Lecture 5: Test of Normality, Probability, and P-value

Part 1: Diving into the Depths of Normality

Part 2: Embracing the World of Probability

Part 3: Mastering the Art of Arrangements

- **3.1 Permutations: Embracing Order in Arrangements**
  - Formula and Explanation
  - Python Implementation
  - R Implementation
- **3.2 Combinations: Finding Unique Groups**
  - Formula and Explanation
  - Python Implementation
  - R Implementation

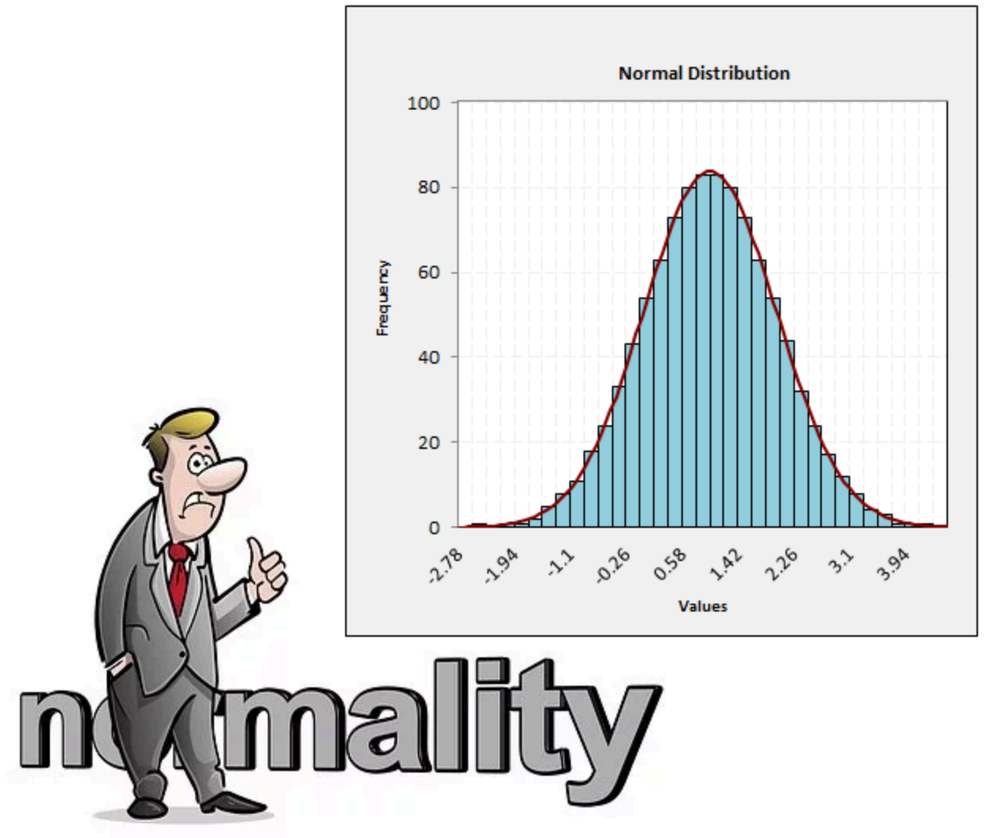
Part 4: Unlocking the Secrets of the P-value

- **4.1 The P-value: A Key to Hypothesis Testing**
- **4.2 Understanding the Null Hypothesis**
- **4.3 The Role of the P-value in Decision Making**
- **4.4 Interpreting the P-value: Low vs. High**
- **4.5 Key Points to Remember**

Introduction

Welcome to Lecture 5! Today, we'll delve into the crucial concept of normality testing in statistics. We'll explore why it's essential, especially when employing hypothesis tests like t-tests or ANOVA. We'll then move on to understanding probability, its rules, and how it relates to the p-value, a cornerstone of statistical inference. Finally, we'll implement these concepts in Python and R.

Part 1: Test of Normality



1.1 Why Test for Normality?

Many statistical tests, like t-tests and ANOVA, operate under the assumption that the data stems from a normally distributed population. This assumption is vital for the validity of the test results. If the data deviates significantly from a normal distribution, the results of these tests might be misleading.

1.2 What is a Normal Distribution?

A normal distribution, also known as a Gaussian distribution or bell curve, is a symmetrical distribution where the data clusters around the mean. Its shape is characterized by the bell-like curve, with the mean, median, and mode all being equal.

1.3 How to Test for Normality

There are two primary approaches to test for normality:

- **Analytical Tests:** These tests provide a p-value to assess normality.
  - **Kolmogorov-Smirnov Test:** A widely used test, but can be sensitive to large sample sizes.
  - **Shapiro-Wilk Test:** Generally considered more powerful than Kolmogorov-Smirnov, especially for smaller sample sizes.
  - **Anderson-Darling Test:** More sensitive to deviations in the tails of the distribution.

  Interpretation of P-value:
  - **p-value <= 0.05:** Reject the null hypothesis. The data is likely not normally distributed.
  - **p-value > 0.05:** Fail to reject the null hypothesis. We don't have enough evidence to say the data is not normally distributed.
- **Graphical Methods:** These methods provide visual aids to assess normality.
  - **Histogram:** Provides a visual representation of the data's distribution, allowing for a comparison with the bell curve shape.
  - **Q-Q Plot (Quantile-Quantile Plot):** Compares the theoretical quantiles of a normal distribution to the actual quantiles of the data. Data points closely following a straight diagonal line suggest normality.

1.4 Limitations of Analytical Tests

- **Sample Size Dependency:** The p-value is highly influenced by the sample size. Large samples may lead to rejecting normality even for minor deviations, while small samples might fail to detect deviations.

1.5 Advantages of Graphical Methods

- **Visual Interpretation:** Graphical methods provide a more intuitive understanding of the data distribution, regardless of sample size.

1.6 Implementation in Python and R

Let's demonstrate how to perform normality tests in Python and R using a sample dataset.

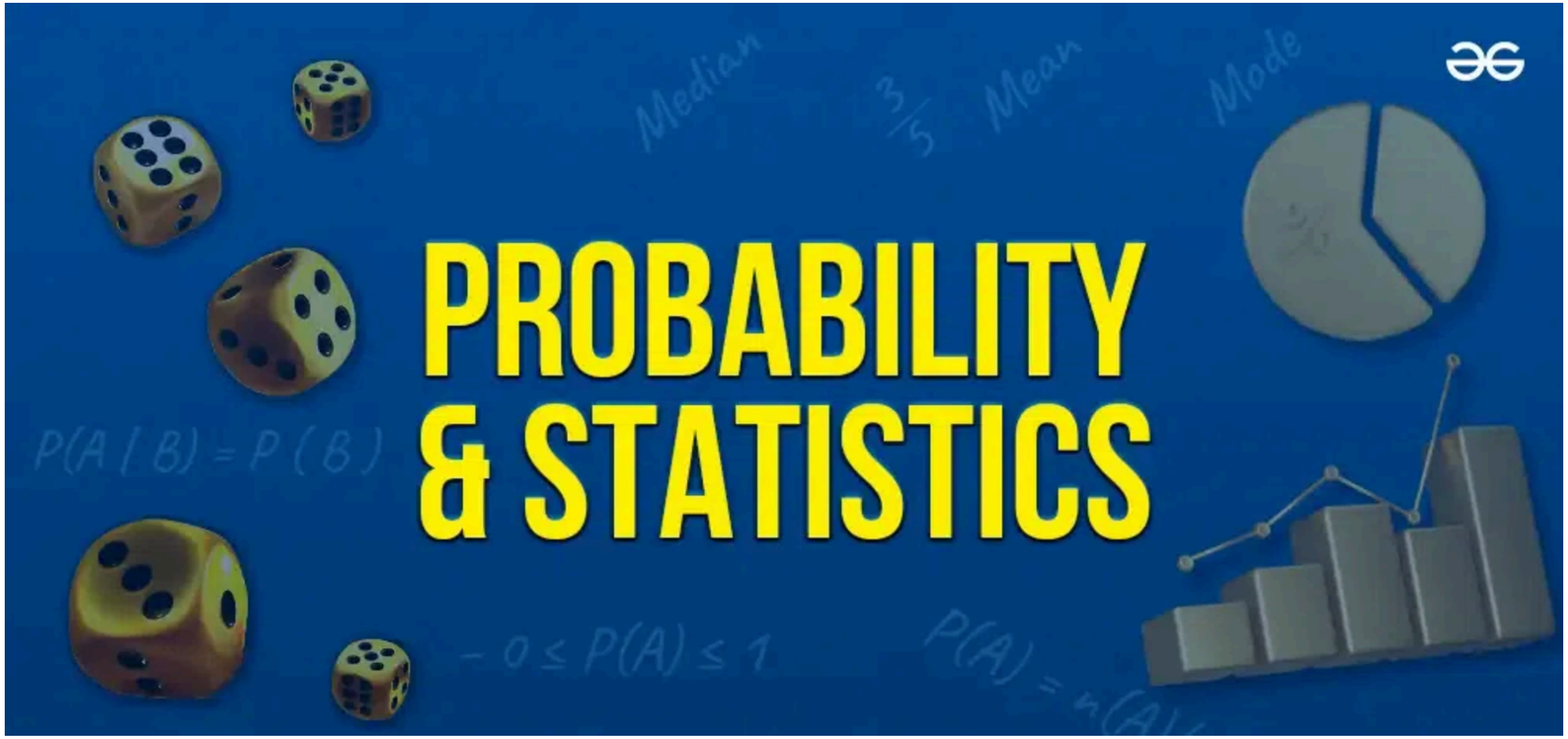
Python (using libraries like scipy.stats)

```
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt

# Load the Iris dataset
iris = pd.read_csv('Iris.csv')

# Select the 'SepalLengthCm' column for normality testing
sepal_length = iris['SepalLengthCm']

# Perform Shapiro-Wilk test
statistic, p_value = stats.shapiro(sepal_length)
print(f"Shapiro-Wilk Test: Statistic={statistic:.3f}, p-value={p_value:.3f}")

# Plot a histogram
plt.hist(sepal_length, bins=10, edgecolor='black')
plt.title('Histogram of Sepal Length')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Frequency')
plt.show()

# Create a Q-Q plot
stats.probplot(sepal_length, dist="norm", plot=plt)
plt.title('Q-Q Plot of Sepal Length')
plt.show()
```

R

```r
# Load the Iris dataset (assuming it's in your working directory)
iris <- read.csv("Iris.csv")

# Select the 'SepalLengthCm' column for normality testing
sepal_length <- iris$SepalLengthCm

# Perform Shapiro-Wilk test
shapiro_test <- shapiro.test(sepal_length)
print(shapiro_test)

# Plot a histogram
hist(sepal_length, breaks = 10, main = "Histogram of Sepal Length",
xlab = "Sepal Length (cm)", ylab = "Frequency")

# Create a Q-Q plot
qqnorm(sepal_length)
qqline(sepal_length)
```

Part 2: Probability, Additive Rule, and Multiplicative Rule



2.1 Introduction to Probability

Probability is a fundamental concept in statistics that helps us quantify uncertainty. It measures the likelihood of an event occurring. We often encounter situations where we need to reason about the chances of something happening. For example:

- **In machine learning:** When building a classification model, we're interested in the probability of a data point belonging to a specific class.
- **In business:** We might analyze the probability of a marketing campaign leading to a certain percentage increase in sales.

Formally, probability is defined as:

Probability (Event) = Number of favorable outcomes / Total number of possible outcomes

Example:

Imagine rolling a fair six-sided die. The probability of rolling a '4' is:

- **Favorable outcome:** Rolling a '4' (1 outcome)
- **Total possible outcomes:** 1, 2, 3, 4, 5, 6 (6 outcomes)

Therefore, P(rolling a '4') = 1/6

2.2 Additive Rule (The "OR" Rule)

Addition Rules for Probability

- Two events are **mutually exclusive events** if they cannot occur at the same time (i.e., they have no outcomes in common)

Addition Rules

$$P(A \text{ or } B) = P(A) + P(B) \quad \text{Mutually Exclusive}$$

$$P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B) \quad \text{Not M. E.}$$

1

The additive rule helps us calculate the probability of one event **OR** another event happening. There are two scenarios to consider:

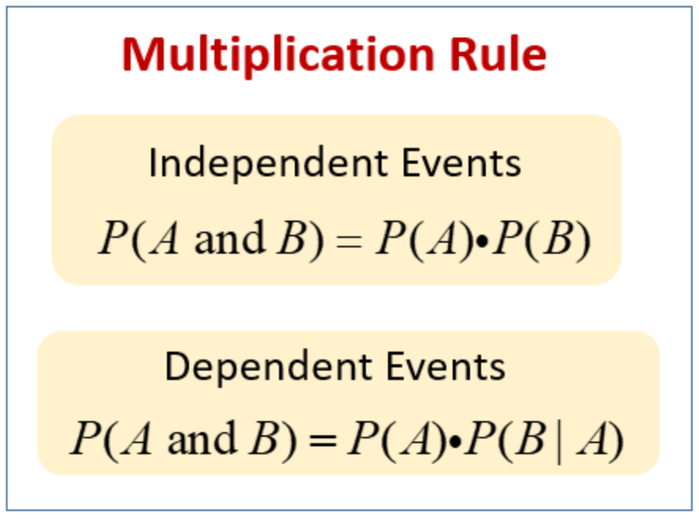
- **Mutually Exclusive Events:** Events that cannot occur simultaneously. For example, rolling a '3' and a '5' on a single roll of a die are mutually exclusive.
  In this case, the additive rule is:
  P(A or B) = P(A) + P(B)
- **Non-Mutually Exclusive Events:** Events that can occur at the same time. For example, drawing a queen and drawing a heart from a deck of cards are not mutually exclusive (you can draw the queen of hearts).

In this case, the additive rule is:

P(A or B) = P(A) + P(B) - P(A and B)

We subtract the probability of both events occurring to avoid double-counting.

2.3 Multiplicative Rule (The "AND" Rule)



The multiplicative rule calculates the probability of two or more events happening **together**. We again have two cases:

- **Independent Events:** Events where the occurrence of one event does not affect the probability of the other. For example, flipping a coin twice – the outcome of the first flip doesn't influence the second.
  The multiplicative rule for independent events is:
  P(A and B) = P(A) * P(B)
- **Dependent Events:** Events where the occurrence of one event affects the probability of the other. For instance, drawing two cards from a deck without replacement – the probability of drawing a certain card on the second draw depends on what was drawn first.
  The multiplicative rule for dependent events is:
  P(A and B) = P(A) * P(B|A)
  Where P(B|A) represents the conditional probability of event B happening *given* that event A has already occurred.

2.4 Examples and Implementation in Python and R:

Let's illustrate these rules with examples and implement them in Python and R:

Example 1 (Additive Rule - Mutually Exclusive):

What is the probability of rolling a '2' **OR** a '6' on a single roll of a die?

- P(rolling a '2') = 1/6
- P(rolling a '6') = 1/6

Since these are mutually exclusive:

P(rolling a '2' or a '6') = 1/6 + 1/6 = 1/3

Python:

```
p_2 = 1/6
p_6 = 1/6
p_2_or_6 = p_2 + p_6
print(f"Probability of rolling a 2 or a 6: {p_2_or_6:.3f}")
```

R:

```
p_2 <- 1/6
p_6 <- 1/6
p_2_or_6 <- p_2 + p_6
cat("Probability of rolling a 2 or a 6:", p_2_or_6, "\n")
```

Example 2 (Additive Rule - Non-Mutually Exclusive):

What is the probability of drawing a queen **OR** a heart from a standard deck of 52 cards?

- P(Queen) = 4/52 (there are 4 queens)
- P(Heart) = 13/52 (there are 13 hearts)
- P(Queen and Heart) = 1/52 (the queen of hearts)

P(Queen or Heart) = 4/52 + 13/52 - 1/52 = 16/52 = 4/13

Python:

```python
import pandas as pd

# Load the Titanic dataset
titanic = pd.read_csv('titanic.csv')

# Calculate probabilities
p_first_class = len(titanic[titanic['Pclass'] == 1]) / len(titanic)
p_survived = len(titanic[titanic['Survived'] == 1]) / len(titanic)
p_first_class_and_survived = len(titanic[(titanic['Pclass'] == 1) & (titanic['Survived'] == 1)]) / len(titanic)

# Apply the additive rule
p_first_class_or_survived = p_first_class + p_survived - p_first_class_and_survived

print(f"Probability of being in first-class or surviving: {p_first_class_or_survived:.3f}")
```

R:

```r
# Load the Titanic dataset (assuming it's in your working directory)
titanic <- read.csv("titanic.csv")

# Calculate probabilities
p_first_class <- nrow(titanic[titanic$Pclass == 1, ]) / nrow(titanic)
p_survived <- nrow(titanic[titanic$Survived == 1, ]) / nrow(titanic)
p_first_class_and_survived <- nrow(titanic[(titanic$Pclass == 1) & (titanic$Survived == 1), ]) / nrow(titanic)

# Apply the additive rule
p_first_class_or_survived <- p_first_class + p_survived - p_first_class_and_survived

cat("Probability of being in first-class or surviving:", p_first_class_or_survived, "\n")
```

Example 3 (Multiplicative Rule - Independent):

What is the probability of flipping a coin twice and getting heads both times?

- P(Heads on the first flip) = 1/2
- P(Heads on the second flip) = 1/2

P(Heads and Heads) = (1/2) * (1/2) = 1/4

Python:

```
p_heads = 1/2
p_heads_heads = p_heads * p_heads
print(f"Probability of two heads in a row: {p_heads_heads:.3f}")
```

R:

```
p_heads <- 1/2
p_heads_heads <- p_heads * p_heads
cat("Probability of two heads in a row:", p_heads_heads, "\n")
```

Example 4 (Multiplicative Rule - Dependent):

In a bag with 3 red marbles and 2 blue marbles, what is the probability of drawing a red marble, **then** another red marble without putting the first one back?

- P(Red on the first draw) = 3/5
- P(Red on the second draw | Red on the first draw) = 2/4 (since one red marble is gone)

P(Red and Red) = (3/5) * (2/4) = 3/10

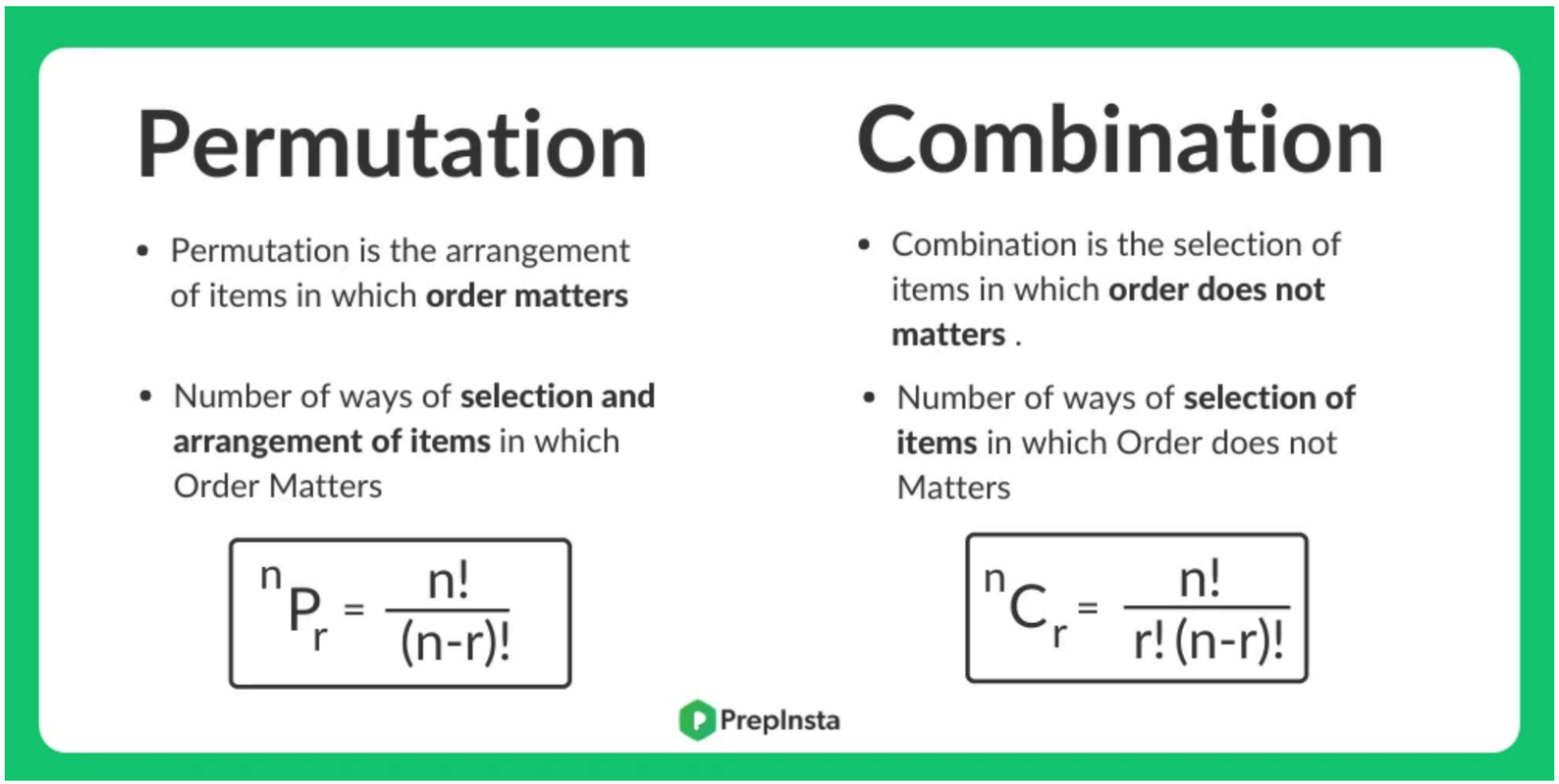
Python:

```
import pandas as pd

# Load the Titanic dataset
titanic = pd.read_csv('titanic.csv')

# Calculate probabilities
p_female = len(titanic[titanic['Sex'] == 'female']) / len(titanic)
p_survived_given_female = len(titanic[(titanic['Sex'] == 'female') & (titanic['Survived'] == 1)]) / len(titanic[titanic['Sex'] == 'female'])

# Apply the multiplicative rule for dependent events
p_female_and_survived = p_female * p_survived_given_female

print(f"Probability of being female and surviving: {p_female_and_survived:.3f}")
```

R:

```r
# Load the Titanic dataset
titanic <- read.csv("titanic.csv")

# Calculate probabilities
p_female <- nrow(titanic[titanic$Sex == "female", ]) / nrow(titanic)
p_survived_given_female <- nrow(titanic[(titanic$Sex == "female") & (titanic$Survived == 1), ]) / nrow(titanic[titanic$Sex == "female", ])

# Apply the multiplicative rule for dependent events
p_female_and_survived <- p_female * p_survived_given_female

cat("Probability of being female and surviving:", p_female_and_survived, "\n")
```

Part 3: Permutation and Combination



3.1 Permutation

Permutation refers to the arrangement of objects in a specific order. In simpler terms, it's about finding how many different ways you can order a set of items.

Example:

Imagine you have 6 different chocolates, and you want to arrange 3 of them in a row. How many different arrangements are possible?

- For the first spot, you have 6 choices.
- Once you've chosen the first chocolate, you have 5 left for the second spot.
- For the third spot, you have 4 chocolates remaining.

Therefore, the total number of permutations is 6 * 5 * 4 = 120.

Formula:

The number of permutations of *n* objects taken *r* at a time is denoted as nPr and calculated as:

nPr = n! / (n - r)!

where:

- n = total number of objects
- r = number of objects taken at a time
- ! denotes factorial (e.g., 5! = 5 * 4 * 3 * 2 * 1)

Python Implementation:

```python
import math

n = 6 # Total chocolates
r = 3 # Chocolates to arrange

permutations = math.factorial(n) / math.factorial(n - r)
print(f"Number of permutations: {permutations}")
```

R Implementation:

```r
n <- 6 # Total chocolates
r <- 3 # Chocolates to arrange

permutations <- factorial(n) / factorial(n - r)
cat("Number of permutations:", permutations, "\n")
```

3.2 Combination

Combination, on the other hand, deals with the selection of objects without regard to order. It's about forming groups or subsets from a larger set.

Example:

Using the same chocolate example, let's say you want to choose 3 chocolates out of 6, but the order you pick them doesn't matter. How many different groups of 3 can you form?

Notice that in this case, the arrangements (Dairy Milk, Gems, Eclairs) and (Gems, Eclairs, Dairy Milk) represent the same group of chocolates.

Formula:

The number of combinations of *n* objects taken *r* at a time is denoted as nCr and calculated as:

nCr = n! / (r! * (n - r)!)

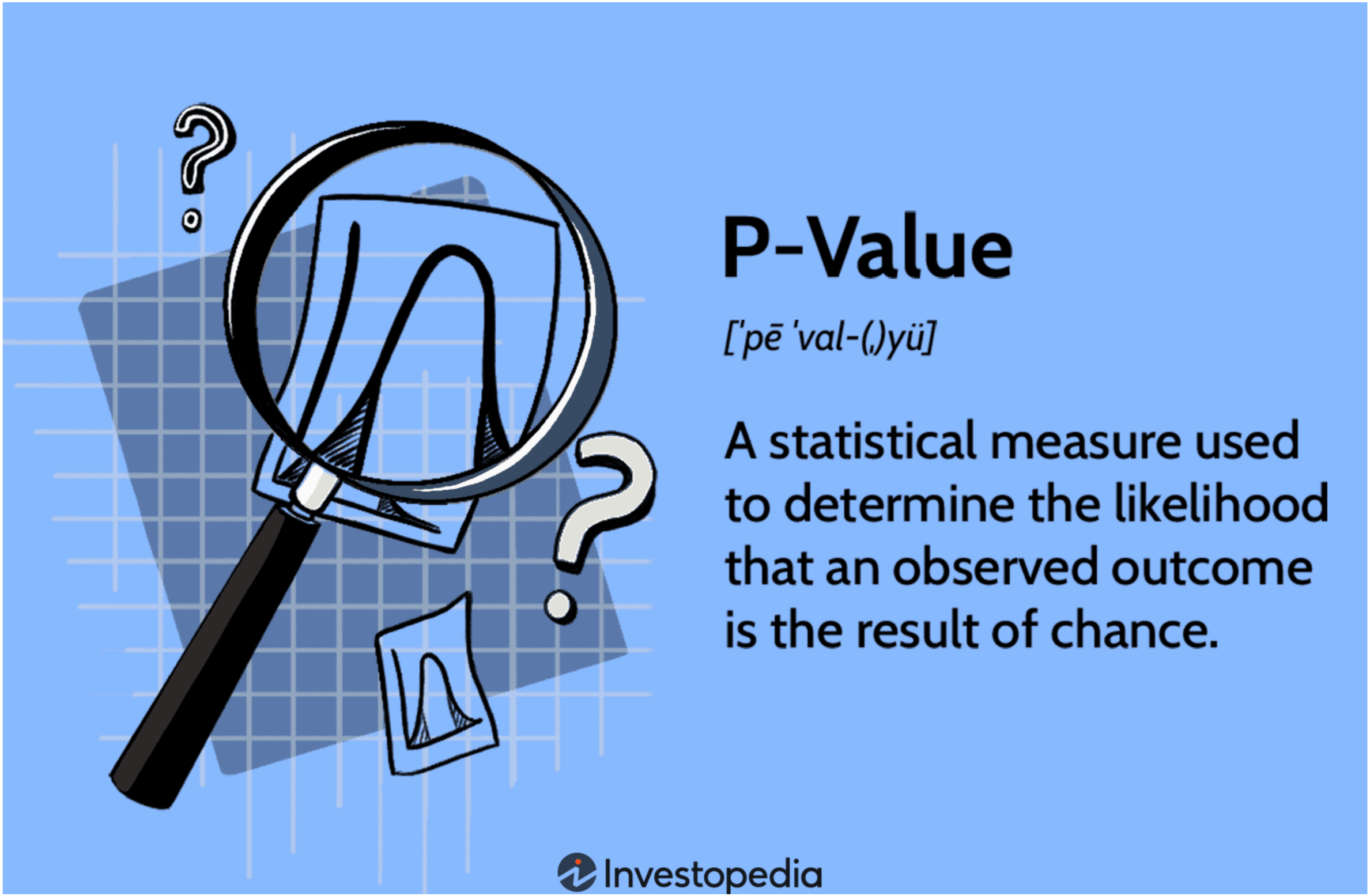
Python Implementation:

```
import math

n = 6 # Total chocolates
r = 3 # Chocolates to choose

combinations = math.factorial(n) / (math.factorial(r) * math.factorial(n - r))
print(f"Number of combinations: {combinations}")
```

R Implementation:

```
n <- 6 # Total chocolates
r <- 3 # Chocolates to choose

combinations <- factorial(n) / (factorial(r) * factorial(n - r))
cat("Number of combinations:", combinations, "\n")
```

Part 4: P-value


P-Value
['pē 'val-(,)yü]
A statistical measure used to determine the likelihood that an observed outcome is the result of chance.
Investopedia

4.1 Understanding the Concept

The p-value is a fundamental concept in hypothesis testing. It helps us determine the strength of evidence against a null hypothesis.

What is a Null Hypothesis?

Before we delve into the p-value, let's briefly discuss the null hypothesis. In hypothesis testing, we start with an assumption called the null hypothesis (denoted as H0). It typically states that there is no effect or no difference between groups.

The Role of the P-value:

The p-value comes into play when we collect data and want to see if the data provides enough evidence to reject the null hypothesis. It quantifies the probability of observing our data (or more extreme data) if the null hypothesis were true.

Interpretation:

- **Low P-value (typically less than or equal to 0.05):** This suggests that the observed data is unlikely to have occurred by chance alone if the null hypothesis were true. We have strong evidence against the null hypothesis, and we often reject it in favor of an alternative hypothesis.
- **High P-value (typically greater than 0.05):** This indicates that the observed data is reasonably likely to occur even if the null hypothesis were true. We don't have enough evidence to reject the null hypothesis.

Analogy:

Imagine you have a bag of 100 coins, and you suspect that it might contain more than just fair coins.

- **Null Hypothesis (H0):** The bag contains only fair coins.
- **Experiment:** You draw 10 coins randomly, and 9 of them come up heads.

This result is quite unusual if the bag contained only fair coins. The p-value would quantify how unusual this result is. A very low p-value would suggest that the bag likely contains some biased coins, leading you to reject the null hypothesis.

Key Points:

- The p-value is a probability, ranging from 0 to 1.
- A smaller p-value indicates stronger evidence against the null hypothesis.
- The choice of significance level (alpha, usually 0.05) is arbitrary but commonly used.
- The p-value does not tell us the probability of the null hypothesis being true or false. It only measures the strength of evidence against the null hypothesis based on the observed data.