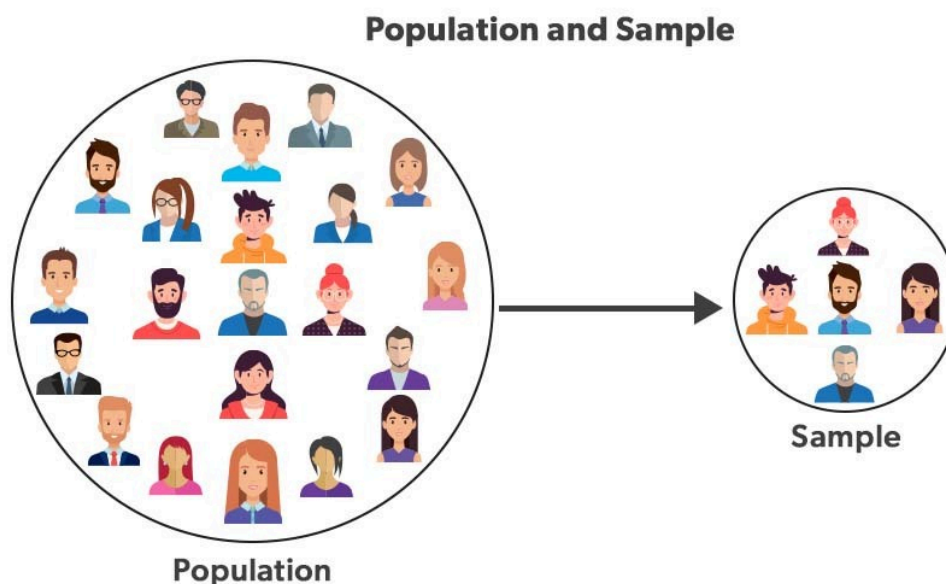


# Math Practice Notes For ML

## Population Data vs. Sample Data



### ♦ 1. Population Data

- **Definition:**  
Population data refers to **all possible data points** or observations in the entire group you're interested in studying.
- **Example:**  
If you're analyzing sales of a product in all branches of a company, and you include **data from every branch**, that's population data.
- **Key Points:**
  - Represents the **whole group**.
  - Usually **difficult** or **expensive** to collect.
  - Used when **complete data** is available or required.
  - Denoted by parameters like  $\mu$  (**mu**) for mean and  $\sigma$  (**sigma**) for standard deviation.

---

## ♦ 2. Sample Data

- **Definition:**  
Sample data refers to a **subset** of the population that is selected for analysis.
- **Example:**  
If you study sales data from **10 out of 100 branches**, that's sample data.
- **Key Points:**
  - Represents a **portion of the population**.
  - Easier, cheaper, and faster to collect.
  - Used to make **inferences about the population**.
  - Denoted by statistics like  $\bar{x}$  (**x-bar**) for mean and **s** for standard deviation.

---

### Difference Between Population and Sample

Feature	Population Data	Sample Data
Scope	Entire group	Subset of the group
Size	Usually large or infinite	Smaller and manageable
Cost & Time	High	Low
Accuracy	More accurate but harder to get	May contain sampling error
Parameters/Statistics	$\mu$ (mean), $\sigma$ (std dev)	$\bar{x}$ (mean), $s$ (std dev)
Usage	For exact results (if possible)	For estimating population parameters

---

### When to Use What?

- **Use Population Data:** When you have access to **all data** (e.g., complete customer database).

- **Use Sample Data:** When population data is too **large** or **unavailable**, and you want to **estimate** or **predict**.

## What is Statistics?

### ◆ Definition:

**Statistics** is the branch of mathematics that deals with the **collection, organization, analysis, interpretation, and presentation** of data.

It helps in **making decisions** based on data, especially when dealing with **uncertainty**.

---

## Types of Statistics

Statistics is broadly classified into two main types:

### 1 Descriptive Statistics

- **Definition:**  
Descriptive statistics deals with **summarizing and organizing** data in a meaningful way.
- **Purpose:**  
To **describe** the main features of a dataset **without drawing conclusions** beyond the data.
- **Common Tools:**
  - Measures of Central Tendency:
    - **Mean, Median, Mode**
  - Measures of Dispersion:
    - **Range, Variance, Standard Deviation**
  - Data Visualization:
    - **Graphs, Charts, Tables, Histograms, Pie Charts**

---

## 2 Inferential Statistics

- **Definition:**  
Inferential statistics makes **predictions or inferences** about a **population** based on a **sample** of data.
- **Purpose:**  
To **generalize, predict, or test hypotheses**.
- **Common Tools:**
  - **Estimation** (e.g., estimating population mean from sample)
  - **Hypothesis Testing** (e.g., t-test, chi-square test)
  - **Confidence Intervals**
  - **Regression Analysis**
  - **ANOVA** (Analysis of Variance)

---

## Subtypes of Descriptive & Inferential Statistics

### ♦ A. Subtypes of Descriptive Statistics

Subtype	Description	Example
Measures of Central Tendency	Shows the center of the data	Mean, Median, Mode
Measures of Dispersion	Shows the spread of the data	Range, Variance, Std. Deviation
Data Distribution	Shows how values are distributed	Frequency, Histogram

---

### ♦ B. Subtypes of Inferential Statistics

Subtype	Description	Example
---------	-------------	---------

Estimation	Estimating population parameters from sample	Population mean
Hypothesis Testing	Testing assumptions/statements	t-test, z-test
Regression	Predicting values based on relationships	Linear regression
ANOVA	Comparing means across groups	Testing 3+ group differences

## Descriptive Statistics

### Descriptive Statistics

#### What is Mean?

##### ♦ Definition:

The **mean** is the average value of a set of numbers. It is calculated by summing all the values and dividing by the total number of values.

##### ♦ Formula:

Mean = Sum of all Values / Number of Values

---

#### Python Code Example

✓ Let's say we have a dataset of product prices.

##### Step-by-step code:

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

# Create a sample dataset using a dictionary
data = {
```

```

"product": ["A", "B", "C", "D", "E", "F", "G"],
"price": [100, 150, 120, 130, 110, 140, 160]
}

# Convert dictionary into DataFrame
df = pd.DataFrame(data)

# Calculate Mean using NumPy
mean_numpy = np.mean(df["price"])

# Calculate Mean using Pandas
mean_pandas = df["price"].mean()

# Print the means
print("Mean (using NumPy):", mean_numpy)
print("Mean (using Pandas):", mean_pandas)

# Plot using Seaborn
sns.histplot(df["price"], bins=5, kde=True, color="orange")

# Add a vertical line showing the mean
plt.axvline(mean_numpy, color='red', linestyle='--', label=f"Mean: {mean_numpy:.2f}")

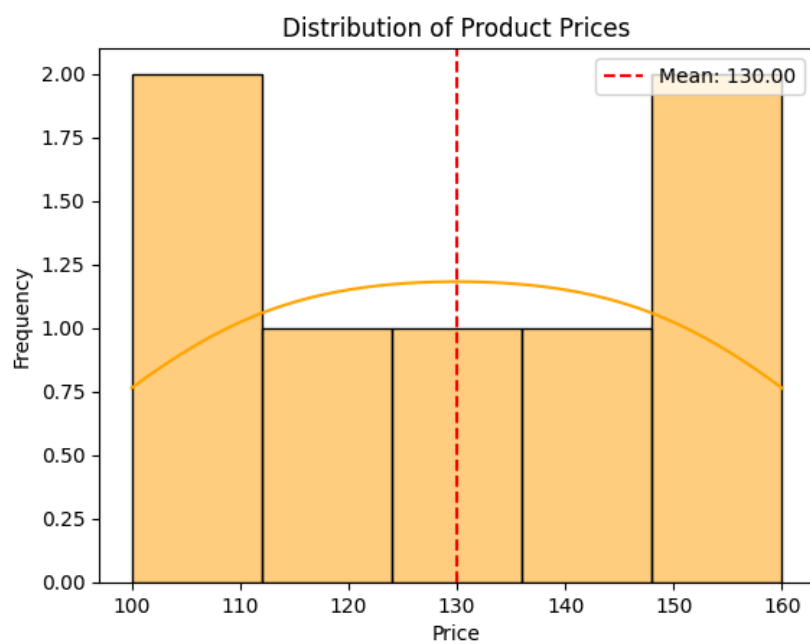
# Add labels and title
plt.title("Distribution of Product Prices")
plt.xlabel("Price")
plt.ylabel("Frequency")
plt.legend()

# Show the plot
plt.show()

```

**Mean (using NumPy): 130.0**

**Mean (using Pandas): 130.0**



## 2. Median

### ♦ Definition:

The **median** is the **middle value** in a dataset when the numbers are arranged in **ascending** or **descending order**.

- If the number of values is **odd**:  
→ Median is the **middle value**.
  - If the number of values is **even**:  
→ Median is the **average of the two middle values**.
- 

### Key Points:

- It **divides** the dataset into **two equal halves**.
  - It is **not affected by outliers** or extreme values.
  - Useful for **skewed distributions**.
- 

### Example:

- Odd Case:  
Data = [10, 20, 30, 40, 50]  
Median = 30
  - Even Case:  
Data = [10, 20, 30, 40]  
Median =  $(20 + 30) / 2 = 25$
- 

### Python Code (Median):

```
import numpy as np
import pandas as pd
```



```
data = [10, 20, 30, 40, 50]
median_np = np.median(data)
median_pd = pd.Series(data).median()

print("Median using NumPy:", median_np)
print("Median using Pandas:", median_pd)
```

---

## 3. Mode

### ◆ Definition:

The **mode** is the value that appears **most frequently** in a dataset.

- A dataset can have:
  - **One mode** (Unimodal)
  - **Two modes** (Bimodal)
  - **More than two modes** (Multimodal)
  - **No mode** (if all values occur equally)

---

### Key Points:

- Best used with **categorical** or **discrete data**.
- Can help identify **common patterns** in data.
- Can exist in **non-numeric data** (e.g., colors, names).

---

### Example:

- Data = [1, 2, 2, 3, 4, 4, 4, 5]  
Mode = 4 (it appears most often)
- 

### Python Code (Mode):

```
import pandas as pd
data = [1, 2, 2, 3, 4, 4, 4, 5]

# Using Pandas
mode_pd = pd.Series(data).mode()
print(mode_pd)
```

### Example Using Mean, Median, Mode

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

# Set Seaborn style for better aesthetics
sns.set(style="whitegrid")

# Create a sample dataset using a dictionary
base_data = {
    "product": ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J",
               "K", "L", "M", "N", "O"],
    "price": [100, 150, 140, 130, 170, 140, 160, 155, 120, 135,
             145, 165, 125, 110, 175],
    "gender": ["Male", "Female", "Female", "Male", "Male", "Female", "Male",
              "Female", "Male", "Female", "Male", "Female", "Male", "Female", "Male"]
}
```

```

"category": ["Electronics", "Clothing", "Grocery", "Electronics", "Clothing",
             "Grocery", "Electronics", "Clothing", "Grocery", "Electronics",
             "Clothing", "Grocery", "Electronics", "Clothing", "Grocery"],

"rating": [4.2, 3.9, 4.2, 3.9, 4.2, 4.2, 4.3, 4.2, 3.6, 4.8,
           3.9, 4.4, 4.0, 4.2, 4.6],

"in_stock": [True, True, False, True, False, True, True, False,
             True, True, False, True, True, False, True]
}

# Convert dictionary into DataFrame
df = pd.DataFrame(base_data)

# Calculate Mean and Median using NumPy
mean_numpy = np.mean(df["price"])
median_numpy = np.median(df["price"])
mode_price = df["price"].mode()[0] # Mode for price
# Print the statistics
print("Mean Price (NumPy):", mean_numpy)
print("Median Price (NumPy):", median_numpy)
print("Mode Price:", mode_price)

# Plot the distribution of prices
plt.figure(figsize=(10, 6))
sns.histplot(df["price"], bins=6, kde=True, color="skyblue")

# Add vertical lines for mean, median, and mode
plt.axvline(mean_numpy, color='red', linestyle='--', label=f"Mean: {mean_numpy:.2f}")
plt.axvline(median_numpy, color='green', linestyle='--', label=f"Median:
{median_numpy:.2f}")
plt.axvline(mode_price, color='black', linestyle='--', label=f"Mode: {mode_price:.2f}")

# Add labels and title
plt.title("Distribution of Product Prices")
plt.xlabel("Price")
plt.ylabel("Frequency")
plt.legend()

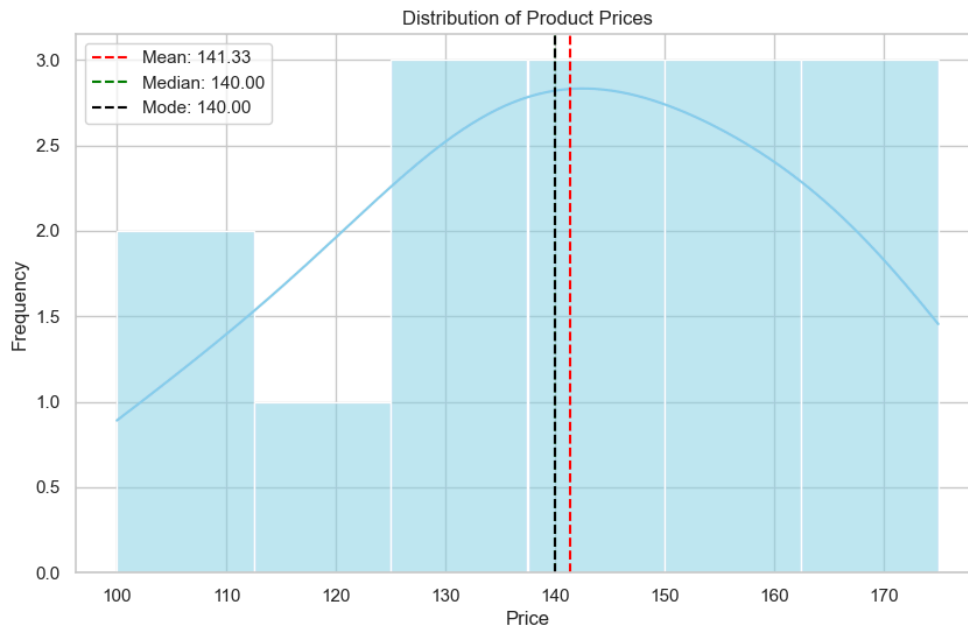
```

```
# Show plot
plt.show()
```

**Mean Price (NumPy): 141.33333333333334**

**Median Price (NumPy): 140.0**

**Mode Price: 140**



## Measures of Variability: Range

### ♦ What is Variability?

**Variability** (or dispersion) tells us how spread out the data is. It helps us understand the *consistency* and *reliability* of the data.

### ♦ 1. Range

#### ✓ Definition:

The **Range** is the difference between the **maximum** and **minimum** value in a dataset.

## Range=Maximum Value-Minimum Value

---

### Example:

If the prices of 5 products are:

[100, 150, 130, 120, 140]

Then:

- Max = 150
  - Min = 100
  - **Range** = 150 - 100 = 50
- 

### Python Code Example:

Using NumPy and Pandas to calculate the range:

```
import numpy as np
import pandas as pd

# Example dataset
prices = [100, 150, 130, 120, 140]

# Convert to Pandas Series
price_series = pd.Series(prices)

# Calculate range
range_val = price_series.max() - price_series.min()

print("Range of Prices:", range_val)
```

**Output:**

**Range of Prices: 50**

---

### Usefulness of Range:

- Quick and simple to calculate
- Useful for understanding the total spread of data

---

### Limitations:

- **Affected by outliers** (very high or very low values)
- Doesn't consider how data is distributed between the extremes

## Mean Absolute Deviation (MAD) with Visualization

### ♦ Objective:

To compare two datasets (Sec A and Sec B student scores) using:

- Mean of Section B
- Mean Absolute Deviation (MAD)
- Visualization with Matplotlib

---

### Key Concepts:

#### Mean:

The average value of a dataset.

Formula:

$$\text{Mean} = \frac{\sum x_i}{n}$$

### Mean Absolute Deviation (MAD):

It shows the average distance of each data point from the **mean**, helping us understand **data spread**.

Formula:

$$\text{MAD} = \frac{1}{n} \sum |x_i - \bar{x}|$$

### Python Code:

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

# Section scores
sec_a = np.array([75, 65, 73, 68, 72, 76])
sec_b = np.array([90, 47, 43, 96, 93, 51])
no = np.array([1, 2, 3, 4, 5, 6])

# Calculate mean of Section B
mean = np.mean(sec_b)

# Calculate Mean Absolute Deviation for both sections
mad_a = np.sum(np.abs(sec_a - mean)) / len(sec_a)
mad_b = np.sum(np.abs(sec_b - mean)) / len(sec_b)

# Print the MAD values
print("MAD of Section A:", mad_a)
print("MAD of Section B:", mad_b)

# Plot the data
plt.figure(figsize=(10, 3))
plt.scatter(sec_a, no, label="Sec A")
```

```
plt.scatter(sec_b, no, color="red", label="Sec B")

# Plot the mean line of Section B
plt.plot([mean]*6, no, color="blue", linestyle="--", label=f"Mean of Sec B: {mean:.2f}")

# Add plot details
plt.legend()
plt.title("Comparison of Section A and B with Mean Line")
plt.xlabel("Marks")
plt.ylabel("Student Number")
plt.grid(True)
plt.show()
```



### Output Description:

- **Red Dots:** Marks of students from Section B
- **Blue Dots:** Marks of students from Section A
- **Blue Dashed Line:** Mean of Section B
- **MAD Values:** Printed in terminal/console



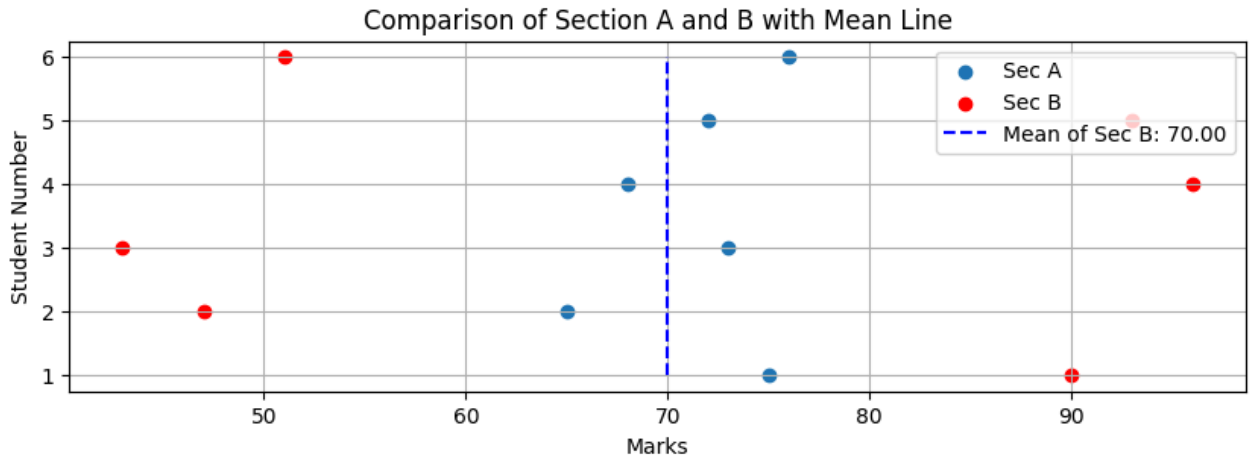
### Conclusion:

- **Higher MAD** means **greater variation** from the mean.
- **MAD helps in comparing variability** between different datasets.
- Visualizing with a mean line gives a clear idea of how scores are distributed around the mean.

**MAD of Section A: 3.8333333333333335**

**MAD of Section B: 23.0**





## Measures of Variability – Variance & Standard Deviation

### Objective:

To understand how much the data is **spread out** from the **mean** using:

- Variance
- Standard Deviation (STD)

### ✓ 1. What is Variance?

- **Definition:**  
Variance measures the **average squared deviation** from the mean.

- **Formula (Population):**

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$

- **Formula (Sample):**

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

- **Interpretation:**
    - A **higher variance** indicates more spread-out data.
    - A **lower variance** means data points are close to the mean.
- 

## ✓ 2. What is Standard Deviation?

- **Definition:**

It is the **square root of the variance**. It represents the **average distance** of data from the mean.
  - **Formula:**

$$\sigma = \sqrt{\sigma^2} \quad \text{or} \quad s = \sqrt{s^2}$$
  - **Interpretation:**
    - **Same units** as the data.
    - Easier to understand compared to variance.
- 

### Python Code Example:

```
import numpy as np

# Sample data
marks = np.array([70, 75, 80, 85, 90])

# Calculate Mean
mean = np.mean(marks)
```

```
# Calculate Variance and Standard Deviation

variance = np.var(marks)      # By default, calculates population variance

std_dev = np.std(marks)      # Population standard deviation


# For sample variance and std

sample_variance = np.var(marks, ddof=1)

sample_std_dev = np.std(marks, ddof=1)


# Print the results

print("Mean:", mean)

print("Population Variance:", variance)

print("Population Std Dev:", std_dev)

print("Sample Variance:", sample_variance)

print("Sample Std Dev:", sample_std_dev)
```

---

#### Output Example:

**Mean: 80.0**

**Population Variance: 50.0**

**Population Std Dev: 7.071**

**Sample Variance: 62.5**

**Sample Std Dev: 7.906**

---

## Conclusion:

Measure	Tells You About	Units
Variance	Spread from the mean (squared)	Squared
Standard Deviation	Average distance from the mean	Same as data

- Use **STD** when you want to **interpret spread** in the same units as data.
- Use **VAR** to understand **mathematical spread** (especially in formulas/statistics).

## Percentiles and Quartiles

---

### 1. What are Percentiles?

#### ♦ Definition:

- A **percentile** indicates the value **below which a given percentage** of data points fall.
- It divides data into **100 equal parts**.

#### ♦ Key Examples:

- **50th Percentile (P50):** Median of the data.

- **25th Percentile (P25):** First Quartile (Q1)
- **75th Percentile (P75):** Third Quartile (Q3)

♦ **Usage:**

- Used in exams, performance ranking, distributions.
- For example: *If you score in the 90th percentile, you did better than 90% of students.*

---

✓ **2. What are Quartiles?**

♦ **Definition:**

- Quartiles are **special percentiles** that divide the data into **four equal parts**.

♦ **Quartile Types:**

Quartile	Percentile	Meaning
Q1	25th	25% of data is below Q1
Q2	50th	50% of data is below Q2 (Median)
Q3	75th	75% of data is below Q3

---

📏 **Interquartile Range (IQR):**

- $IQR = Q3 - Q1$
- It measures the **spread of the middle 50%** of the data.
- Helps identify **outliers**.

---

🐍 **Python Code Example (Using NumPy and Pandas):**

```

import numpy as np
import pandas as pd

# Sample data
data = np.array([10, 20, 30, 40, 50, 60, 70, 80, 90, 100])

# Using NumPy
p25 = np.percentile(data, 25)
p50 = np.percentile(data, 50)
p75 = np.percentile(data, 75)

# Interquartile Range
iqr = p75 - p25

print("Q1 (25th percentile):", p25)
print("Q2 (50th percentile - Median):", p50)
print("Q3 (75th percentile):", p75)
print("IQR:", iqr)

```



### Output Example:

**Q1 (25th percentile): 32.5**  
**Q2 (50th percentile - Median): 55.0**  
**Q3 (75th percentile): 77.5**  
**IQR: 45.0**



### Conclusion:

Term	Divides Data Into	Use
Percentiles	100 parts	Ranking, distribution, scores
Quartiles	4 parts	Spread and outlier detection

- ✓ Percentiles = Fine-grained view
- ✓ Quartiles = Broader summary

## Boxplot to Understand Percentiles and Quartiles

♦ A Boxplot is a graphical representation of the distribution of data based on:

- Minimum
- Q1 (25th percentile)
- Median (Q2 / 50th percentile)
- Q3 (75th percentile)
- Maximum
- Outliers (if any)

---

### Python Code for Boxplot

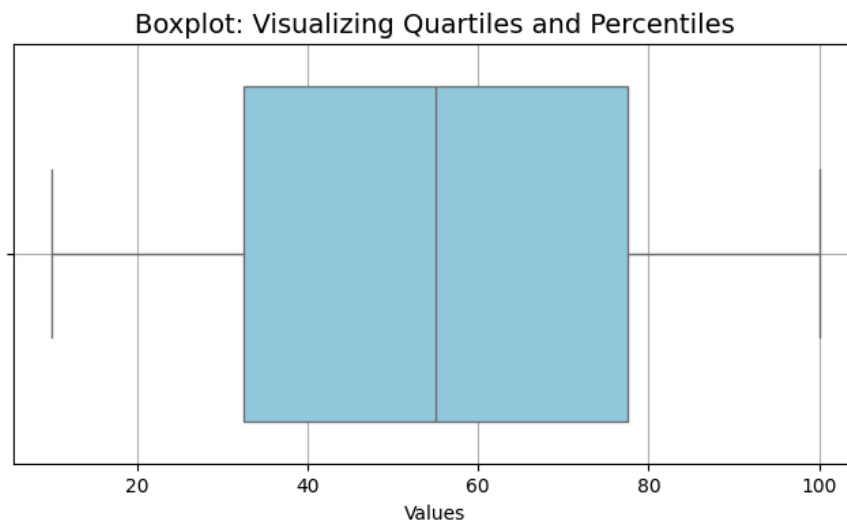
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Sample dataset
data = np.array([10, 20, 30, 40, 50, 60, 70, 80, 90, 100])

# Create a boxplot
plt.figure(figsize=(8, 4))
sns.boxplot(x=data, color="skyblue")

# Add labels and title
plt.title("Boxplot: Visualizing Quartiles and Percentiles", fontsize=14)
plt.xlabel("Values")
plt.grid(True)

plt.show()
```



## What is Skewness in Statistics?

**Skewness** is a measure of **asymmetry** in the distribution of data. It tells us **how much** and in **which direction** the data deviates from a **normal (symmetrical) distribution**.

---

### Types of Skewness:

1. ♦ **Positive Skew (Right-Skewed):**
  - Tail is stretched **to the right**.
  - **Mean > Median > Mode**
  - Example: Income distribution (a few people earn much more than the average).
2. ♦ **Negative Skew (Left-Skewed):**
  - Tail is stretched **to the left**.
  - **Mean < Median < Mode**
  - Example: Age at retirement (most people retire at a similar age, but some retire very early).



### 3. No Skew (Symmetrical):

- Data is **evenly distributed**.
- **Mean = Median = Mode**
- Example: Perfectly normal bell curve.

## Python Example with Visualization

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

# Function to calculate skewness manually
def manual_skewness(data):
    n = len(data)
    mean = np.mean(data)
    std = np.std(data)
    skewness = np.sum((data - mean)**3) / n / (std**3)
    return skewness

# Create sample data
data_right = np.random.exponential(scale=2, size=1000) # Positively skewed
data_left = -1 * np.random.exponential(scale=2, size=1000) # Negatively skewed

# Calculate skewness
skew_right = manual_skewness(data_right)
skew_left = manual_skewness(data_left)

print(f"Skewness (Right Skewed): {skew_right:.2f}")
print(f"Skewness (Left Skewed): {skew_left:.2f}")

# Plot both
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
sns.histplot(data_right, kde=True, color="orange")
```

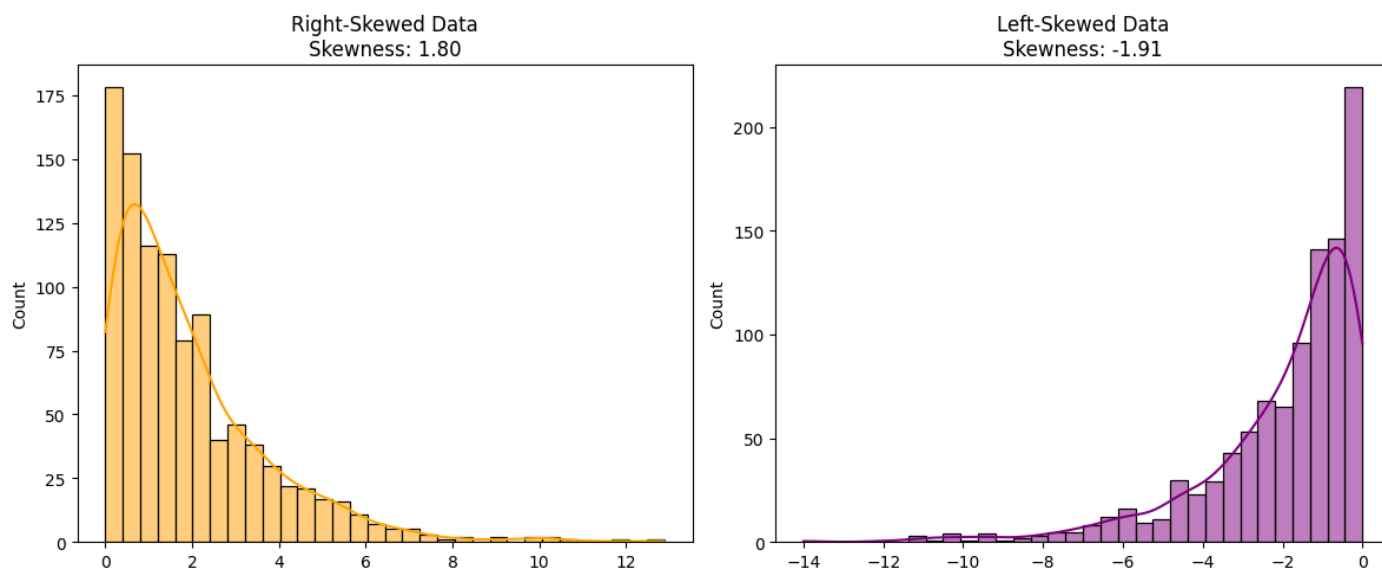
```
plt.title(f"Right-Skewed Data\nSkewness: {skew_right:.2f}")

plt.subplot(1, 2, 2)
sns.histplot(data_left, kde=True, color="purple")
plt.title(f"Left-Skewed Data\nSkewness: {skew_left:.2f}")

plt.tight_layout()
plt.show()
```

**Skewness (Right Skewed): 1.80**

**Skewness (Left Skewed): -1.91**



## 1. What is a Random Variable?

A **Random Variable** is a variable whose value is **determined by the outcome** of a random experiment.

- It can take different values each time the experiment is repeated.
- Two main types:

- **Discrete Random Variable:** Takes **countable** values (e.g., number of heads).
- **Continuous Random Variable:** Takes **infinite** values in a range (e.g., height, weight).

## 2. What is Probability?

**Probability** is the **likelihood** of an event occurring.

- It is always between **0 and 1**.
- Formula:  

$$P(E) = \frac{\text{Favorable outcomes}}{\text{Total outcomes}}$$

$$P(E) = \frac{\text{Favorable outcomes}}{\text{Total outcomes}}$$

## 3. What is Probability Distribution Function (PDF)?

A **Probability Distribution** describes how **probabilities are distributed** over values of the random variable.

➤ **Two types of probability distributions:**


### A. Discrete Probability Distribution

Used for **discrete random variables**.

#### **PMF (Probability Mass Function):**

- It gives the **probability** of each discrete value.
- 

$P(X = x)$   
]

 **Example:** Tossing a die —  $P(X=3) = 1/6$

## B. Continuous Probability Distribution

Used for **continuous random variables**.

### ✓ PDF (Probability Density Function):

- Describes the **relative likelihood** of a random variable to take a value.
- The **area under the curve** = 1.
- We **don't get exact value probabilities** but intervals.

 **Example:** Height distribution in a population

### ✓ CDF (Cumulative Distribution Function):

- Shows the **probability that a random variable is less than or equal to a value**.

$$F(x) = P(X \leq x)$$

- **Monotonically increasing** function.

## 4. Normal Distribution (Bell Curve)

- A **continuous**, symmetric distribution.
- Most values cluster around the **mean**, with fewer farther away.
- Defined by:
  - **Mean ( $\mu$ )**
  - **Standard Deviation ( $\sigma$ )**

 PDF Formula:

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

### Properties:

- Bell-shaped curve
- Mean = Median = Mode
- 68%-95%-99.7% Rule (Empirical Rule):
  - 68% values lie within  $1\sigma$
  - 95% within  $2\sigma$
  - 99.7% within  $3\sigma$

## 5. Standard Normal Distribution

- A special type of **Normal Distribution**
- Mean ( $\mu$ ) = 0
- Standard Deviation ( $\sigma$ ) = 1
- Useful for **Z-scores** and standardization.

## Summary Table

Concept	Type	Function	Description
Random Variable	Discrete / Continuous	–	Output of a random experiment
PMF	Discrete	$P(X=x)P(X=x)$	Gives probability of each value
PDF	Continuous	–	Probability over intervals

CDF	Both	$P(X \leq x)P(X \leq x)$	Cumulative probability
Normal Distribution	Continuous	Bell Curve	Mean = Median = Mode
Standard Normal	Continuous	Z-curve	$\mu = 0, \sigma = 1$

## Covariance and Correlation – Notes

### ♦ 1. Covariance (सह-संबद्धता)

#### Definition:

Covariance measures how two variables change together.

- If both variables increase or decrease together → **Positive covariance**
- If one increases while the other decreases → **Negative covariance**

#### Formula:

$$\text{Cov}(X, Y) = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{n}$$

**Limitation:** Covariance is not standardized; it depends on the units of variables.

### ♦ 2. Correlation

#### Definition:

Correlation measures both the **strength** and **direction** of a linear relationship between two variables.

#### Formula (Pearson):

$$r = \frac{\text{Cov}(X, Y)}{\sigma_X \cdot \sigma_Y}$$

- Value of  $r$  is always between **-1 and 1**
  - $r = 1$ : perfect positive linear relation
  - $r = -1$ : perfect negative linear relation
  - $r = 0$ : no linear relationship

### Types:

- Pearson (linear)
- Spearman (rank-based)
- Kendall Tau (rank correlation)

## Difference Between Covariance and Correlation

Feature	Covariance	Correlation
Meaning	Direction of relation	Direction + strength of relation
Value Range	$-\infty$ to $\infty$	-1 to 1
Units	Not standardized	Unit-less (standardized)
Use	Intermediate step to correlation	Final, interpretable measure

### Example

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

# Sample data
x = np.array([2, 4, 6, 8, 10])
y = np.array([1, 3, 4, 6, 8])

# Create DataFrame
```

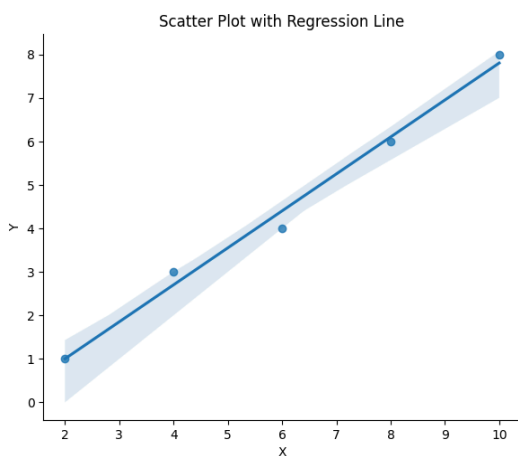
```
df = pd.DataFrame({"X": x, "Y": y})

# Covariance
cov_matrix = np.cov(x, y) # by default, gives sample covariance
print("Covariance Matrix:\n", cov_matrix)
print("Covariance (X,Y):", cov_matrix[0, 1])

# Correlation
corr_matrix = np.corrcoef(x, y)
print("Correlation Matrix:\n", corr_matrix)
print("Correlation (X,Y):", corr_matrix[0, 1])
```

```
Covariance Matrix:
[[10.   8.5]
 [ 8.5  7.3]]
Covariance (X,Y): 8.5
Correlation Matrix:
[[1.         0.99484975]
 [0.99484975 1.         ]]
Correlation (X,Y): 0.9948497511671097
```

```
# Scatter plot with regression line
sns.lmplot(x="X", y="Y", data=df, height=5, aspect=1.2)
plt.title("Scatter Plot with Regression Line")
plt.show()
```





## Central Limit Theorem (CLT)

### ◆ Definition:

The **Central Limit Theorem** states that:

When we take **many random samples** (with replacement) of a given size from **any population** (not necessarily normally distributed), the **sampling distribution of the sample mean** tends to be **approximately normal** as the sample size increases.

---

### ✓ Key Points:

- Works even if the original population is **not normal**.
- The **mean** of the sample means  $\approx$  population mean ( $\mu$ ).
- The **standard deviation** of the sample means is called the **Standard Error (SE)**:

$$SE = \frac{\sigma}{\sqrt{n}}$$

- Becomes more accurate as **sample size (n)** increases (typically  $\geq 30$  is enough).
- 



### Why is CLT Important?

- Helps us apply **normal distribution-based statistics** to real-world data.
  - Forms the **basis of many statistical techniques**, including hypothesis testing and confidence intervals.
- 



## Python Example – Central Limit Theorem

Let's simulate CLT using a **non-normal population**:

```
import numpy as np
```

```

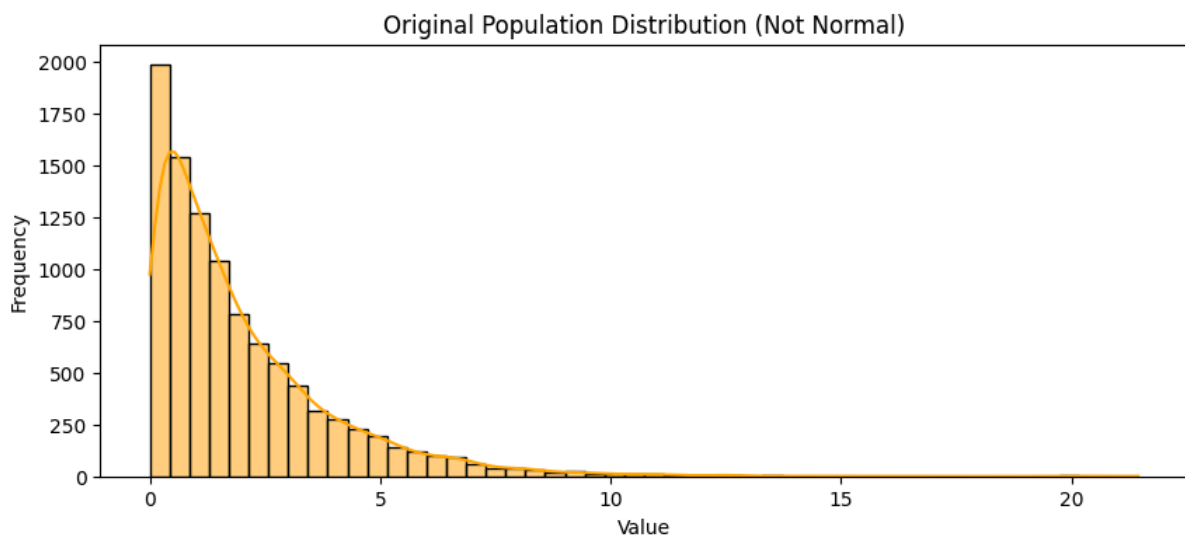
import matplotlib.pyplot as plt
import seaborn as sns

# Set random seed
np.random.seed(0)

# Create a non-normal population (exponential distribution)
population = np.random.exponential(scale=2, size=10000)

# Plot original population
plt.figure(figsize=(10, 4))
sns.histplot(population, bins=50, kde=True, color="orange")
plt.title("Original Population Distribution (Not Normal)")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()

```



### Simulating Sampling Distribution of the Mean

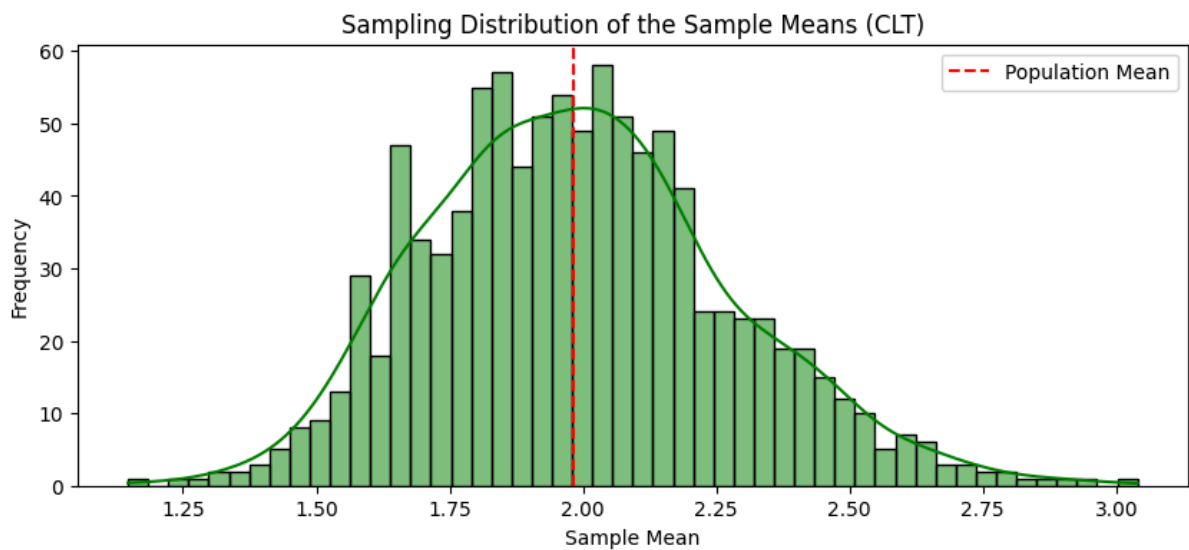
```

sample_means = []

# Take 1000 samples, each of size 50
for _ in range(1000):
    sample = np.random.choice(population, size=50)
    sample_means.append(np.mean(sample))

```

```
# Plot the distribution of sample means
plt.figure(figsize=(10, 4))
sns.histplot(sample_means, bins=50, kde=True, color="green")
plt.title("Sampling Distribution of the Sample Means (CLT)")
plt.xlabel("Sample Mean")
plt.ylabel("Frequency")
plt.axvline(np.mean(population), color='red', linestyle='--', label='Population Mean')
plt.legend()
plt.show()
```



## Summary in Simple Terms

- Think of CLT as saying: *“Even if your data is weird, the average of repeated samples behaves nicely (i.e., normally).”*
- This is **why normal distribution appears so often** in statistics.

## Hypothesis Testing

### What is Hypothesis Testing?

Hypothesis testing is a **statistical method** used to make decisions using data. It helps us determine whether a claim about a population parameter (like the mean) is **true or false**.

---

### Types of Hypotheses

Term	Meaning
Null Hypothesis ( $H_0$ )	A default assumption or claim — <i>there is no effect or difference</i> .
Alternative Hypothesis ( $H_1$ or $H_a$ )	What you want to prove — <i>there is an effect or difference</i> .

### Examples:

- $H_0$ : The average exam score is **75**.
  - $H_1$ : The average exam score is **not 75**.
- 

### Steps of Hypothesis Testing

1. **State the Hypotheses**
  - Null Hypothesis ( $H_0$ )
  - Alternative Hypothesis ( $H_1$ )
2. **Set the Significance Level ( $\alpha$ )**
  - Common choices: 0.05, 0.01, 0.10
  - This is the probability of rejecting  $H_0$  when it's actually true.
3. **Choose the Appropriate Test**

- **Z-test** (if population standard deviation is known, large sample size)
- **T-test** (if standard deviation is unknown, small sample size)

#### 4. Calculate the Test Statistic

- e.g., z-score:

$$z = \frac{\bar{x} - \mu}{\sigma / \sqrt{n}}$$

#### 5. Make a Decision

- Compare the z-score with the critical value
- Or use the **p-value**
- If  $p\text{-value} < \alpha \rightarrow \text{Reject } H_0$

#### 6. Conclusion

- Reject or fail to reject the null hypothesis
- Interpret result in context



## Example of Z-Test in Python

### Problem:

A factory claims that its bulbs last **on average 1200 hours**. A sample of 40 bulbs has a **mean life of 1170 hours** with a known **population standard deviation of 100 hours**. Test the claim at **5% significance level**.

---

### ✓ Python Code:

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

# Given data
mu = 1200      # population mean
sample_mean = 1170
```

```

sigma = 100    # population standard deviation
n = 40         # sample size
alpha = 0.05   # significance level

# Step 4: Calculate z-score
z = (sample_mean - mu) / (sigma / np.sqrt(n))

# Step 5: Calculate p-value (two-tailed test)
p_value = 2 * (1 - stats.norm.cdf(abs(z)))

# Step 6: Decision
print("Z-score:", z)
print("P-value:", p_value)

if p_value < alpha:
    print("Reject the Null Hypothesis (H0)")
else:
    print("Fail to Reject the Null Hypothesis (H0)")

```

```

Z-score: -1.8973665961010275
P-value: 0.05777957112359733
Fail to Reject the Null Hypothesis (H0)

```



### Interpretation:

If the p-value is less than 0.05, we reject  $H_0$  – meaning the factory's claim is likely incorrect.



### Summary Table:

Test	Use Case
<b>Z-test</b>	Population standard deviation is known, sample size large
<b>T-test</b>	Population standard deviation unknown, small sample size

<b>Chi-square test</b>	Categorical data
<b>ANOVA</b>	Comparing means of 3+ groups