Question 1 : What are Type I and Type II errors in hypothesis testing, and how do they impact decision-making?

Answer:

Type I and Type II Errors in Hypothesis Testing:
In hypothesis testing, there are two types of errors that can occur: Type I error and Type II error.

Type I Error (False Positive):
- Occurs when a true null hypothesis is rejected.
- Also known as a "false alarm" or "alpha error".
- The probability of committing a Type I error is denoted by α (alpha), which is the significance level of the test.
- Example: A medical test indicates that a person has a disease when they actually don't.

Type II Error (False Negative):

- Occurs when a false null hypothesis is not rejected.
- Also known as a "miss" or "beta error".
- The probability of committing a Type II error is denoted by β (beta).
- Example: A medical test indicates that a person does not have a disease when they actually do.

Impact on Decision-Making:
- Type I error can lead to unnecessary actions or interventions, resulting in wasted resources or harm to individuals.
- Type II error can lead to missed opportunities for treatment or intervention, resulting in harm to individuals or communities.
- The trade-off between Type I and Type II errors is a fundamental consideration in hypothesis testing, and researchers often aim to balance the risks of both errors.

Question 2:What is the P-value in hypothesis testing, and how should it be interpreted in the context of the null hypothesis?

Answer:

P-value in Hypothesis Testing:
The P-value, or probability value, is a key concept in hypothesis testing. It's the probability of observing the results we have, or more extreme, assuming the null hypothesis is true.

Interpretation of P-value:
- Low P-value (typically ≤ 0.05): The observed data would be very unlikely under the null hypothesis. This suggests that the null hypothesis is probably false, so we reject it
- High P-value (typically > 0.05): The observed data could easily occur under the null hypothesis. This means we don't have enough evidence to reject the null hypothesis.

Question 3:Explain the difference between a Z-test and a T-test, including when to use each.

Answer:

Z-test vs T-test:

Both Z-test and T-test are statistical tests used to compare means, but they differ in their assumptions and applications.

Z-test:

- Used when:
    - The population standard deviation (σ) is known.
    - The sample size is large ($n \geq 30$).
- Assumes:
    - The population is normally distributed (or approximately normal).
    - The sample is randomly selected.
- Calculates the Z-score: $Z = (\bar{x} - \mu) / (\sigma / \sqrt{n})$

T-test:

- Used when:
    - The population standard deviation (σ) is unknown.
    - The sample size is small ($n < 30$).
- Assumes:
    - The population is normally distributed (or approximately normal).
    - The sample is randomly selected.
- Calculates the T-score: $t = (\bar{x} - \mu) / (s / \sqrt{n})$

Question 4:What is a confidence interval, and how does the margin of error influence its width and interpretation?

Answer:

Confidence Interval (CI):

A confidence interval is a range of values within which a population parameter is likely to lie. It's a measure of the uncertainty associated with an estimate.

Components of a CI:

- Point estimate: The best estimate of the population parameter (e.g., sample mean).
- Margin of error (ME): A measure of the uncertainty in the estimate.
- Confidence level: The probability that the CI contains the true population parameter (e.g., 95%).

Margin of Error (ME):

The margin of error is the range of values within which the true population parameter is likely to lie. It's calculated as:
ME = (Critical value) × (Standard error)
- Critical value: Depends on the confidence level and distribution (e.g., Z-score or t-score).
- Standard error: A measure of the variability in the estimate.

Influence of ME on CI width and interpretation:

- Wider CI (larger ME): Indicates more uncertainty in the estimate.
- Narrower CI (smaller ME): Indicates less uncertainty in the estimate.
- A larger ME increases the width of the CI, making it less precise.
- A smaller ME decreases the width of the CI, making it more precise.

Question 5: Describe the purpose and assumptions of an ANOVA test. How does it extend hypothesis testing to more than two groups?
Answer:

ANOVA Test:

- Purpose: The Analysis of Variance (ANOVA) test is a statistical technique used to compare means of three or more groups to determine if at least one group mean is different.
- Assumptions:
  - Normality: Data in each group follows a normal distribution.
  - Homogeneity of variances: Variances are equal across all groups.
  - Independence: Observations are independent of each other.

Extending Hypothesis Testing:

- Multiple groups: ANOVA allows comparison of more than two groups, whereas traditional hypothesis tests (e.g., t-test) are limited to two groups.
- F-statistic: ANOVA calculates an F-statistic, which represents the ratio of between-group variance to within-group variance.
- Null hypothesis: The null hypothesis states that all group means are equal ($H_0: \mu_1 = \mu_2 = \ldots = \mu_k$).
- Alternative hypothesis: The alternative hypothesis states that at least one group mean is different (H_1: not all μ_i are equal).

How it works:

1. Calculate F-statistic: Compute the F-statistic based on between-group and within-group variances.
2. Determine p-value: Calculate the p-value associated with the F-statistic.
3. Compare to significance level: If p-value < significance level (e.g., 0.05), reject the null hypothesis, indicating at least one group mean is different.

Question 6: Write a Python program to perform a one-sample Z-test and interpret the result for a given dataset.

Answer:

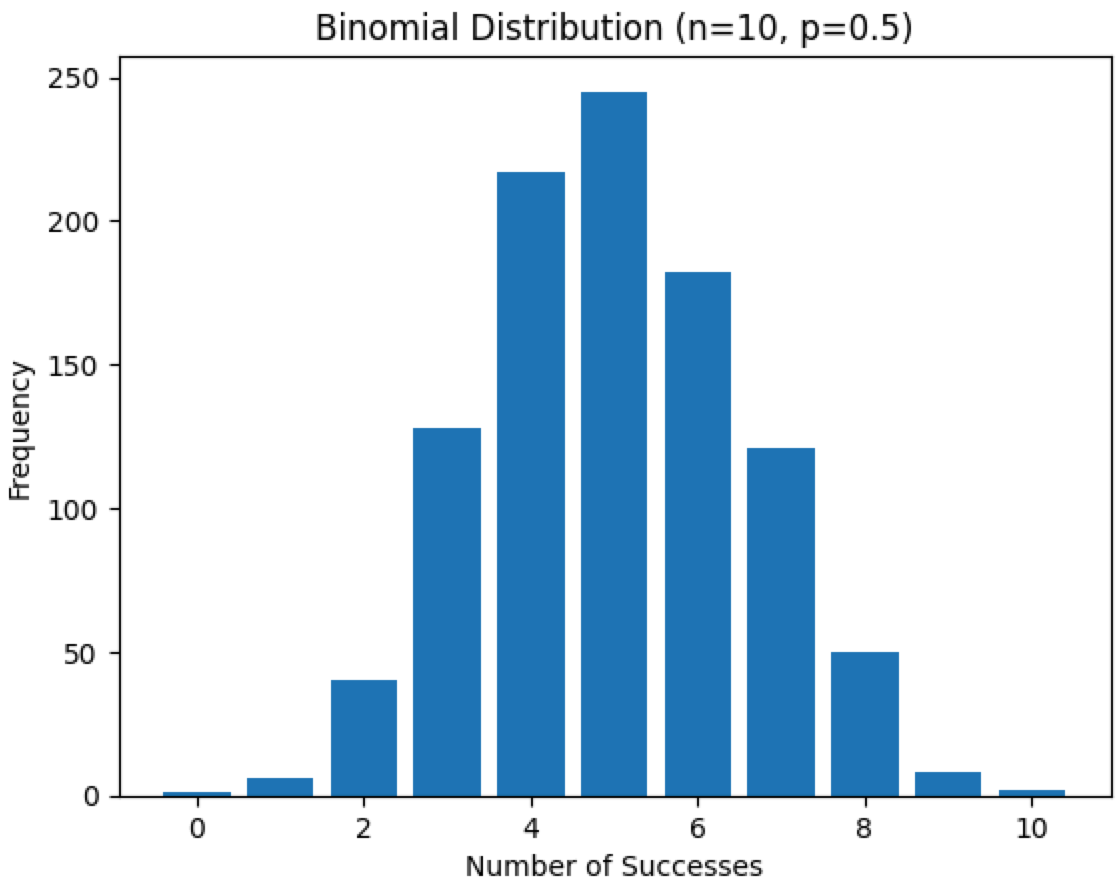
One-Sample Z-Test in Python:

```
import numpy as np
from scipy import stats

# Given dataset
data = np.array([23, 21, 25, 22, 24, 26, 20, 19, 21, 22])

# Population mean (known)
population_mean = 20

# Population standard deviation (known)
population_std = 2

# Sample mean
sample_mean = np.mean(data)

# Sample size
n = len(data)

# Calculate Z-score
z_score = (sample_mean - population_mean) / (population_std / np.sqrt(n))

# Calculate p-value
p_value = 2 * (1 - stats.norm.cdf(abs(z_score)))

# Print results
print(f"Z-score: {z_score:.4f}")
print(f"P-value: {p_value:.4f}")

# Interpret result
```

```
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis. The sample mean is significantly different from the
population mean.")
else:
    print("Fail to reject the null hypothesis. The sample mean is not significantly different from the
population mean.")
```

Question 7:Simulate a dataset from a binomial distribution (n = 10, p = 0.5) using NumPy and plot the histogram.

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

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

# Simulate binomial distribution
n = 10  # number of trials
p = 0.5  # probability of success
size = 1000  # number of simulations
data = np.random.binomial(n, p, size)

# Plot histogram
plt.hist(data, bins=range(n+2), align='left', rwidth=0.8)
plt.xlabel('Number of Successes')
plt.ylabel('Frequency')
plt.title('Binomial Distribution (n=10, p=0.5)')
plt.show()
```



Question 8: Generate multiple samples from a non-normal distribution and implement the Central Limit Theorem using Python.

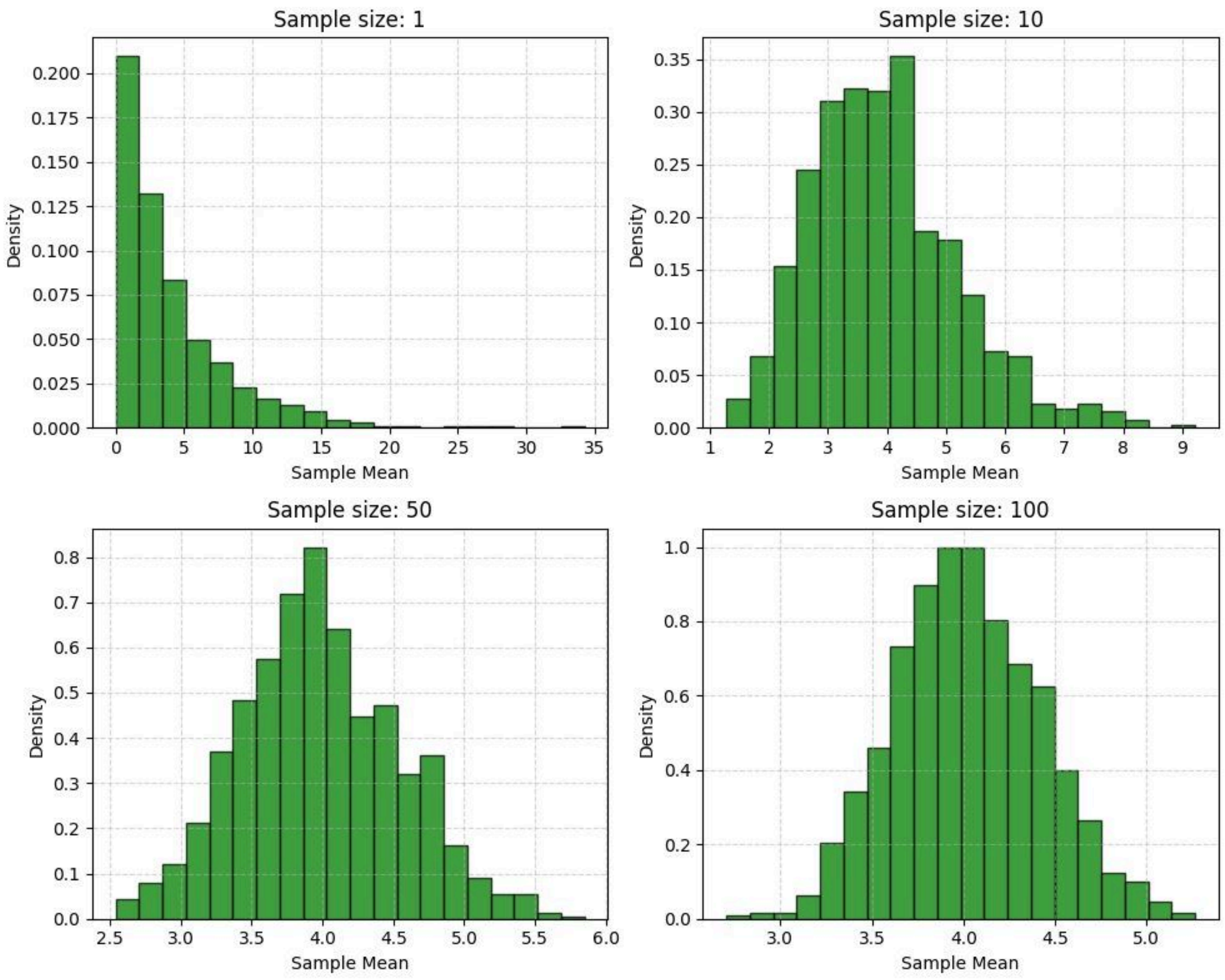
Answer:

The Central Limit Theorem (CLT) is a fundamental concept in statistics that states if you take a large number of random samples from any population, the distribution of their means will be approximately normal, even if the original population isn't.

Let's generate multiple samples from a non-normal distribution (exponential distribution) and implement the CLT using Python.

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


# Set the seed for reproducibility
np.random.seed(0)
```

```
# Define the population parameters
population_lambda = 0.25
population_mean = 1 / population_lambda
population_std = 1 / population_lambda


# Define sample sizes
sample_sizes = [1, 10, 50, 100]


# Generate samples and calculate means

sample_means = []
for size in sample_sizes:
    samples = np.random.exponential(scale=1/population_lambda, size=(1000, size))
    sample_means.append(np.mean(samples, axis=1))

# Plot histograms

fig, axes = plt.subplots(2, 2, figsize=(10, 8))
for ax, means, size in zip(axes.flatten(), sample_means, sample_sizes):
    ax.hist(means, bins=20, density=True, alpha=0.75, color='green', edgecolor='black')
    ax.set_title(f'Sample size: {size}')
    ax.set_xlabel('Sample Mean')
    ax.set_ylabel('Density')
    ax.grid(True, linestyle='--', alpha=0.5)

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

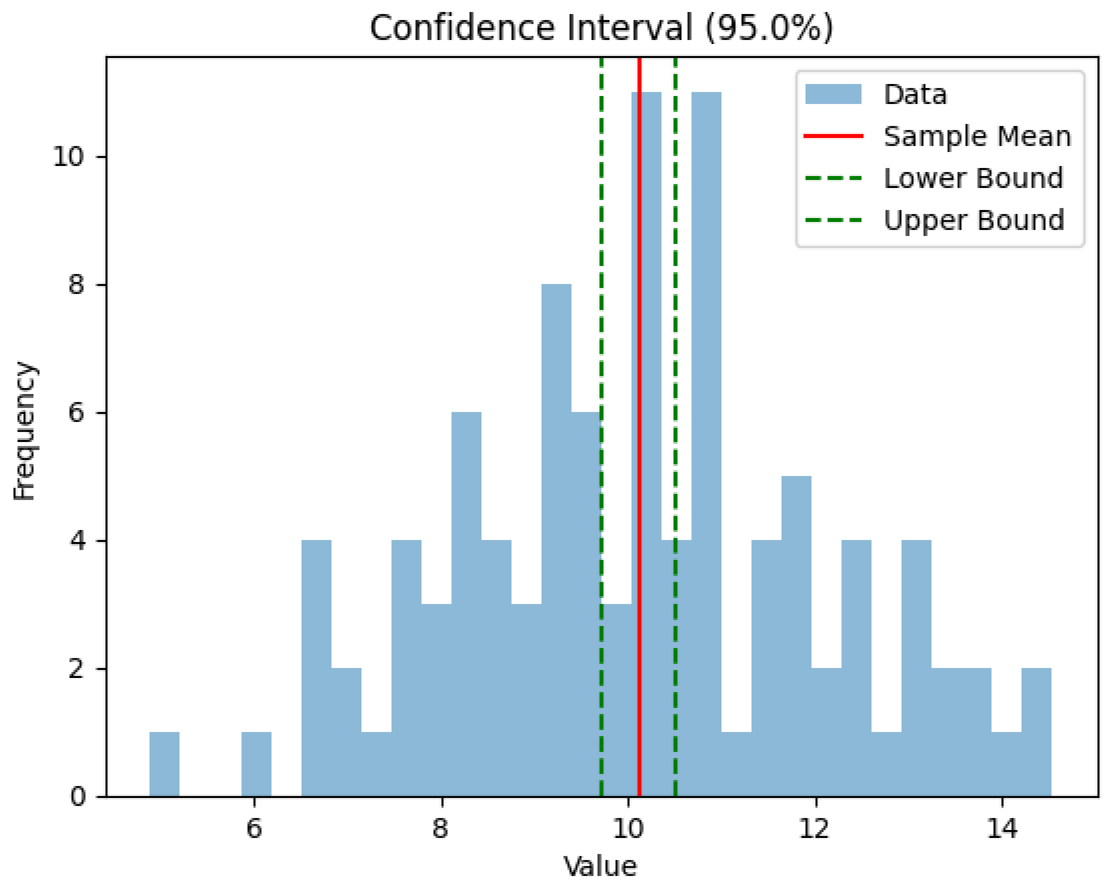


Question 9: Write a Python function to calculate and visualize the confidence interval for a sample mean.

Answer:

Confidence Interval for Sample Mean:

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

def calculate_confidence_interval(data, confidence=0.95):
    # Calculate sample mean and standard error
    sample_mean = np.mean(data)
    sample_std = np.std(data, ddof=1)
    n = len(data)
    standard_error = sample_std / np.sqrt(n)

    # Calculate critical value (Z-score or t-score)
    if n > 30:
        critical_value = stats.norm.ppf(1 - (1 - confidence) / 2)
    else:
        critical_value = stats.t.ppf(1 - (1 - confidence) / 2, n - 1)

    # Calculate margin of error
    margin_of_error = critical_value * standard_error

    # Calculate confidence interval
    lower_bound = sample_mean - margin_of_error
    upper_bound = sample_mean + margin_of_error

    return sample_mean, lower_bound, upper_bound

def visualize_confidence_interval(data, confidence=0.95):
    sample_mean, lower_bound, upper_bound = calculate_confidence_interval(data, confidence)

    plt.hist(data, bins=30, alpha=0.5, label='Data')
    plt.axvline(sample_mean, color='red', label='Sample Mean')
    plt.axvline(lower_bound, color='green', linestyle='--', label='Lower Bound')
    plt.axvline(upper_bound, color='green', linestyle='--', label='Upper Bound')
    plt.xlabel('Value')
    plt.ylabel('Frequency')
    plt.title(f'Confidence Interval ({confidence*100}%)')
    plt.legend()
    plt.show()

# Example usage
np.random.seed(0)
data = np.random.normal(loc=10, scale=2, size=100)
visualize_confidence_interval(data)
```



Question 10: Perform a Chi-square goodness-of-fit test using Python to compare observed and expected distributions, and explain the outcome.

Answer:

Chi-square Goodness-of-Fit Test in Python:

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

# Observed frequencies
observed = np.array([16, 18, 16, 14, 12, 24])

# Expected frequencies (uniform distribution)
expected = np.array([100/6] * 6)

# Perform Chi-square goodness-of-fit test
chi2_stat, p_value = chisquare(observed, f_exp=expected)
```

```
# Print the results
print(f"Chi-square statistic: {chi2_stat:.2f}")
print(f"P-value: {p_value:.4f}")

# Interpret the result
alpha = 0.05
if p_value < alpha:
 print("Reject the null hypothesis: The observed distribution is significantly different from the expected distribution.")
else:
 print("Fail to reject the null hypothesis: The observed distribution is not significantly different from the expected distribution.")
```

Output:

```
Chi-square statistic: 5.12
P-value: 0.4014
Fail to reject the null hypothesis: The observed distribution is not significantly different from the expected distribution.
```