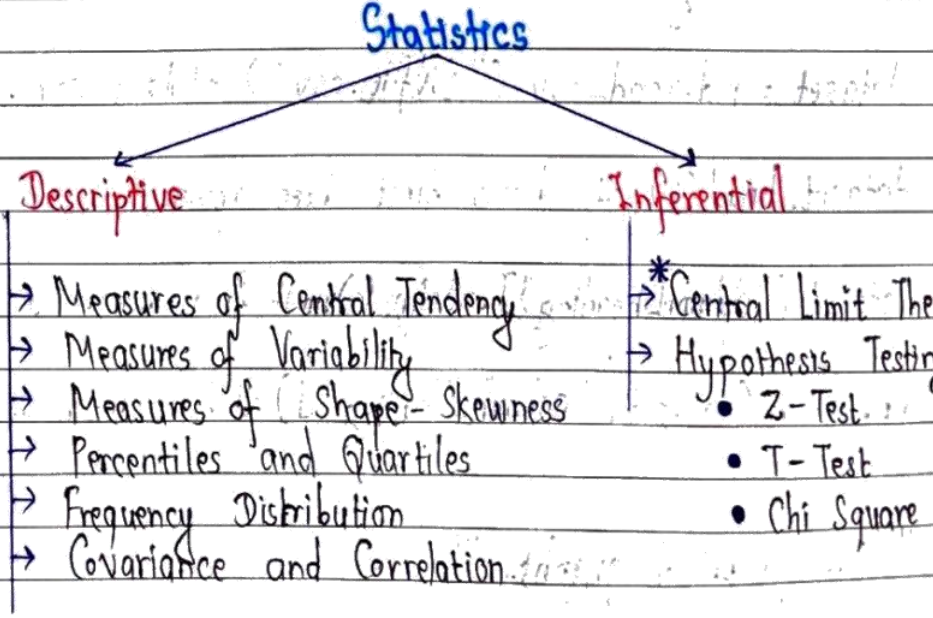


**What is statistics?**
Statistics is a set of mathematical methods and tools that enable us to answer important questions about data.

It is divided into two categories: Descriptive and Inferential

**Descriptive Statistics :** (on population data)
This offers methods to sumarized data by transforming raw observations into meaningful information that is easy to interpret and share.

**Inferential Statistics :** (on sample data → mostly used)
This offers methods to study experiments done on small samples of data chalk out the inferences to the enitre population (entire domain).



# 1. Measures of Central Tendency

- mean
- median
- mode

## 1. Mean : (Average)

Mean = $\frac{\text{Sum of all data}}{\text{Total no. of data}}$

```
import numpy as np
import pandas as pd

ar = np.array([4,5,6,2,1,8,5,6,4,7])

np.sum(ar)/len(ar) # manual process

np.mean(ar) # automatic process

''' For solving real world problems. '''

dataset = pd.read_csv("Students.csv") # read csv file

dataset.head(3) # shows first three rows

dataset["Age"].mean() # find mean using pandas

np.mean(dataset["Age"]) # find mean using numpy

''' For graphical output '''
```

```
import matploitlib.pyplot as plt # to create a graphs
import seaborn as sns # for advancements in graphs

mn = np.mean(dataset["Age"]) # store mean in mn

sns.histplot(x="Age", data=dataset, bins=[i for i in
        range(0, 100, 10)])
# bins is used to manage bins of graph columns.
# This is column vs data graph.
plt.plot([mn for i in range(0, 250)], [i for i in
        range(0, 250)], c="red")
# Display mean in graph.
plt.show() # display graph with all figures
```

## 2. Median : (Middle Value)

- Arrange data in ascending order.
- Odd no. of data → middle value
  Even no. of data → average of both middle values.

```
""" For finding median """

dataset.isnull().sum() # displays total number of nulls in
                       # each columns in dataset.

dataset["Age"].fillna(dataset["Age"].mean(),
        inplace=True)
```

```
# filling up the null values by mean of dataset column.
# (data cleaning uses measurement of central tendency)

# Now, there is no null value in column. So, can find median
np.median(dataset["Age"]) # find median using numpy

dataset["Age"].median() # find median using pandas

''' for Graphical output. '''

md = np.median(dataset["Age"]) # store median in md

sns.histplot(x="Age", data=dataset, bins=[i for i
    in range(0, 100, 10)]) # Graph format
plt.plot([md for i in range(0, 250)], [i for i in range
    (0, 250)], c="blue")
    # Plot mean in graph.
plt.show() # Display graph with all figures
```

## 3. Mode : (Most Repeated Value)

- Mode is a data with highest frequency (repeatation).
- Commonly used in catagorical data. e.g. gender, class, material status, sex, etc.

```
''' for finding mode. '''
```

Date
Page 7

```
dataset = pd.read_csv("Students.csv")
dataset["Age"].mode()[0]  # calculate mode using pandas.
    # index 0 is used because it returns a series and the
    # mode is the first data of series.

dataset["Age"].value_counts()  # Shows frequencies of
    # datas in column.


"""To get Graphical output."""

mo = dataset["Age"].mode()[0]  # Store mode in mo

sns.histplot(x="Age", data=dataset, bins=[i for i in
        range(0, 100, 10)])  # Graph format
plt.plot([mo for i in range(0, 250)], [i for i in range
        (0, 250)], c="green")  # Plot mode in graph
plt.show()  # Display graph with all figures


"""To display mean, median and mode in same graph"""

sns.histplot(x="column", data=dataset, bins=[i for i in
        range(0, 100, 10)])  # Graph format
plt.plot([mn for i in range(0, 250)], [i for i in range
        (0, 250)], c="red", label="mean")
    # Plot mean in a graph(start, end)
    # label is used to show the labeling of colours with
    # the graphs in legend.
plt.plot([md for i in range(0, 250)], [i for i in range
        (0, 250)], c="blue", label="median")
```

```
# Plot median in graph
plt.plot([mo for i in range(0, 250)], [i for i in
    range(0, 250)], c="green", label="mode")
# Plot mode in graph
plt.legend()  # Shows legend according to given labels
plt.show()  # Show graph with all figures
```

## 2. Measures of Variability

### 1. Range

- Range is the difference between the maximum and minimum values in the dataset.
  i.e. Range = maximum - minimum.
- It provides a simple measure of the spread of the data, but it can be sensitive to outliners.

```
''' for finding range of given dataset '''

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt  # not used
import seaborn as sns  # not used

dataset = pd.read_csv("Students.csv")

dataset.head(3)  # display first three rows
```

Date
Page 9

```
min_r = dataset["Age"].min() # Returns min value
max_r = dataset["Age"].max() # Returns max value.

min_r, max_r  # Prints min and max values
              # Example: (0.42, 80.0)

range = max_r - min_r  # Gives range of column data.
range  # Displays range
```

## 2. Mean Absolute Deviation

- The mean absolute deviation of a dataset is the average distance between each datapoint and the mean.
- It gives us an idea about the variability in dataset.
- Mathematically, $MAD = \frac{\sum |x_i - \bar{x}|}{n}$

"Making choice based on data of sections"

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

sec_a = np.array([75, 65, 73, 68, 72, 67])
sec_b = np.array([90, 47, 43, 96, 93, 51])
```

```
# Created two arrays with some data using numpy.
no = np.array([1,2,3,4,5,6])
  # Array for y-axis values.

plt.figure(figsize=(10,3)) # Fix figure size.
plt.scatter(sec_a, no, label="sec A")
    # Draw scatter graph of sec-a vs. no
plt.scatter(sec_b, no, c="blue", label="sec B")
    # Draw scatter graph of sec-b
plt.plot([70,70,70,70,70,70], no, c="red",
        label="mean")
    # plot a line of mean from 0 to 6.
plt.legend()  # display legend according to labels
plt.show()  # show graph with all figures.

"""From the digram, the less scatterred data is less
deviated and should be choosen.
But sometimes drawing graphs can be a tedious
task or even not possible. So, we need to have some
mathematical calculations to meet our needs.

The calculations can be done as follows:"""

# Find mean of both sections
# Mean absolute division is used when we have got the
# same mean while figuring out for best.
# i.e sec_a and sec_b has same mean
mean = np.mean(sec_b) # Same for sec_a.
mean  # display mean  : 70.0

# Find mean absolute deviation for both.
```

```
mad_a = np.sum(np.abs(sec_a - mean)) / len(sec_a)
mad_b = np.sum(np.abs(sec_b - mean)) / len(sec_b)

mad_a, mad_b  # display mean absolute deviation:
              # (3.3333333333333335, 23.0)
```

"We found that mean absolute deviation of sec-a is less than mean standerd deviation of sec-b.
So, sec-a should be our choice."

"If we got same mean absolute deviation for both data then standard deviation is used."

## 3. Standerd Deviation

- The standerd deviation is the measure of the amount of variation or dispersion of a set of values.
- A low standerd deviation indicates the values tends to be close to mean (also called the expected value) of the set, while a high standerd deviation indicates that the values are spread out over a wider range.
- Mathematically, $\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$

where, $\sigma$ = population standerd deviation.
$N$ = the size of the population
$x_i$ = each value from the population

$\mu$ = the population mean.

"Calculating standerd deviation on previous lists."

np.std (sec-a), np.std (sec-b) # calculate and show:
# (3.862..., 23.180...)

## 4. Variance

- Variance is a measure of how data points differ from a mean.
- According to layman, a variance is a measure of how far a set of data (numbers) are spread out from their mean (average) value.
- Mathematically,

$$\text{Var} = \sigma^2 = \frac{\sum (x_i - \bar{x})^2}{N}$$

"Calculating variance on previous lists."

np.var (sec-a), np.var (sec-b) # calculate variance and
# display: (14.91..., 537.33...)

"Calculating variance and standerd deviation on real

world data. If the variance and standerd deviation is high it is a scatterd data but as a data scientist, we always perfer less scattered data for our work. '''

# Example :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns  # import necessary libraries.

dataset = pd.read_csv("Students.csv")

dataset.head(3)

dataset["Age"].var()  # Calculate varrance.

dataset["Age"].std()  # Calculate standerd devration.

# Pot a column vs data graph for visualization.
sns.histplot(x="Age", data=dataset, bins=np.arrange(0,100,10))
plt.show()
```

''' If we don't want them to calculate all we have another method which will show {count, mean, std, min, 25%, 50%, 75%, max} of all the numerical columns '''

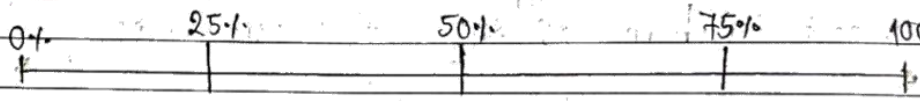
```
dataset.describe()
```

# 3. Percentiles and Quartiles

## Percentiles

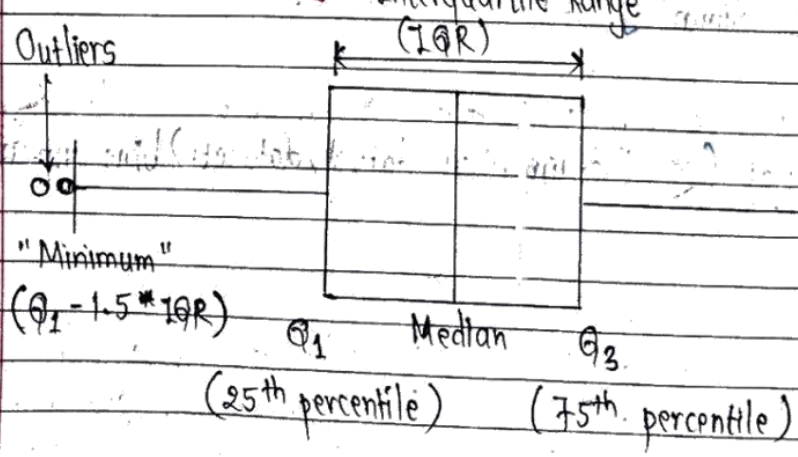
- Percentiles are used in statistics to give you a number that describes the value that a given percent of the values are lower than. It ranges from 0% to 100%.

## Quartiles




- The equal four divisions of percentiles made are called quartiles.

## Interquartile Range (IQR)




From diagram;

$$IQR = Q_3 - Q_1$$

NOTE:

- Percentage vs Percentile : Percentage works on fixed defined value whereas percentile works on undefined base value. Also, percentage shows obtained value whereas percentile shows obtained ranking.

- Outliers : The data beyond our range.

eg 2, 3, 4, 5, 6, 2, 3, 4, 5, 6, 100, 200, 500

data in range — outliers

From diagram; the outliers are values beyond Minimum and Maximum which can be calculated using IQR.

""" Using Percentiles and Quartiles to solve real world problems. """

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

dataset = pd.read_csv("Students.csv")

dataset.head(3) # Display first three rows

dataset.isnull().sum() # Display total number of null
    # values present in each column in dataset

dataset["Age"].fillna(dataset["Age"].mean(), inplace
    = True)
```

```
# filling up the null values by mean of dataset
# column.

# Calculating percentiles.
np.percentile(dataset["Age"], 50)
np.percentile(dataset["Age"], 25)
np.percentile(dataset["Age"], 75)

# Displaying all percentiles 0%, 25%, 50%, 75%
# and 100% of all numerical columns at once.
dataset.describe()

"""If you find a large gap between min and 25%
or max and 75% there exist outliers. No. of
outliers is directly proportional to the gap."""

# We can see outliers by using a box plot.
sns.boxplot(x="Age", data=dataset)
plt.show

"""Percentiles are used to identify and remove outliers
which working with machile learning algorithms."""
```

## 4. Measures of Shape - Skewness

### Skewness

- Skewness measures the asymmetry of the distribution.

- A skewness value of 0 indicates a perfectly symmetrical distribution.

- Positive skewness indicates that the distribution is skewed to the right (i.e. the tail is longer on the right), while negative skewness indicates a left skew (i.e. the tail is longer on the left).

- Mathematically,

$$\text{Skewness} = \frac{\Sigma (x_i - \bar{x})^3}{(N-1) \cdot \sigma^3}$$

## Frequency and Cumulative Distribution

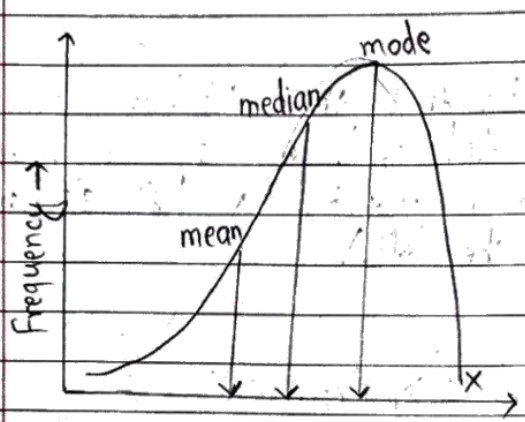
- A frequency distribution is a table that shows the number of occurances of different values in a dataset.

- A cumulative distribution shows the accumulation of frequencies up to a certain point. It is obtained by adding up the frequencies as you move through the values.

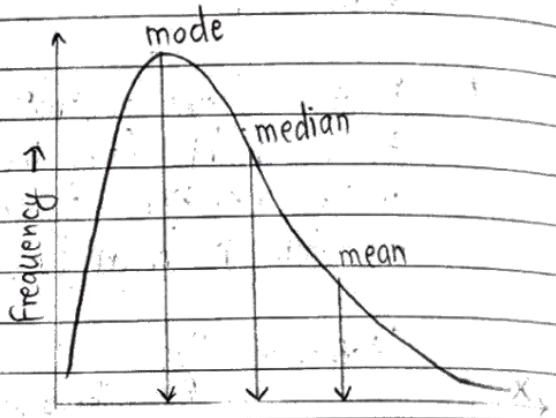
| Value Range | Frequency | Cumulative |
|---|---|---|
| 10 - 20 | 5 | 5 |
| 20 - 30 | 8 | 5+8 = 13 |
| 30 - 40 | 12 | 13+12 = 25 |
| 40 - 50 | 6 | 25+6 = 31 |

- Histogram is used for visualization.

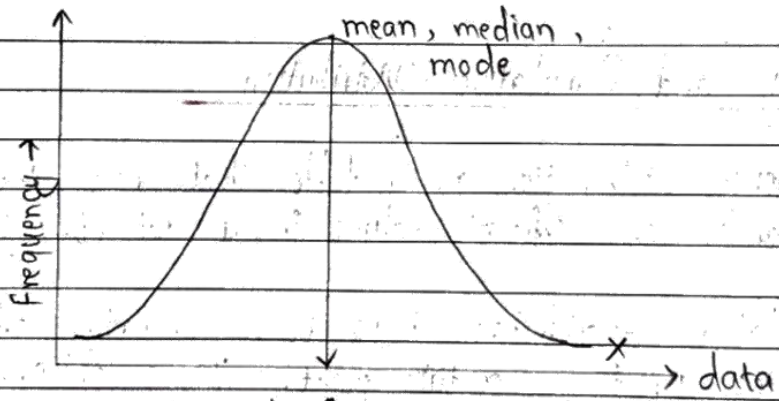
## Skewness Charts.




(a) Negatively skewed




(b) Positively skewed.




(c) Normal (no skew)

From diagrams :

(a) When negatively skewed → mean < median < mode
(b) When positively skewed → mode < median < mean
(c) When no skew → mean = mode = median

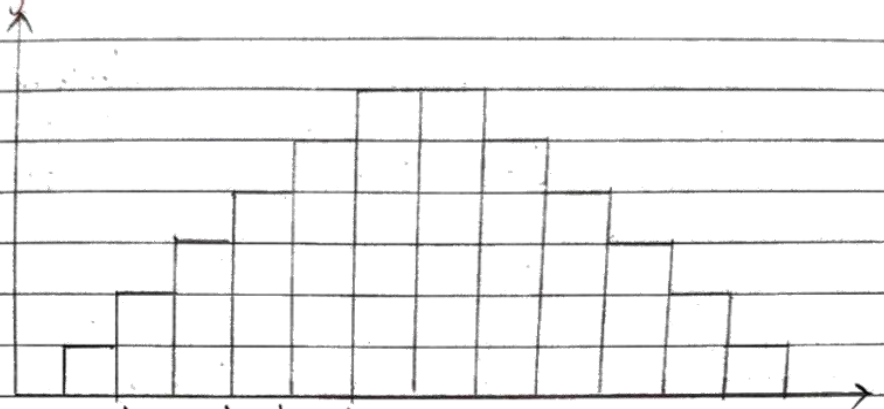
## Working Practically in Jupyter :

```
import pandas as pd
```

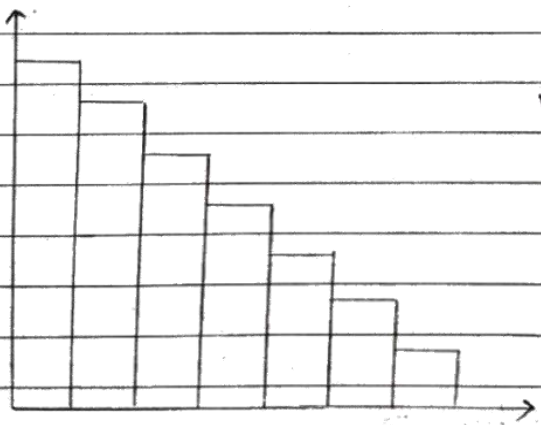
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

dataset = pd.read_csv("Students.csv")

dataset.head(3)

dataset["Age"].skew() # Returns skewness of column
''' If skewness > 0 → positively skewed
    If skewness < 0 → negatively skewed
    If skewness = 0 → no skew. '''

# Plot a histogram for visualization.
sns.histplot(x="Agen", data = dataset)
plt.show()


''' Visualization by creating your own data. '''

data = np.random.normal(0, 100, 100) # Create a random
                                     # array of data.

data # Display created data.

df = pd.DataFrame({"x": data}) # Create a column x where
                               # all data is placed in different rows.
df["x"].skew() # Gives skewness

# Create a histogram for visualization.
sns.histplot(x="x", data = df)
```

```
plt.show()

''' The normal non-skew data can be created as follows:'''

data = [2,3,3,4,4,4,5,5,5,5,6,6,6,6,6,7,7,7,7,7,
        7,7,8,8,8,8,8,9,9,9,9,10,10,10,11,
        11,12]  # non-skew data.

df = pd.DataFrame({"x": data}) # Create a column
   # named x under which all data is placed.

df["x"].skew()  # 0 skew

# Histogram will be symmetrical.
sns.histplot(x="x", data=df, bins=[2,3,4,5,6,7,8,9,
                                   10,11,12,13])
plt.show()

# Mean, Median, Mode will be same.
df["x"].mean(), df["x"].median(), df["x"].mode
    ()[0]    # (7.0, 7.0, 7)


''' Mean, Median and Modes will not be same in
negatively and positively skewed charts (data).
for negatively skewed: mean < median < mode
for positively skewed: mode < median < mean '''
```

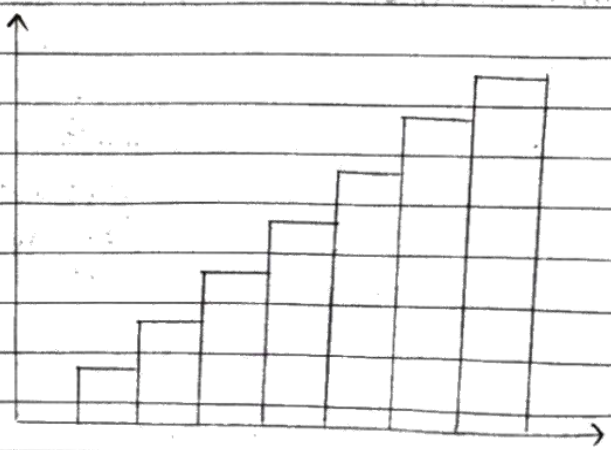
## Symmetric (normal) vs skewed and normal distributions:



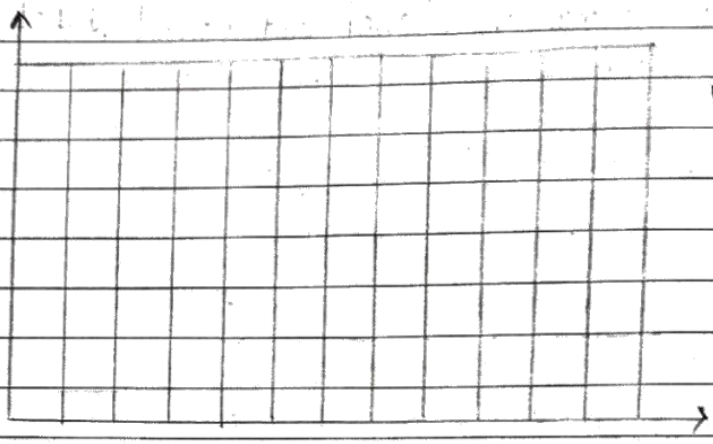
Normal distribution
(unimodal, symmetric, the "bell curve")



Right-skewed distribution
(Positively skewed)

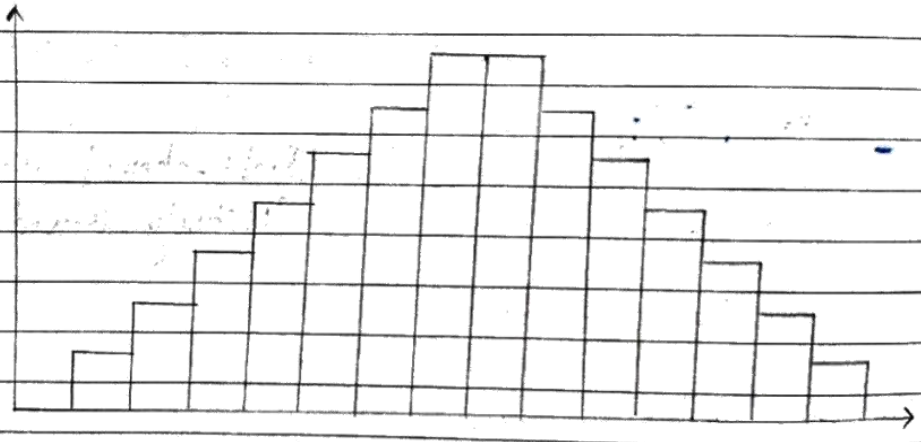


Left-skewed distribution
(Negatively skewed)

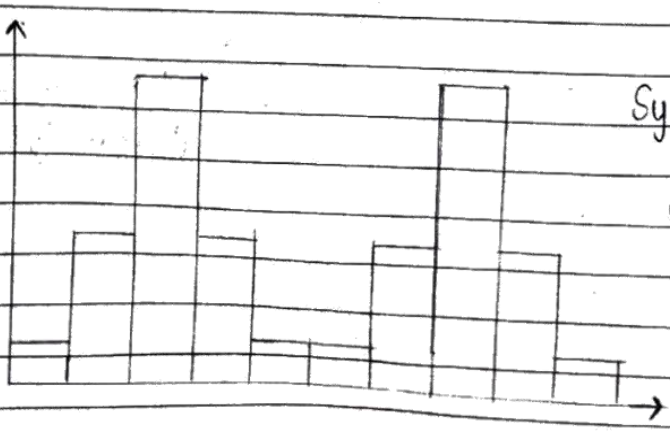


Uniform distribution
(equal spread,
no peaks)

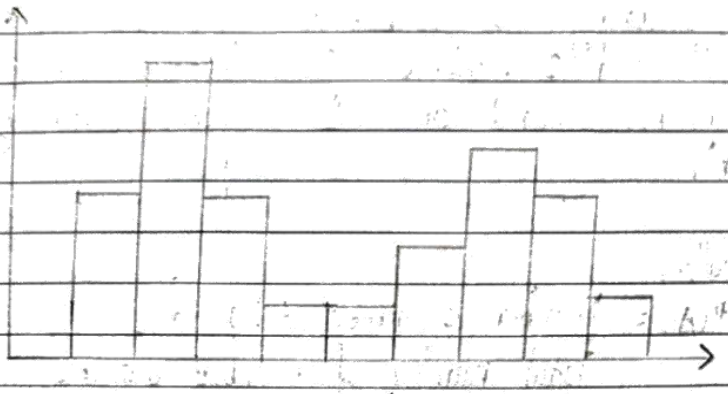
## Unimodal vs Bimodel distributions.



Normal distribution
(unimodal, symmetric, the "bell curve")



Symmetrical bimodel
distribution
(two modes)



Non-symmetric bimodal distribution
(two nodes)

## 5. Probability

**Random Variables :** A Random variable x is a function that assigns a real number to each outcome in the sample space of a random experiment.

1. **Descrete Random Variable :** A random variable that takes on a countable number of distinct values. e.g. dice roll

2. **Continuous Random Variable :** A random variable that can take an any value within a given range or interval. e.g. height of people in a sample.
This doesn't have any fix no. of possible outcomes.

### Probability

G Probability measures the likelihood of a particular outcome

or event occuring. It is typically expressed as a number between 0 and 1, where 0 indicates impossibility (event will not occur) and 1 indicates certainity (event will occur).

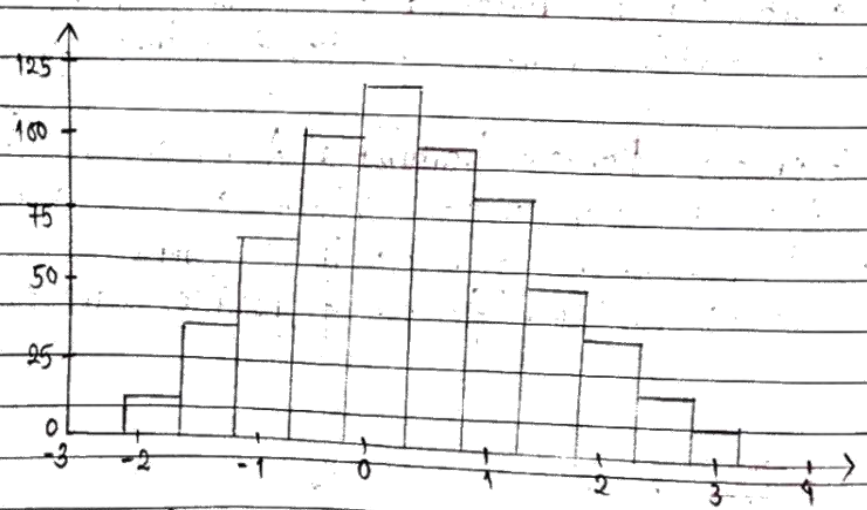
- Mathematically,

$$P(A) = \frac{\text{Number of times A occurs}}{\text{Total number of possible outcomes}}$$

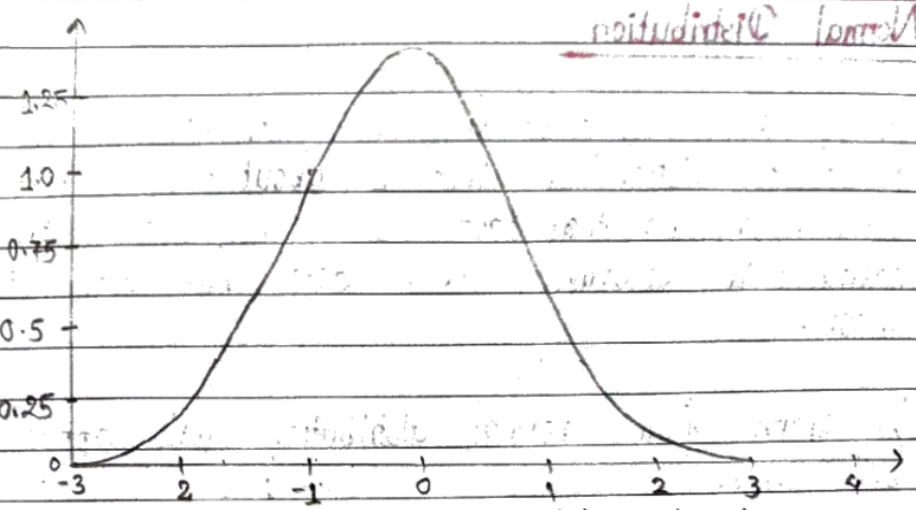
## Probability Distribution

- Probability distributions describe how the probabilities of different outcomes are distributed over the sample space of a random variable.

- Two types of probability distribution are:
  (a) continuous probability distributions
  (b) descrete probability distributions




(a) Discrete probability distribution




(b) Continous probability distribution.

## Probability Distribution Functions

- A probability distribution function is the mathematical function that gives the probabilities of occurrence of different possible outcomes for an experiment.

- Types of probability distribution functions are:
  (a) Probability density function (PDF)
  (b) Probability mass function (PMF)
  (c) Cumulative density function (CDF)

(a) **Probability density function (PDF)** : This function is defined on continous data. Density means total number of data in a particular range.

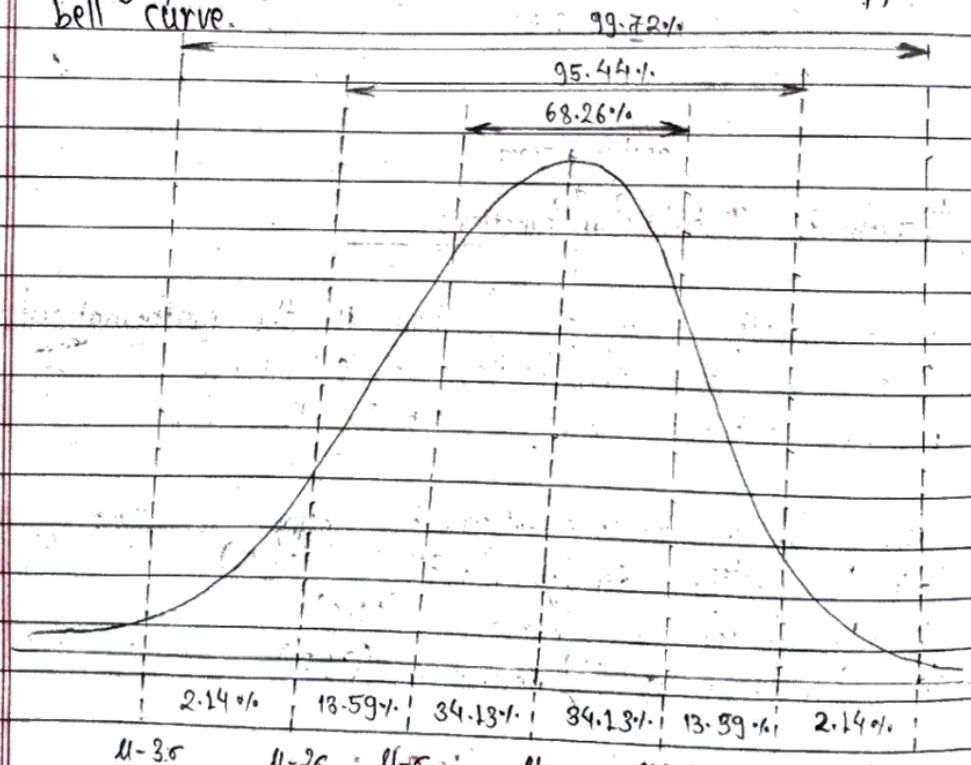
(b) **Probability mass function (PMF)** : This function is defined on descrete data. Mass means frequency of data.

(c) **Cumulative density function (CDF)** : This shows the percentage of data before and after a certain point.

## 6. Normal Distribution

It is known as the Gaussian distribution, is a probability distribution that is symmetric about the mean, showing that data near the mean are more frequent in occurrence than data far from the mean.

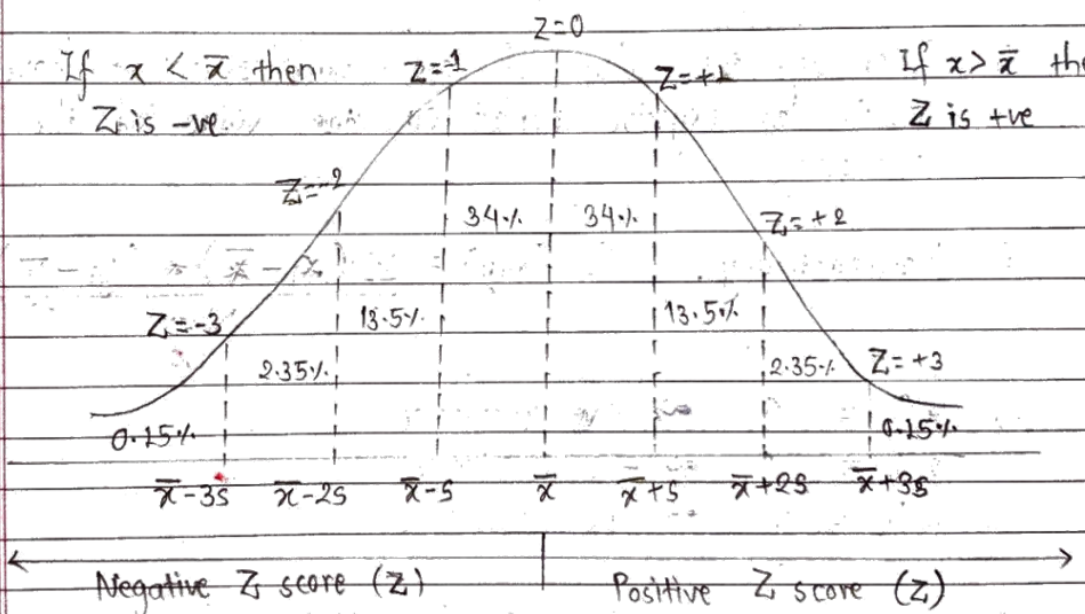
In graph form, normal distribution will appear as a bell curve.




Mathematically, $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

# Standard Normal Distribution

- The standard normal distribution, also known as the Z distribution or Z-score, is a special case of the normal distribution.
- mean ($\mu$) of 0 and a standard deviation ($\sigma$) of 1.




In diagram; $\bar{x}$ = mean score
$x$ = individual score.

# 7. Covariance and Correlation.

## Covariance

- Covariance signifies the direction of the linear relationship between the two variables. By direction we mean if the variables are directly proportional or inversely proportional to each other.
- Increasing the value of one variable might have a positive or negative impact on the value of the other variable.
- Mathematically, $Cov(x,y) = \frac{\sum (x_i - \bar{x}) * (y_i - \bar{y})}{N}$
- $x\uparrow \Rightarrow y\uparrow$ ⇝ +ve covariance.
  $x\downarrow \Rightarrow y\uparrow$ ⇝ -ve covariance.
  $x\uparrow\downarrow \Rightarrow y$ ⇝ no covariance.
- The range of covariance is $-\infty$ to $\infty$.

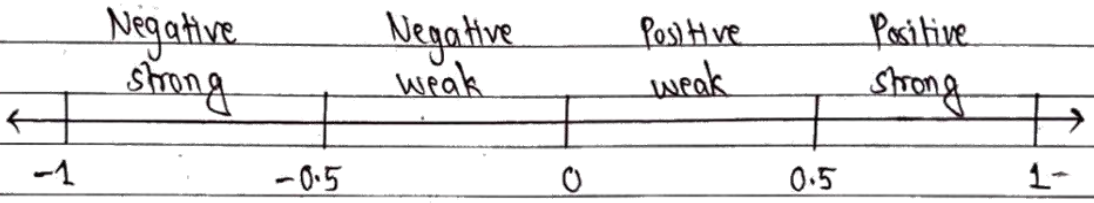
## Correlation

- Correlation is a method of statistical evaluation used to study the strength of a relationship between two, numerically measured, continous variables.
- Mathematically, $Correlation = \frac{Cov(x,y)}{\sigma_x * \sigma_y}$

Where,

cov is covariance
$\sigma x$ is standard deviation of x
$\sigma y$ is standard deviation of y.

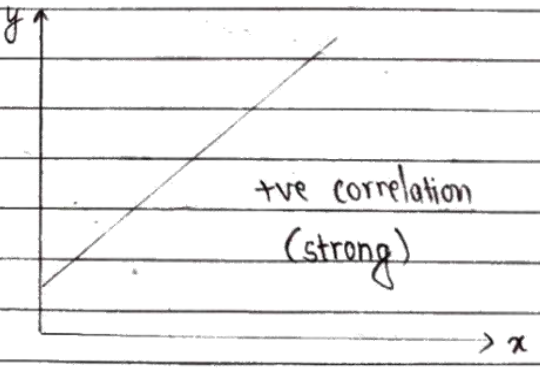
- The range of correlation is -1 to 1 which is converted from range of covariance for better analysis.

- 

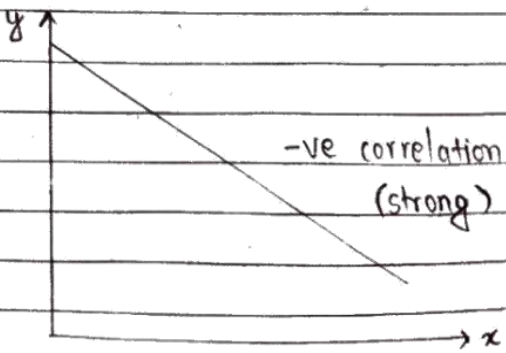
Parts of correlation

- 

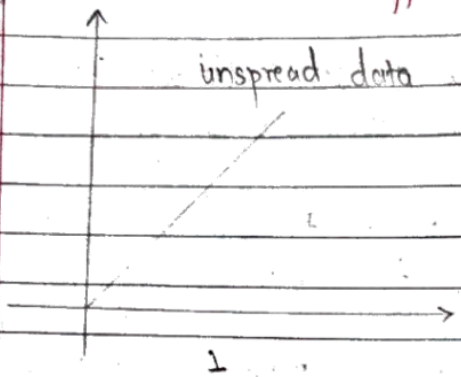
| x | y |
|---|---|
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |
| 4 | 5 |
| 5 | 6 |

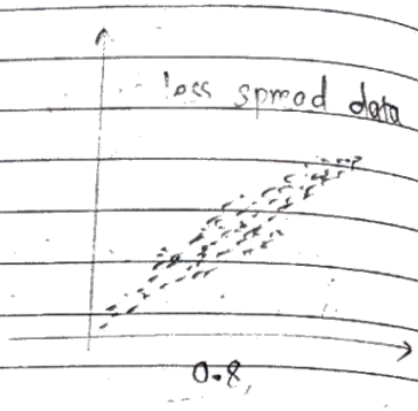


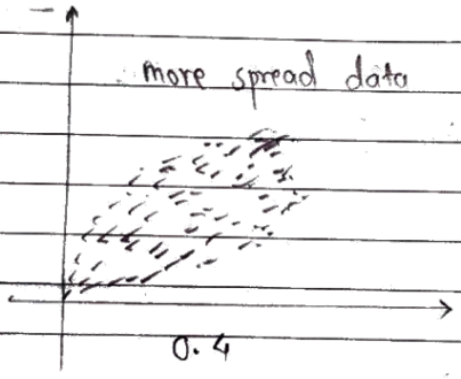

| x | y |
|---|---|
| 1 | 5 |
| 2 | 4 |
| 3 | 3 |
| 4 | 2 |
| 5 | 1 |

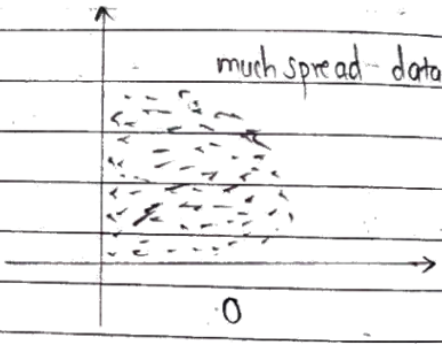


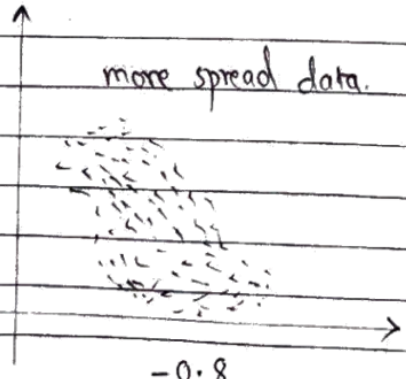
# Pearson Correlation Coefficient:

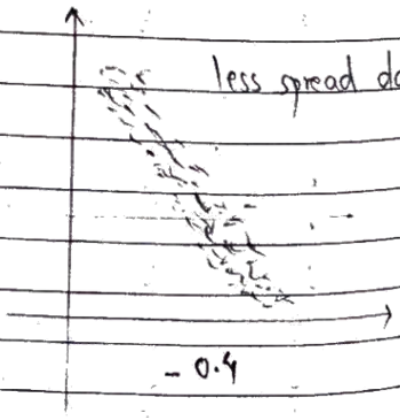


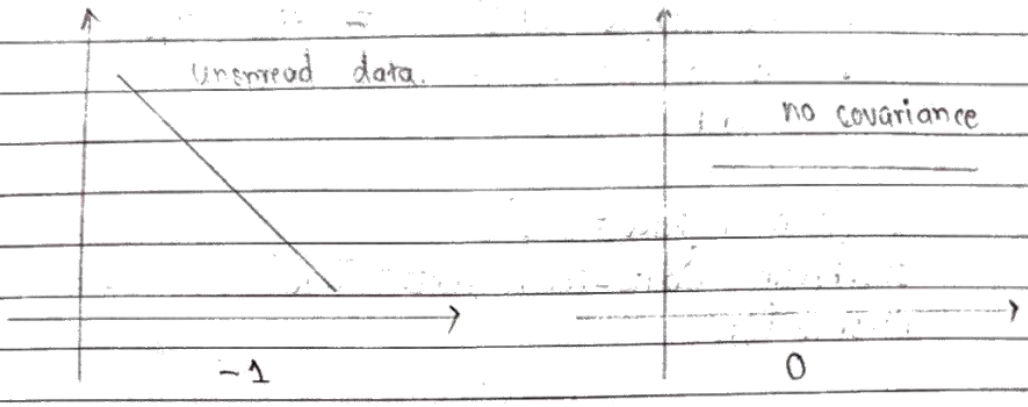






## Working Practically in Jupyter:

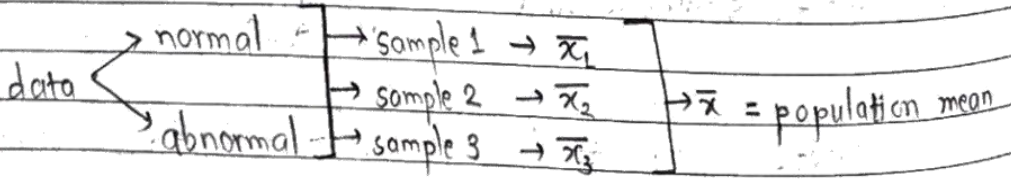
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

dataset = pd.read_csv("Students.csv")

dataset.head(3) # display first three head rows.

dataset.isnull().sum() # displays sum of null values
                       #in each columns.

dataset.info() # You can check data types using this.

data_corr = dataset.select_dtypes(["float64","int64"]).corr()
            # find correlation on selected datatypes.
data_corr # displays what data_corr variable stores.

data_cov = dataset.select_dtypes(["float64","int64"]).cov()
data_cov # For covariance
```

```
plt.figure(figsize=(4,3))    # adjust size
sns.heatmap(data_corr, annot=True)
plt.show()

plt.figure(figsize=(4,3))
sns.heatmap(data_cov, annot=True)
plt.show();
```

# Correlation is more used than covariance in machine learning.

## 8. Central Limit Theorem

The central limit theorem (CLT) states that when plotting a sample distribution of means the means of the sample will be equal to the population mean and the sample distribution will approach normal distribution with variance equal to standard error.




There are few assumption behind the CLT:

- The sample data must be sampled and selected randomly from the population.
- There shouldn't be any multicollinearity in the sampled data which is one sample should not influence the other samples.

- The sample size should be no more than 10% of the population. Generally, sample size greater than 30 ($n > 30$) is considered good.

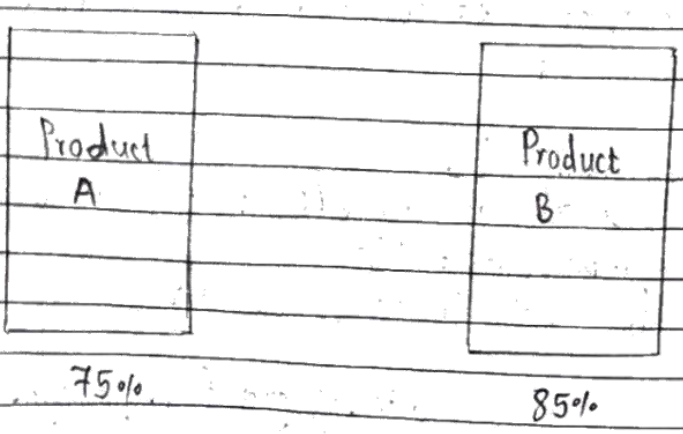
## Working practically in Jupyter:

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

pop_data = [np.random.randint(10,100) for i in range
                                    (10000)]
    # Creates a list of 10000 integers generated randomly.
pop_table = pd.DataFrame({"Pop_data": pop_data})
    # Creates a data frame with column Pop_data and
    # rows pop_data.

np.mean(pop_data) # Find mean of pop_data.

plt.figure(figsize=(5,5))
sns.kdeplot(x="Pop_data", data=pop_table)
plt.show();

sample_mean = []
for no_sample in range(60): # select 60 samples
    sample_data = []
    for data in range(500): # Select 500 data in
                            #each sample
        sample_data.append(np.random.choice(pop_data))
    sample_mean.append(np.mean(sample_data))
```

```
np.mean(sample_mean)

sample_M = pd.DataFrame({"Sample_Mean": sample_mean})

plt.figure(figsize=(5,5))
sns.kdeplot(x="Sample_Mean", data=sample_M)
plt.show();

# We find mean of pop-data and sample_mean is
# nearly equal which proves CLT.
```

## 9. Hypothesis Testing

- It is a part of statistical analysis, where we test the assumptions made regarding a population parameter.
- It is generally used when we were to compare a single group with an external standard and two or more groups with each other.



# Termonology Used in Hypothesis Testing

- **Null Hypothesis :** is a statistical theory that suggests there is no statistical significance exists between the populations. It is denoted by $H_0$ and read as H-naught.

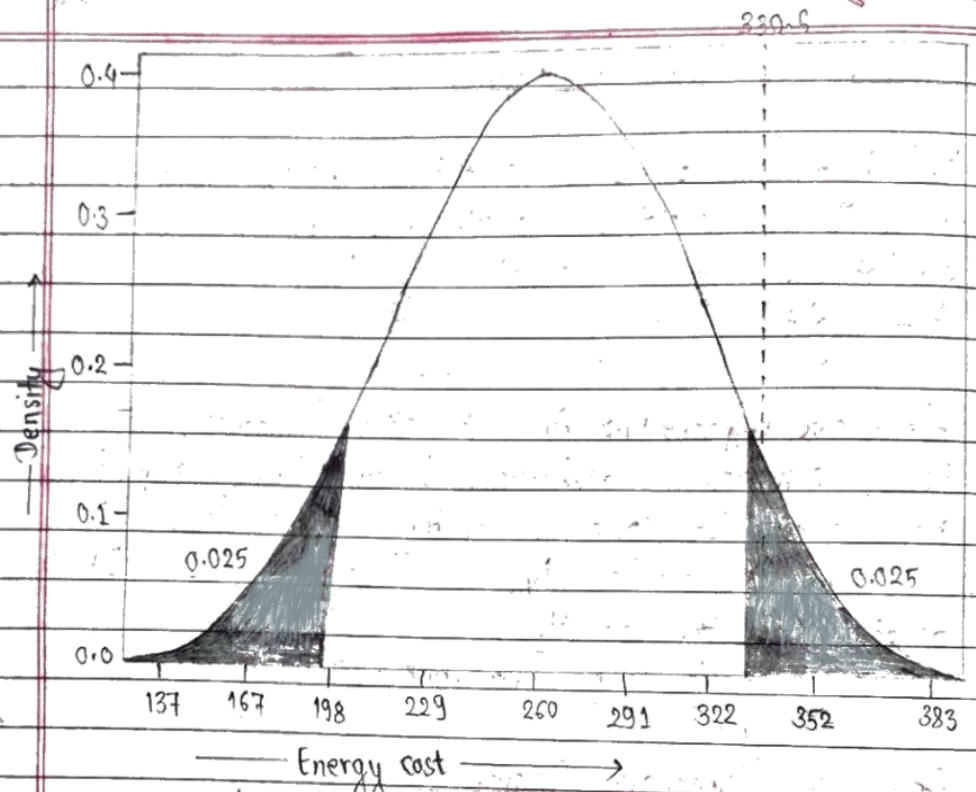
- **Alternative Hypothesis :** An alternative hypothesis suggests there is a significant difference between the population parameters. It could be greater or smaller. Basically, it is the contrast of the Null Hypothesis. It is denoted by $H_a$ or $H_1$

## Steps of Hypothesis Testing

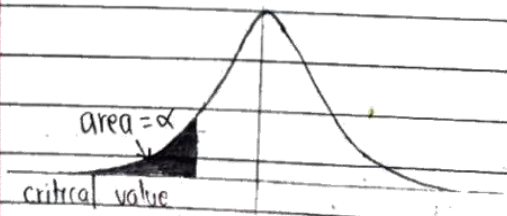
- State null ($H_0$) and alternative ($H_1$) hypothesis.
- Choose level of significance ($\alpha$).
- Find critical values.
- Find test statistic
- Draw your conclusion.

- **State null ($H_0$) and alternative ($H_1$) hypothesis :**
  $H_0$ must always contains equality ($=$).
  $H_1$ always contain differences ($\neq$, $>$, $<$).

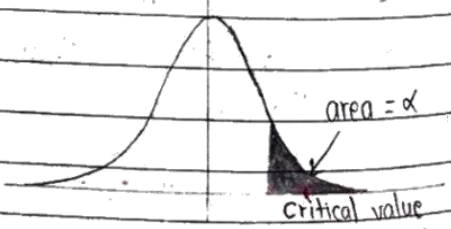
- **Choose level of significance :**
  It is a fixed probability of wrongly rejecting a True Null Hypothesis.
  It is denoted by $\alpha$ (alpha).



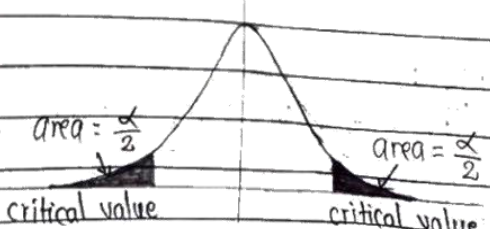

Visual representation of level of significance




left tailed (worse)




right tailed (better)




two tailed (non-directional)

Reject $H_0$ ($H_a$)

Do not reject $H_0$ ($H_a$)

- **Find critical values:**
  Using the test statistic find critical values like z value, t value, etc.

- **Find test statistic:**
  Find the required test like: Z-test, T-test, Chi-square test, etc. These test gives critical values.

Z test → population mean
→ population standard deviation
→ sample ≥ 30
→ sample mean

T test → population mean
→ sample standard deviation
→ sample < 30
→ sample mean

Chi-square test → checks goodness and relations between two variables (independance)

- **Draw your conclusion:**
  Conclude which one to be choosen either $H_0$ or $H_a$ for better performance.

## 1. Z Test

↳ Data should be normally distributed

↳ Given:
- population mean
- population standard deviation
- sample mean
- no. of sample ≥ 30

Mathematically,

$$Z_{test} = \frac{\bar{x} - \mu}{\left(\frac{\sigma}{\sqrt{n}}\right)}$$

where, $\bar{x}$ = sample mean
$\mu$ = population mean
$\sigma$ = population standard deviation
$n$ = no. of sample

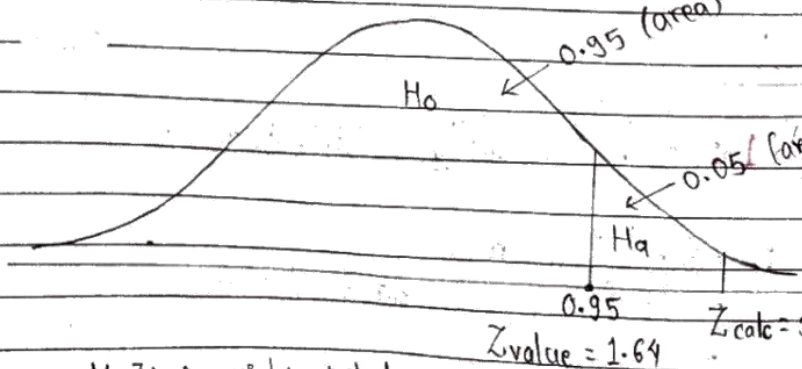
**Example 1:**

A teacher claims that the mean score of students in his class is greater than 82 with a standard deviation of 20. If a sample of 81 students was selected with a mean score of 90.

**Solution:**

$H_o : \mu \neq 82$ (opposing / null hypothesis)
$H_a : \mu > 82$ (supporting / alternative hypothesis)
Assume that $\alpha = 0.05$
Given, $\bar{x} = 90$
$\sigma = 20$
$n = 81$

Total area under curve is always 1 being a probability distribution




# It is right tailed so, Zvalue should be of $(1-\alpha)$.

Date
Page 39

Z value = Sum of row and column z values of $(1-\alpha)$ area from same tailed z table (choose close value)

= 1.6 + 0.04 = 1.64

$$Z_{calc} = \frac{\bar{x} - \mu}{\left(\frac{\sigma}{\sqrt{n}}\right)} = \frac{90 - 82}{\left(\frac{20}{\sqrt{81}}\right)} = \frac{8 \times 9}{20} = 3.6$$

$\because Z_{calc} > Z_{value}$, the teacher was right.

## Working practically in Jupyter:

```
import scipy.stats as st # library with constants
                         # stats has statistical analysis
import numpy as np

sample_mean = 90
population_mean = 82
population_std = 20
n = 81
alpha = 0.05   # Assume

z_calc = (sample_mean - population_mean) / (population_std /
                                            np.sqrt(n))
z_calc

z_table = st.norm.ppf(1-alpha) # norm contains functions
                               # related to normal distribution
z_table

if z_table < z_calc :
    print("Teacher is right")
else:
    print("Teacher is wrong")
```

Date
Page 40

## Example 2:

### Scenario:

Imagine you work for an e-commerce company, and your team is responsible for analyzing customer purchase data. You want to determine whether a new website design has lead to a significant increase in the average purchase amount compared to the old design.

### Data:

You have collected data from a random sample of 30 customers who made purchase on the old website design and 30 customers who made purchase on the new website design. You have the sample means, sample standard deviations, and sample sizes for both groups.

- Old design data = [45.2, 42.8, 38.9, 48.5, 41.0, 44.6, 40.5, 42.7, 39.8, 41.4, 44.3, 39.7, 42.1, 40.6, 43.0, 42.2, 41.5, 39.6, 44.0, 43.1, 38.7, 43.9, 42.0, 41.9, 42.8, 43.7, 41.3, 40.9, 42.5, 41.6]

- New design data = [48.5, 49.1, 50.2, 47.8, 48.7, 49.9, 48.0, 50.5, 49.8, 49.6, 48.2, 48.9, 49.7, 50.3, 49.4, 50.1, 48.6, 48.3, 49.0, 50.0, 48.4, 49.3, 49.5, 48.8, 50.6, 50.4, 48.1, 49.2, 50.7, 50.8]

- Population standard deviation = 2.5

### Solution:

$H_0$ : new website = old website
$H_a$ : new website > old website

Assuming no change in population mean i.e. $\mu_{new} = \mu_{old}$
Given, $\sigma = 2.5$
$n = 30$
We need, $\bar{x}_{new}$ and $\bar{x}_{old}$ → calculate from given samples.
Now, assuming $\alpha = 0.05$ :
Find $z_{value}$ from z table, $z_{value} = 1.645$
And,

$$z_{calc} = \frac{(\bar{x}_{new} - \bar{x}_{old}) - (\mu_{new} - \mu_{old})}{\left(\frac{\sigma}{\sqrt{n}}\right)}$$

Since, $\mu_{new} = \mu_{old}$

$$\therefore z_{calc} = \frac{\bar{x}_{new} - \bar{x}_{old}}{\left(\frac{\sigma}{\sqrt{n}}\right)} = 16.110$$

Hence, you can draw your conclusion that $H_a$ is right i.e. new website is better than old website.

## Working practically in Jupyter :

```
import scipy.stats as st
import numpy as np

Old_design_data = np.array([<Old design data>])
New_design_data = np.array([<New design data>])
        # np.array() converts lists to array for fast
        # calculation


pop_std = 2.5
n_sp = len(New_design_data)   # 30
alpha = 0.05 #Assume
```

```
mean_new = np.mean (New_design_data)
mean_old = np.mean (Old_design_data)

z_cal = (mean_new - mean_old) / (pop_std / np.sqrt (n_sp))
z_cal

z_table = st.norm.ppf (1- alpha)
z_table

if z_calc > z_table :
    print (" New website is better than old.")
else :
    print (" New website is same as old.")
```

## 2. T Test

- Data should be normally distributed.
- Given :
  - population mean
  - sample mean
  - sample standard deviation
  - no. of sample < 30
- Mathematically,

$$t_{test} = \frac{\bar{x} - \mu}{\left(\frac{s}{\sqrt{n}}\right)}$$

Where, $\bar{x}$ = sample mean
$\mu$ = population mean

$s$ = standard deviation of sample
$n$ = no. of sample

## Example 1:

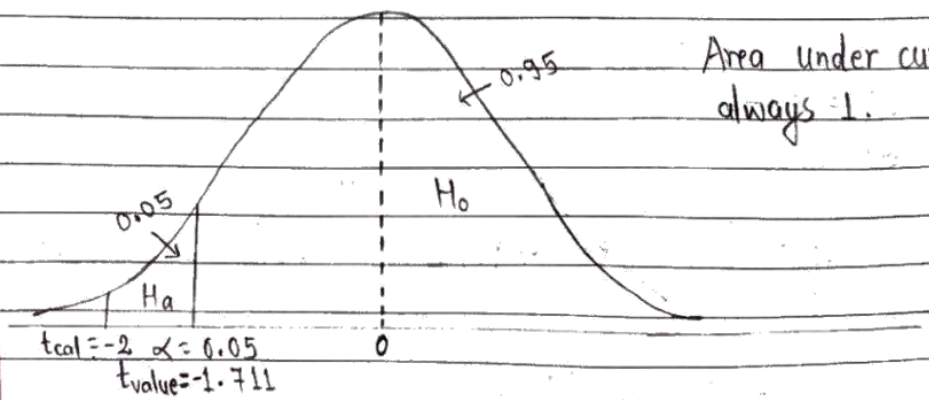
A manufacturer claims that the average weight of a bag of potato chips is 150 grams. A sample of 25 bags is taken and the average weight is found to be 148 grams, with a standard deviation of 5 grams. Test the manufacturer's claim using a one-tailed t-test with a significance level of 0.05.

## Solution:

$H_0$: $\mu = 150$ gm (null hypothesis)
$H_a$: $\mu < 150$gm (alternative hypothesis)

Given,

$\alpha = 0.05$
$\mu = 150$
$\bar{x} = 148$
$s = 5$
$n = 25$




# It is left tailed so, tvalue should be of $\alpha$.

degree of freedom, df = n-1 = 25-1 = 24.

tvalue = the value that is present in df row and $\alpha$ column in t-table.
= -1.711 (-ve sign indicates it is left tailed)

$$t_{cal} = \frac{\bar{x} - \mu}{\left(\frac{s}{\sqrt{n}}\right)} = \frac{148 - 150}{\left(\frac{5}{\sqrt{25}}\right)} = \frac{148 - 150}{1} = -2$$

Hence, the claims of the company is wrong as we find $t_{cal} < t_{value}$.

## Working practically in Jupyter:

```
import scipy.stats as st
import numpy as np

alpha = 0.05
pop_mean = 150
sam_mean = 148
sam_std = 5
n = 25

df = n -1
df

t_calc = (sam_mean - pop_mean) / (sam_std / np.sqrt(n))
t_calc

t_table = st.t.ppf(alpha, df)
t_table
```

if t_calc > t_table :
    print ("The claim of manufracturer is right")
else:
    print ("The claim of manufracturer is wrong")

## Example 2:

A company wants to test whether there is a difference in productivity between two teams. They randomly select 20 employees from each team and record their productivity scores. The mean productivity for Team A is 80 with a standard deviation of 5, while the mean productivity score for Team B is 75 with a standard deviation of 6. Test at a 5% level of significance whether there is a difference productivity between two teams.

## Solution:

$H_0$ : Productivity of A − Productivity of B = 0
$H_a$ : Productivity of A − Productivity of B ≠ 0 (two tailed)
Given,

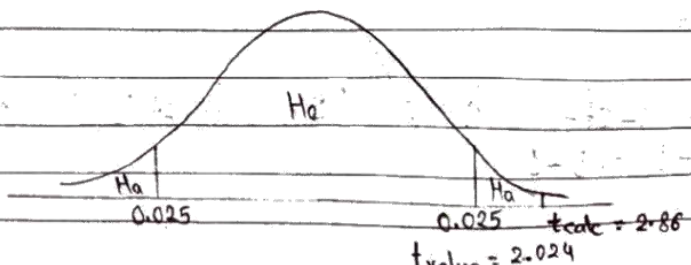
$2\alpha = 5\% = 0.05 \Rightarrow \alpha = 0.025$ [∵ two tailed]

$n_A = n_B = 20$

$\overline{x}_A = 80$ and $S_A = 5$

$\overline{x}_B = 75$ and $S_B = 6$

Assume that $\mu_A = \mu_B$.



degree of freedom, $df = n_A - 1 + n_B - 1$

$= 20 - 1 + 20 - 1$

$= 38$

tvalue = -2.024 [From t-table]

$$t_{calc} = \frac{(\bar{x}_A - \bar{x}_B) - (\mu_A - \mu_B)}{\sqrt{\frac{S_A^2}{n_A} + \frac{S_B^2}{n_B}}}$$

Since, $\mu_A = \mu_B$

$$\therefore t_{calc} = \frac{\bar{x}_A - \bar{x}_B}{\sqrt{\frac{S_A^2}{n_A} + \frac{S_B^2}{n_B}}} = 2.86$$

Since, tcalc > tvalue there is some difference between productivity of A and productivity of B.

Working practically in Jupyter:

```
import numpy as np
from scipy.stats import t

level_of_significance = 5/100   # 5%
nA = nB = 20
mean_A = 80
std_A = 5
mean_B = 75
std_B = 6

alpha = level_of_significance / 2   # It is two tailed
df = nA - 1 + nB - 1
```

Date
Page 43

```
t_table = t.ppf (alpha, df)   # Testing for left tail
t_calc = (mean_A - mean_B) / (np.sqrt ((np.square (std_A)
        /nA) + (np.square (std_B)/nB))
print ("t_table =", t_table, "and t_calc =", t_calc)

if t_calc > t_table:
    print ("Productivity of A != Productivity of B")
else:
    print ("Productivity of A = Productivity of B")
```

## Example 3:

A company wants to test whether a new training program improves the typing speed of its employees. The typing speed of 20 employees was recorded before and after the training program. The data is given below. Test at a 5% level of significance whether the training program has an effect on the typing speed of the employees.

- Before : 50, 60, 45, 65, 55, 70, 40, 75, 80, 65, 70, 60, 50, 55, 45, 75, 60, 50, 65, 70

- After : 60, 70, 55, 75, 65, 80, 50, 85, 90, 70, 75, 65, 55, 60, 50, 80, 65, 55, 70, 75

## Solution:

$H_0$ : Speed before training − Speed after training = 0
$H_a$ : Speed before training − Speed after training ≠ 0

Given,

$\alpha = 5\% = 0.05 \Rightarrow \frac{\alpha}{2} = 0.025$ [two tailed test]

$n = 20$

degree of freedom, $df = n - 1 = 20 - 1 = 19$

Assume that $\mu_{after} = \mu_{before}$ [$\because$ not given]

Calculate $\bar{x}_{before}$, $\bar{x}_{after}$, $S_{before}$ and $S_{after}$ from the given sample data.

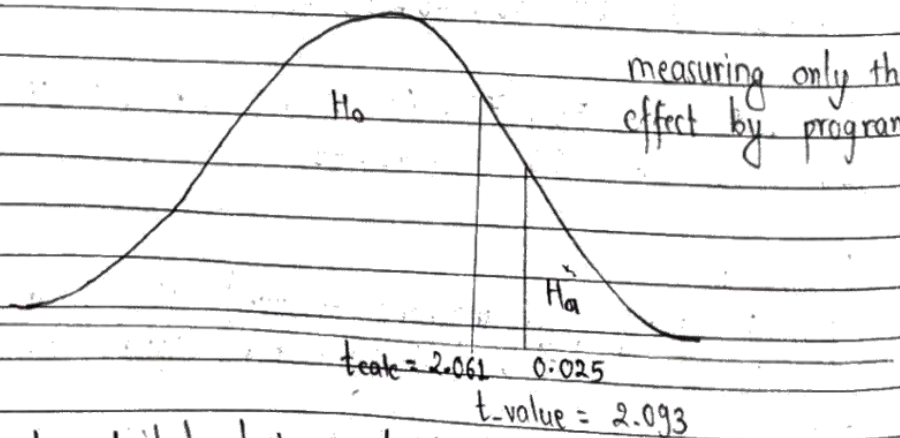
t-value = 2.093 [from t-table]

$$t\text{-calc} = \frac{(\bar{x}_{after} - \bar{x}_{before}) - (\mu_{after} - \mu_{before})}{\sqrt{\frac{S_{after}^2}{n_{after}} + \frac{S_{before}^2}{n_{before}}}}$$

$\because \mu_{after} = \mu_{before}$ and $n_{after} = n_{before} = n$

$$\therefore t\text{-calc} = \frac{\bar{x}_{after} - \bar{x}_{before}}{\sqrt{\frac{S_{after}^2 + S_{before}^2}{n}}} = 2.061$$

Clearly, $2.061 < 2.093$ i.e. t-calc < t-value so, the training program doesn't have any effect on typing speed of employees.




measuring only the +ve effect by program.

# two tailed but analyzing only right tail.

## Working practically in Jupyter :

```
import numpy as np
from scipy.stats import t

before = np.array([<before data>])
after = np.array([<after data>])
level_of_significance = 5 / 100   # 5%
n = 20

alpha = level_of_significance / 2   # It is two tailed
df = n-1

mean_before = np.mean(before)
mean_after = np.mean(after)
std_before = np.std(before)
std_after = np.std(after)

t_table = t.ppf(1-alpha, df)  # Testing for right tail
t_calc = (mean_after - mean_before) / np.sqrt((np.
        square(std_after) + np.square(std_before))/n)
print(f"t-table = {t_table} and t-calc = {t_cal}")

if t_calc > t_table:
    print("The training program is effective")
else:
    print("The training program is not effective")
```

## 3. Chi Square Test

- Applicable in two types of situations :
  i. Goodness
  ii. Independance

- Mathematically, $$X^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

  where, $X^2$ = chi square
  $O_i$ = observed value
  $E_i$ = expected value

**Example 1 :**

A fair dice is rolled 120 times and the following results are obtained :

Face 1 : 22 times
Face 2 : 17 times
Face 3 : 20 times
Face 4 : 26 times
Face 5 : 22 times
Face 6 : 13 times

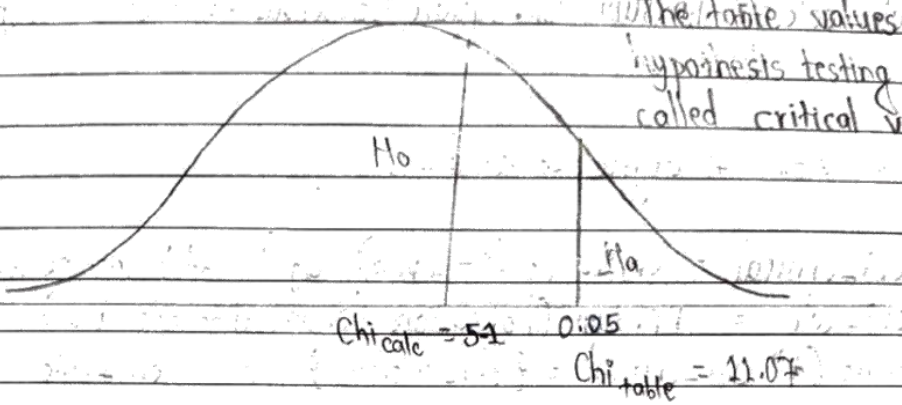
Test at 5% level of significance whether the dice is fair.

**Solution :**

$H_0$ : dice is fair
$H_a$ : dice is not fair

In dice, number of faces, $n = 6$
Degree of freedom, $df = n - 1 = 6 - 1 = 5$

level of significance, $\alpha = 5\% = 0.05$




The expected value for each face when a dice is rolled 120 times is 20 for each.
The observed values are given above.

$Chi_{table}$ = the value that is present in df row and $\alpha$ column in chi square table.
= 11.07

$$Chi_{calc} = \sum \frac{(O_i - E_i)^2}{E_i} = 5.1$$

Clearly $Chi_{calc} < Chi_{table}$ so, the dice is fair i.e. Ho is right.

## Working practically in Jupyter

```
import numpy as np
from scipy.stats import chi2

observed = np.array([22, 17, 20, 26, 22, 13])
expected = np.array([20, 20, 20, 20, 20, 20])
```

```
alpha = 0.05  # Given level of significance is 5%
n=6  # Total number of possibilities when a dice is
     # rolled

df = n-1  # degree of freedom

chi_critical = chi2.ppf (1-alpha), df)  # Right tailed
chi_cale = np.sum (np.square (observed - expected) / expected)
print (f" chi_critical = {chi_critical} and chi_calc = {chi_calc}")

if chi_cale > chi_critical :
    print ("Dice is not fair")
else :
    print ("Dice is fair")
```

## Example 2

A study was conducted to investigate whether there is a relationship between gender and the perferred genre of music. A sample of 235 people was selected, and the data collected is shown below. Test at a 5% level of significance whether there is significant association between gender and music preference.

| | Pop | Hip Hop | Classical | Rock |
|---|---|---|---|---|
| Male | 90 | 45 | 25 | 10 |
| Female | 35 | 30 | 20 | 30 |

## Solution :

$H_0$ : No association

$H_a$ : Has association

Given:
no. of sample, n = 235
level of significance, $\alpha$ = 5% = 0.05
no. of rows, rows = 2
no. of columns, columns = 4
Now,

$$df = (rows - 1) * (columns - 1)$$
$$= (2-1) * (4-1)$$
$$= 3$$

So,

$chi_{critical} = 7.815$ [From chi-square table]

We are given the observed values in the given table. To find the expected value for each observed value proceed as follows:

i.) find the sum of each column and row.

ii.) expected value $(i, j) = \frac{\Sigma R_i \times \Sigma C_j}{\text{No. of sample}}$

So,

$\Sigma R_1 = 120$ and $\Sigma R_2 = 115$

$\Sigma C_1 = 75$, $\Sigma C_2 = 75$, $\Sigma C_3 = 45$ and $\Sigma C_4 = 40$

We've,

observed values = [40, 45, 25, 10, 35, 30, 20, 30]

For expected values:

expected value (1,1) = $\frac{\Sigma R_1 \times \Sigma C_1}{n} = 38.29$

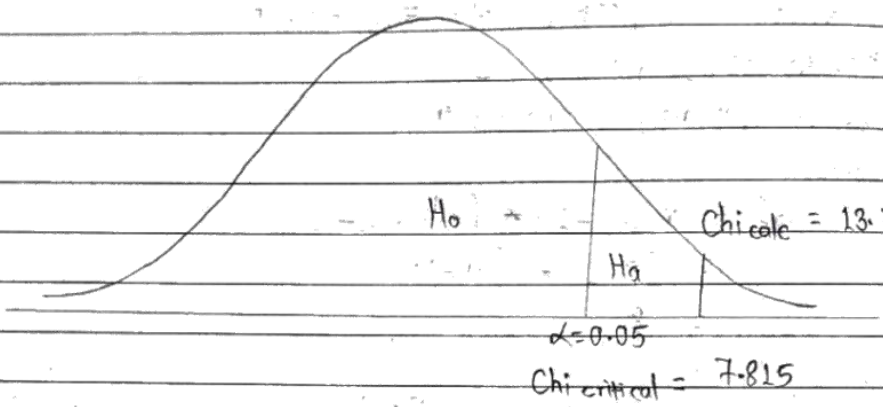
expected value (1,2) = $\frac{\Sigma R_1 \times \Sigma C_2}{n} = 38.29$

Similarly, all the expected values can be calculated.

Finally,

expected values = [38.29, 38.29, 22.97, 20.42, 36.70, 36.70, 22.02, 19.57]

$$chi_{calc} = \sum \frac{(O_i - E_i)^2}{E_i} = 13.78$$




Clearly $chi_{calc} > chi_{critical}$ so, there is association between gender and music preference.

## Working practically in Jupyter:

```
import numpy as np
from scipy.stats import chi2

alpha = 0.05 # Given level of significance is 5%
n = 235 # Sample of 235 people was selected

rows = 2 # from table
columns = 4

row1 = np.array([40, 45, 25, 10])
row2 = np.array([35, 30, 20, 30])

df = (rows-1) * (columns-1) # Calculate degree of freedom
df
```

```
chi_critical = chi2.ppf (alpha, df)
print ("chi-critical =", chi_critical)

sum_row1 = np.sum (row1)
sum_row2 = np.sum (row2)
sum_rows = np.array ([sum_row1, sum_row2]) # Create an
# array of sum of rows.
sum_rows

sum_columns = row1 + row2  # Find sum of all columns at
once in an array.
sum_columns

# Find expected values
expected = []
for i in sum_rows:
    for j in sum_columns:
        expected.append (i*j/n)
expected

# Get observed values
observed = np.append (row1, row2)
observed

chi_calc = np.sum (np.square (observed - expected) / expected)

if chi_calc > chi_critical:
    print ("There is association")
else:
    print ("There is no association")
```