Statistics and Probability Refresher, and Python Practice

1. Type of Data (Numerical, Categorial, ordinal)

1. Numerical Data (Quantitative Data) Numerical data consists of measurable quantities and can be either discrete or continuous.

Discrete (countable values): Number of students in a class (e.g., 25, 30, 40). Number of cars in a parking lot (e.g., 10, 15, 20).
Continuous (measurable values with decimals): Height of individuals (e.g., 5.6 feet, 170.2 cm). Temperature readings (e.g., 36.5°C, 98.6°F).

2. Categorical Data (Qualitative Data) Categorical data represents groups, labels, or categories without a natural order.

Examples: Colors of cars (e.g., Red, Blue, Black). Gender (e.g., Male, Female, Other). Blood type (e.g., A, B, AB, O).

3. Ordinal Data Ordinal data represents categories with a meaningful order but unequal intervals.

Examples: Customer satisfaction levels (e.g., Poor, Average, Good, Excellent). Education level (e.g., High School, Bachelor's, Master's, PhD). Spice level in food (e.g., Mild, Medium, Spicy, Extra Spicy).

Double-click (or enter) to edit

Double-click (or enter) to edit

2. Mean, Median, and Mode - Practical Examples

1. Mean (Average) The mean is the sum of all values divided by the total number of values.

Example: Test scores: [85, 90, 78, 92, 88]
Mean = 85 + 90 + 78 + 92 + 88 / 5
Mean = 86.6

```
import numpy as np
import statistics as s
from scipy import stats

scores = [85, 90, 78, 92, 88]

mean_score = np.mean(scores)
print("Meas of Test Scores:",mean_score)
```

```
Meas of Test Scores: 86.6
```

2. Median (Middle Value) The median is the middle number when values are arranged in order.

Example 1 (Odd set of numbers): [10, 20, 30, 40, 50] → Median = 30

Example 2 (Even set of numbers): [10, 20, 30, 40, 50, 60]

Median = 30 + 40 / 2
= 35

```
odd_list = [10, 20, 30, 40, 50]

median_result = np.median(odd_list)
print("Median of Odd set of numbers:",median_result)

even_list = [10, 20, 30, 40, 50, 60]
median_result = np.median(even_list)
print('\nMedian of Even set of numbers:',median_result)
```

```
Median of Odd set of numbers: 30.0

Median of Even set of numbers: 35.0
```

3. Mode (Most Frequent Value) The mode is the number that appears the most in a dataset.

Example: [1, 2, 2, 3, 3, 3, 4, 5] → Mode = 3 (appears most frequently)
[4, 4, 7, 7, 9] → Modes = 4 & 7 (bimodal) but 4 occurance first

```
mode_list = [1, 2, 2, 3, 3, 3, 4, 5]
print(f'Mode of List value:',stats.mode(mode_list))

mode_list = [4, 4, 7, 7, 9]
print(f'\nMode of List value:',stats.mode(mode_list))
```

```
Mode of List value: ModeResult(mode=3, count=3)

Mode of List value: 4
```

```
# Sample data
data = [85, 90, 78, 92, 88, 90, 78, 92, 85, 90]

# Calculate Mean
mean_value = np.mean(data)

# Calculate Median
median_value = np.median(data)

# Calculate Mode
mode_value = stats.mode(data, keepdims=True).mode[0]
# keepdims=True for compatibility with recent SciPy versions

# Display results
print(f"Mean: {mean_value}")
print(f"Median: {median_value}")
print(f"Mode: {mode_value}")
```

```
Mean: 86.8
Median: 89.0
Mode: 90
```

3. Calculate Variance and Standard Deviation

Variance (σ^2 or s^2): Measures the spread of data points around the mean.
Standard Deviation (σ or s): Square root of variance, representing dispersion in the same unit as the data.
ddof=0 → Population variance/standard deviation
ddof=1 → Sample variance/standard deviation (unbiased estimator)

```
import numpy as np

# Sample data
data = [85, 90, 78, 92, 88, 90, 78, 92, 85, 90]

# Calculate Variance
variance = np.var(data, ddof=0)  # Population variance
sample_variance = np.var(data, ddof=1)  # Sample variance

# Calculate Standard Deviation
std_dev = np.std(data, ddof=0)  # Population standard deviation
sample_std_dev = np.std(data, ddof=1)  # Sample standard deviation

# Display results
print(f"Population Variance: {variance}")
print(f"Sample Variance: {sample_variance}")
```

```
print(f"Population Standard Deviation: {std_dev}")
print(f"Sample Standard Deviation: {sample_std_dev}")
```

```
Population Variance: 24.759999999999998
Sample Variance: 27.51111111111111
Population Standard Deviation: 4.975942121849891
Sample Standard Deviation: 5.2451035367389185
```

4. Probability Density Function (PDF) and Probability Mass Function (PMF)

Probability Density Function (PDF) vs. Probability Mass Function (PMF) Probability Mass Function (PMF)

Used for discrete random variables (e.g., number of heads in coin flips, number of students in a class). It gives the probability of a specific discrete outcome. The sum of all PMF values equals 1.

```
# PMF Example: Probability of getting 3 heads in 5 fair coin flips
n = 5   # Number of trials
p = 0.5 # Probability of heads
k = 3   # Number of successes

prob_3_heads = binom.pmf(k, n, p)
print(f"Probability of getting exactly 3 heads: {prob_3_heads:.4f}")
```

```
Probability of getting exactly 3 heads: 0.3125
```

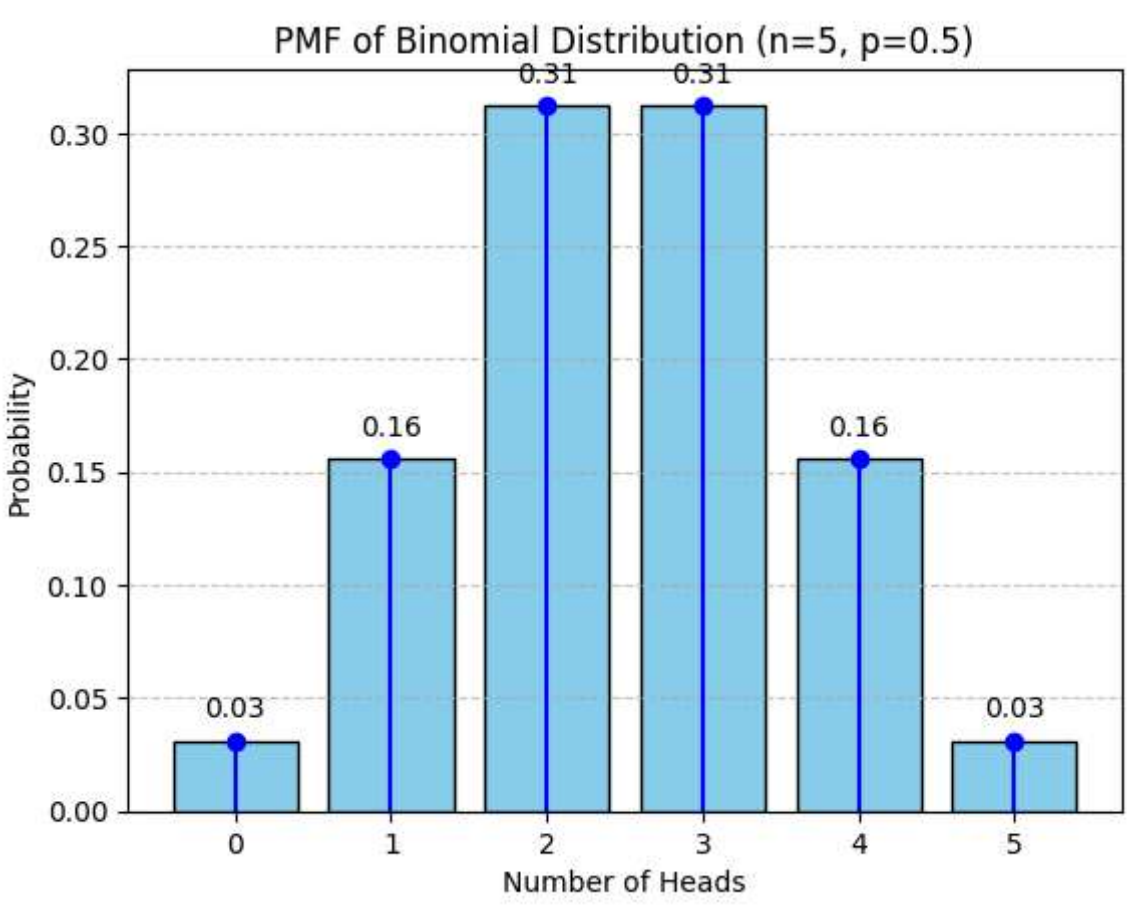
```
import matplotlib.pyplot as plt
from scipy.stats import binom

# Parameters
n = 5   # Number of trials (coin flips)
p = 0.5 # Probability of heads
x = np.arange(0, n + 1)  # Possible outcomes (0 to 5 heads)
pmf_values = binom.pmf(x, n, p)  # Compute PMF

# Plot PMF
plt.bar(x, pmf_values, color='skyblue', edgecolor='black')
plt.stem(x, pmf_values, linefmt='blue', markerfmt='bo', basefmt=" ")
plt.xlabel("Number of Heads")
plt.ylabel("Probability")
plt.title("PMF of Binomial Distribution (n=5, p=0.5)")
plt.xticks(x)
plt.grid(axis='y', linestyle='--', alpha=0.7)
```

```
# Show values on top of bars
for i, val in enumerate(pmf_values):
    plt.text(x[i], val + 0.01, f"{val:.2f}", ha='center', fontsize=10)

plt.show()
```



Probability Density Function (PDF)

Used for continuous random variables (e.g., height, weight, temperature). It represents the probability of a variable falling within a range, not a specific value (since for continuous data, $P(X = x) = 0$ P(X=x)=0).
The total area under the PDF curve equals 1.

Example (Height of people following a normal distribution):

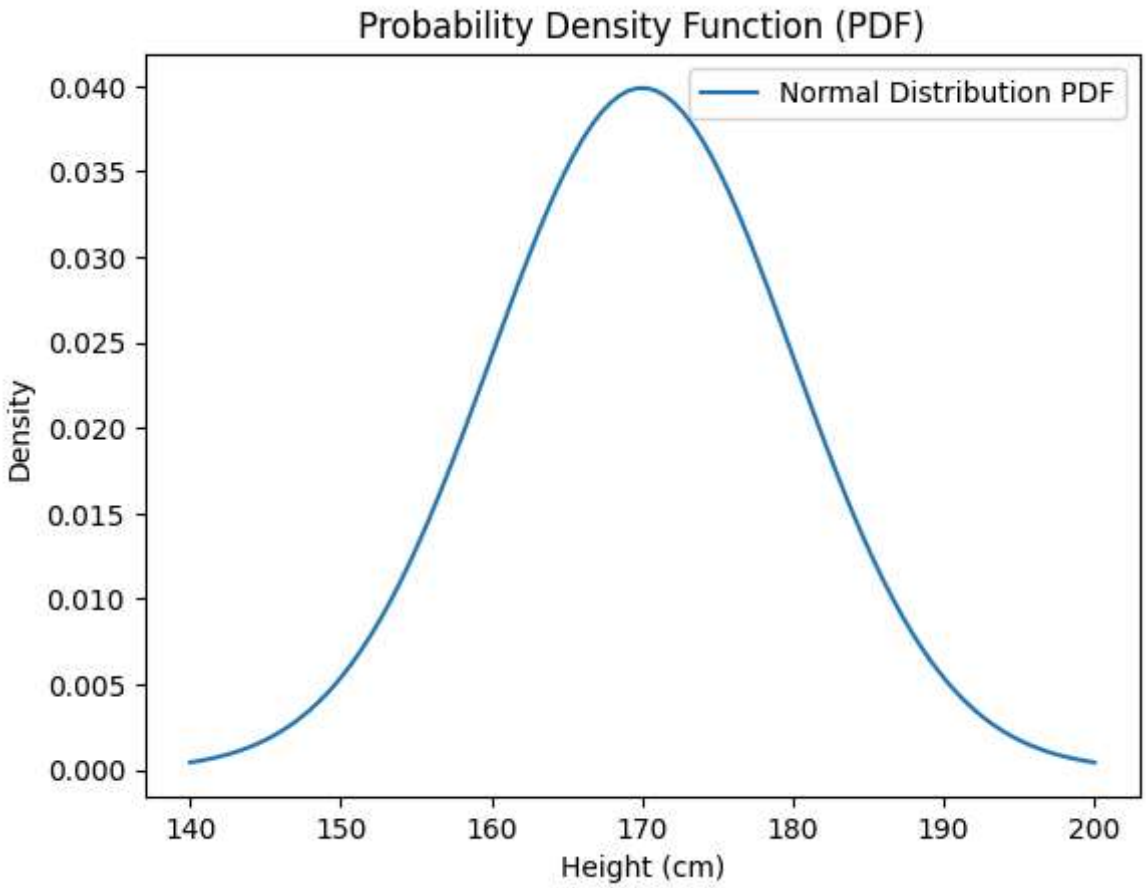
Mean height (μ μ) = 170 cm
Standard deviation (σ) = 10 cm
PDF will tell us the likelihood of a person's height falling in a certain range.

```
from scipy.stats import norm
# Normal Distribution Parameters
mu = 170  # Mean
```

```
sigma = 10 # Standard Deviation
x = np.linspace(140, 200, 100)

# Compute PDF
pdf_values = norm.pdf(x, mu, sigma)

# Plot PDF
plt.plot(x, pdf_values, label="Normal Distribution PDF")
plt.xlabel("Height (cm)")
plt.ylabel("Density")
plt.title("Probability Density Function (PDF)")
plt.legend()
plt.show()
```



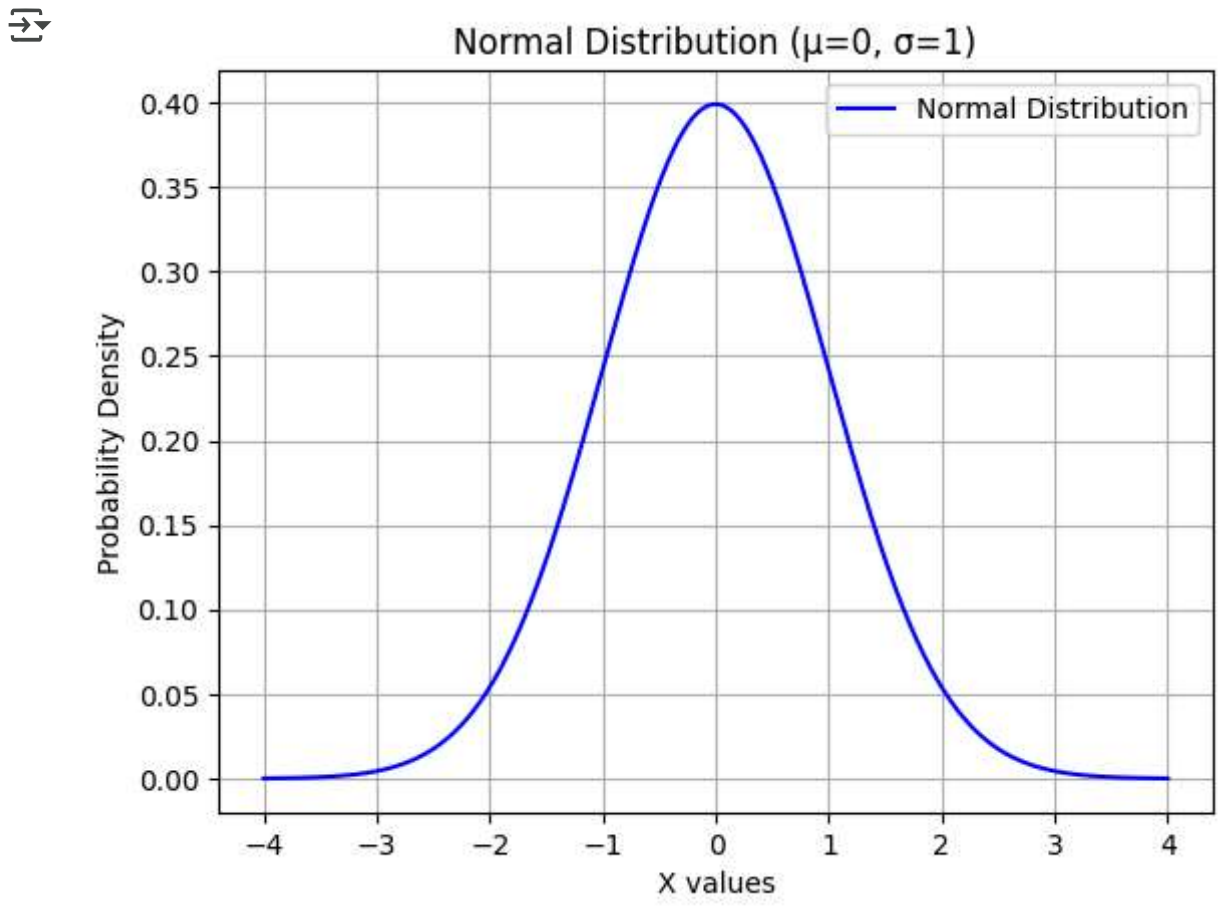
5. Common Data Distributions: Normal, Binomial, Poisson

1. Normal Distribution (Gaussian Distribution) Definition: A continuous probability distribution that is symmetric around the mean. The famous "bell curve" shape.

Used in natural phenomena like height, weight, IQ scores. Defined by mean (μ) and standard deviation (σ).

```
# Generated X value
X = np.linspace(-4,4,10000)
mu, sigma = 0, 1  # Mean = 0, Standard Deviation = 1

#Calculate PDF
norm_pdf = norm.pdf(X,mu,sigma)
# Plot
plt.plot(X, norm_pdf, label="Normal Distribution", color="blue")
plt.xlabel("X values")
plt.ylabel("Probability Density")
plt.title("Normal Distribution (μ=0, σ=1)")
plt.legend()
plt.grid()
plt.show()
```



2. Binomial Distribution Definition: A discrete probability distribution for a fixed number of trials (n) with two possible outcomes (success/failure).

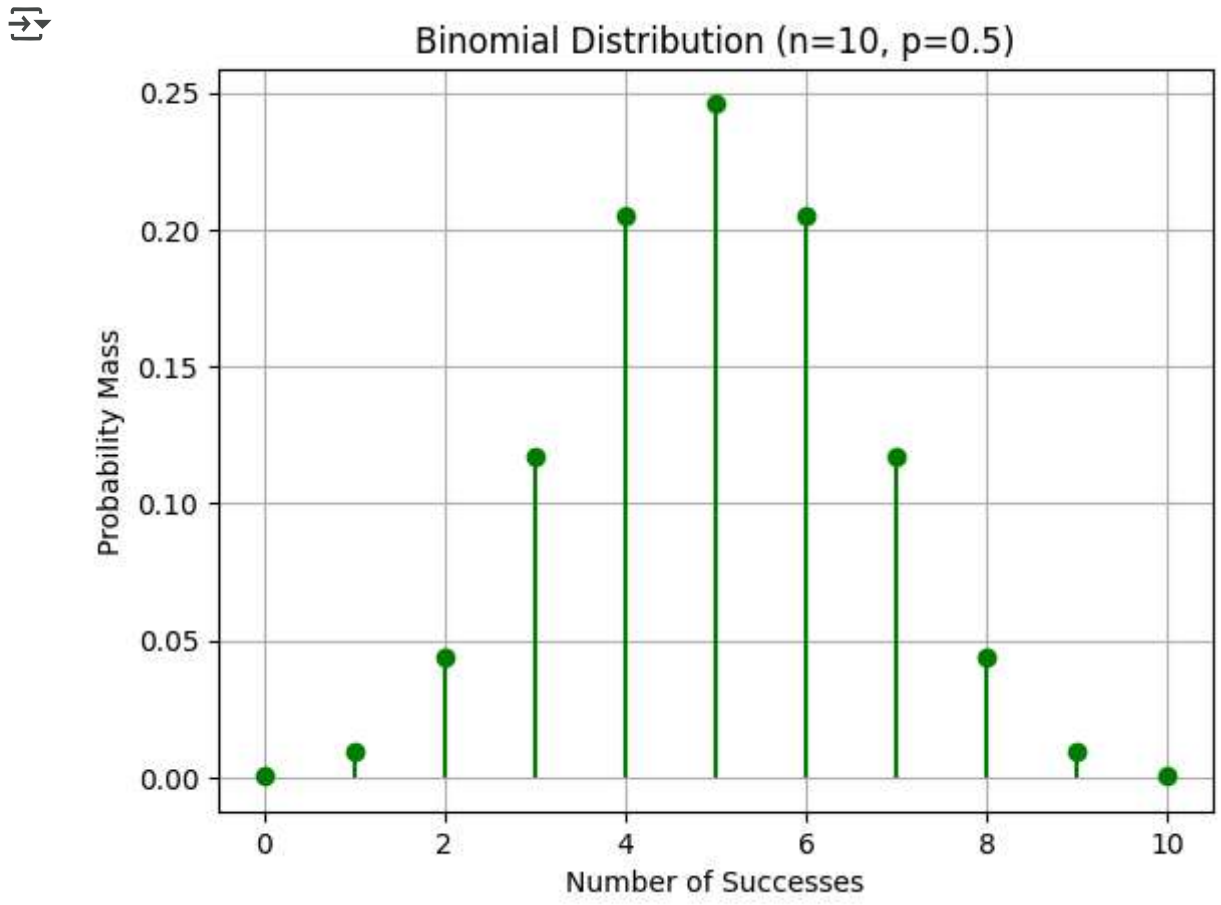
Used in coin flips, pass/fail tests, manufacturing defects. Defined by n (number of trials) & p (probability of success per trial).

```
from scipy.stats import binom

# Parameters
```

```
n, p = 10, 0.5  # 10 trials, 50% success probability
x = np.arange(0, n+1)
pmf_values = binom.pmf(x, n, p)

# Plot
plt.stem(x, pmf_values, linefmt="green", markerfmt="go", basefmt=" ")
plt.xlabel("Number of Successes")
plt.ylabel("Probability Mass")
plt.title("Binomial Distribution (n=10, p=0.5)")
plt.grid()
plt.show()
```



3. Poisson Distribution Definition: A discrete probability distribution that models the number of events occurring in a fixed interval of time/space, given a constant mean rate.

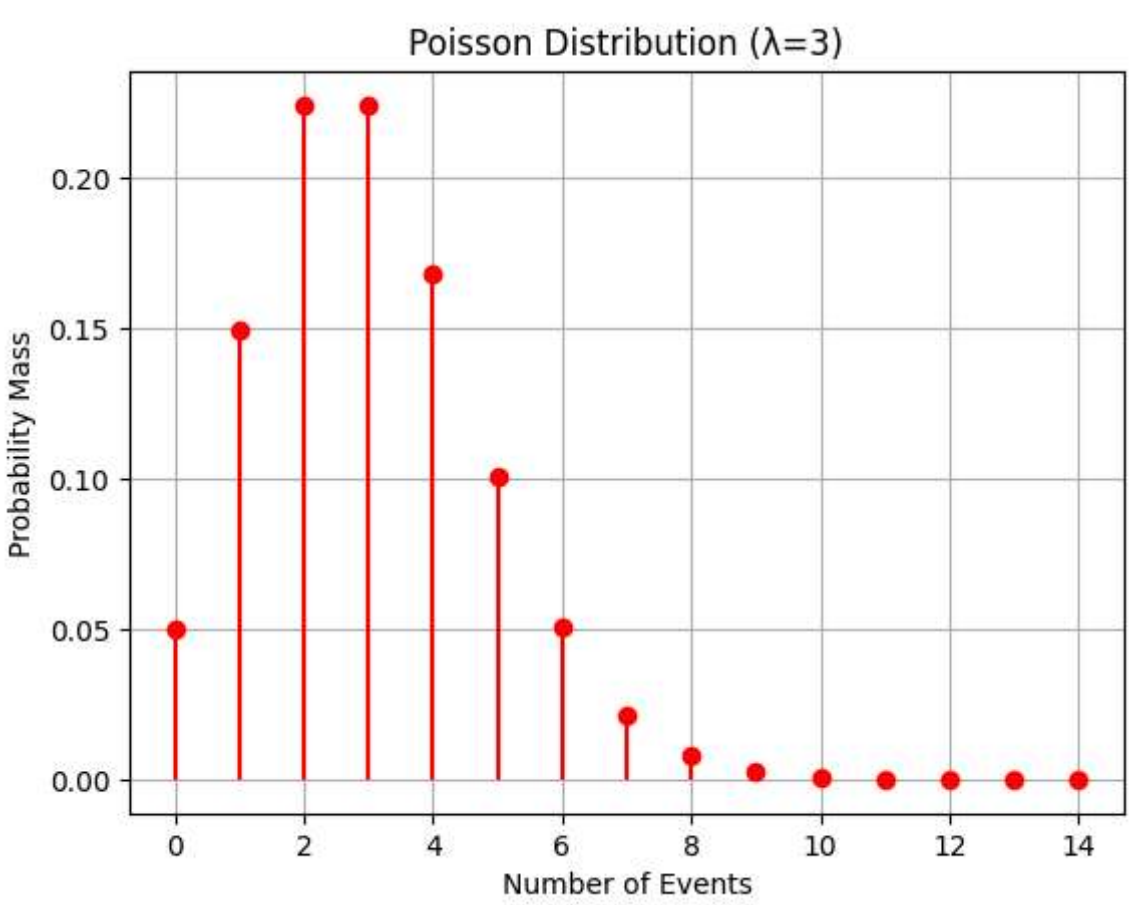
Used in call center arrivals, server requests, rare disease occurrences. Defined by λ (average event rate per interval).

```
from scipy.stats import poisson

# Parameters
lambda_ = 3  # Average occurrences per interval
x = np.arange(0, 15)
```

```
pmf_values = poisson.pmf(x, lambda_)

# Plot
plt.stem(x, pmf_values, linefmt="red", markerfmt="ro", basefmt=" ")
plt.xlabel("Number of Events")
plt.ylabel("Probability Mass")
plt.title("Poisson Distribution (λ=3)")
plt.grid()
plt.show()
```



6. Covariance and correlation

Covariance: Covariance measures the degree to which two random variables change together. It tells us whether an increase in one variable leads to an increase or decrease in the other variable. However, the magnitude of covariance depends on the scale of the variables, making it difficult to compare covariance values across different data sets.

Formula: For two variables X and Y, the covariance is calculated as:

Formula:

For two variables X and Y, the covariance is calculated as:

$$\text{Cov}(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \mu_X)(Y_i - \mu_Y)$$

Where:

- X_i and Y_i are the individual data points for variables X and Y,
- μ_X and μ_Y are the means of X and Y,
- n is the number of data points.

A positive covariance means that as one variable increases, the other tends to increase as well (and vice versa).

A negative covariance indicates that as one variable increases, the other tends to decrease.

A covariance of zero suggests that there is no linear relationship between the variables.

2. Correlation: Correlation is a normalized form of covariance that adjusts for the scales of the variables. This makes it easier to interpret, as the correlation coefficient always falls within the range of − 1 −1 to 1
3. A correlation of 1 1 indicates a perfect positive relationship, − 1 −1 indicates a perfect negative relationship, and 0 0 indicates no linear relationship.

Formula:

The correlation coefficient ρ (also known as Pearson's correlation coefficient) between two variables X and Y is calculated as:

Quick Actions

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

Where:

- $\text{Cov}(X,Y)$ is the covariance between X and Y,
- σ_X and σ_Y are the standard deviations of X and Y.

Key Differences:

Scale Dependence: Covariance depends on the units of the variables, while correlation is unitless and normalized, making it easier to compare.

Interpretation: Covariance gives the direction of the relationship (positive or negative), but correlation gives both the strength and direction in a more interpretable way.

```
# Example datasets
X = np.array([10, 20, 30, 40, 50])
Y = np.array([15, 25, 35, 45, 55])

# Calculate Covariance
covariance = np.cov(X, Y) #[0][1]

# Calculate Correlation
correlation = np.corrcoef(X, Y) #[0][1]

print(f"Covariance: {covariance}")
print(f"Correlation: {correlation}")
```

```
Covariance: [[250. 250.]
 [250. 250.]]
Correlation: [[1. 1.]
 [1. 1.]]
```

7. Conditional Probability

Conditional Probability is the probability of an event occurring given that another event has already occurred. It is used to refine the probability of an event when additional information is available

Formula:

The conditional probability of an event A given event B is denoted as $P(A|B)$, and it is defined as:

Quick Actions

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Where:

- $P(A|B)$ is the probability of event A occurring given that event B has occurred.
- $P(A \cap B)$ is the probability that both events A and B occur (i.e., the intersection of A and B).
- $P(B)$ is the probability of event B occurring.

Intuition:

- If $P(A \cap B)$ is high (i.e., A and B tend to occur together), and $P(B)$ is also high (i.e., B is likely to occur), then $P(A|B)$ will be high.
- If $P(B)$ is small or A and B rarely happen together, $P(A|B)$ will be low.

Suppose we are rolling a fair die and want to find the probability that the roll is a 4 (event A) given that the roll is an even number (event B).

- $P(A) = \frac{1}{6}$ (the probability of rolling a 4).
- $P(B) = \frac{3}{6} = \frac{1}{2}$ (the probability of rolling an even number, i.e., 2, 4, or 6).
- $P(A \cap B) = \frac{1}{6}$ (the probability of rolling a 4, which is both even and a 4).

Now, the conditional probability $P(A|B)$ is:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{\frac{1}{6}}{\frac{1}{2}} = \frac{1}{3}$$

Thus, the probability of rolling a 4 given that the roll is even is $\frac{1}{3}$.

```
# Define the probabilities
P_A_and_B = 1/6  # P(A ∩ B)
P_B = 1/2        # P(B)

# Calculate conditional probability P(A|B)
P_A_given_B = P_A_and_B / P_B

print(f"Conditional Probability P(A|B): {P_A_given_B}")
```

```
Conditional Probability P(A|B): 0.3333333333333333
```

2. Drawing a Card from a Deck:

- **Problem:** Find the probability of drawing a red card (event A) given that the card is a face card (event B).
- **Solution:**
  - $P(A \cap B) = \frac{6}{52}$ (2 red face cards: Jack, Queen, King for both Hearts and Diamonds)
  - $P(B) = \frac{12}{52}$ (12 face cards total: Jack, Queen, King of each suit)
  - $P(A|B) = \frac{\frac{6}{52}}{\frac{12}{52}} = \frac{1}{2}$

3. Coin Toss:

- **Problem:** Find the probability of getting heads (event A) given that the outcome is heads or tails (event B).
- **Solution:**
  - $P(A \cap B) = \frac{1}{2}$ (heads)
  - $P(B) = 1$ (since it's certain that the outcome is heads or tails)
  - $P(A|B) = \frac{1}{2}$

8. Baye's Theorm

Bayes' Theorem is a fundamental theorem in probability theory that describes how to update the probability of a hypothesis based on new evidence. It allows you to calculate the probability of an event, given prior knowledge or conditions, and new data or evidence.

Bayes' theorem is given by the formula:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Where:

Bayes' Theorem allows us to update our beliefs about the probability of an event A A, taking into account new evidence B B. The formula adjusts the prior probability of A A based on how likely B B is when A A is true and how likely B B is overall.

- $P(A)$ is the prior probability, the initial probability of event A before seeing the evidence B

```
# Given values
P_A = 0.01  # Prior probability that the person has the disease
P_B_given_A = 0.9  # Likelihood that the person tests positive given they have the disease
P_B_given_not_A = 0.05  # Likelihood that the person tests positive given they don't have the
P_not_A = 0.99  # Prior probability that the person does not have the disease

# Step 1: Calculate P(B) (the total probability of testing positive)
P_B = (P_B_given_A * P_A) + (P_B_given_not_A * P_not_A)

# Step 2: Apply Bayes' Theorem to calculate P(A|B)
P_A_given_B = (P_B_given_A * P_A) / P_B

print(f"Probability that the person has the disease given a positive test result: {P_A_given_
```

```
Probability that the person has the disease given a positive test result: 0.1538
```

Start coding or generate with AI.

Start coding or generate with AI.