Name : Ravi kumar yadav
E-mail : kumaryadavravi016@gmail.com
Assignment name : Data structure
Drive : drive
Github : Github

1 Assignment of Satistics Adavance -2

2 Theory

2.0.1 1) What is hypothesis testing in statistics

Hypothesis testing is a statistical method used to make inferences or draw conclusions about a population based on a sample of data. It involves testing a claim (the hypothesis) using sample data to decide whether to reject or fail to reject it.

2.0.2 2) What is the null hypothesis, and how does it differ from the alternative hypothesis ?

(i) Null Hypothesis (H): The null hypothesis states that there is no effect or no difference in the population. It serves as a starting point for statistical testing.

(ii) Alternative Hypothesis (H): The alternative hypothesis suggests that there is an effect or difference, opposing the null hypothesis. It represents the researcher's hypothesis or claim

2.0.3 3) What is the significance level in hypothesis testing, and why is it important?

The significance level ( ) is the threshold for deciding whether a hypothesis test result is statistically significant. It represents the probability of rejecting the null hypothesis when it is actually true (Type I error). Common significance levels are 0.05 or 0.01. It's important because it controls the risk of making incorrect conclusions

2.0.4 4) What does a P-value represent in hypothesis testing?

The P-value is the probability of obtaining a result at least as extreme as the one observed, assuming that the null hypothesis is true. It quantifies the strength of evidence against the null hypothesis.

2.0.5 5) How do you interpret the P-value in hypothesis testing ?

(i) If the P-value is less than the significance level (e.g., 0.05), you reject the null hypothesis, indicating strong evidence against it.

(ii)If the P-value is greater than or equal to the significance level, you fail to reject the null hypothesis, suggesting insufficient evidence to support the alternative hypothesis

2.0.6 6) What are Type 1 and Type 2 errors in hypothesis testing ?

(i) Type 1 error (False positive): Occurs when the null hypothesis is rejected when it is actually true.

(ii) Type 2 error (False negative): Occurs when the null hypothesis is not rejected when it is actually false.

2.0.7 7) What is the difference between a one-tailed and a two-tailed test in hypothesis testing ?

(i) One-tailed test: Tests if a parameter is greater than or less than a certain value (only in one direction).

(ii)Two-tailed test: Tests if a parameter is different from a certain value (in both directions, greater and lesser).

2.0.8 8) What is the Z-test, and when is it used in hypothesis testing?

A Z-test is used when you are testing the population mean (or proportions) when the population variance is known or the sample size is large (usually n $>$ 30). It's used to compare sample data against a population with a known standard deviation

2.0.9 9) How do you calculate the Z-score, and what does it represent in hypothesis testing ?

The Z-score is calculated using:

Z= X− /(/ n)

where X is the sample mean, is the population mean, is the population standard deviation, and n is the sample size.

It represents how many standard deviations a sample mean is away from the population mean

2.0.10 10) What is the T-distribution, and when should it be used instead of the normal distribution ?

The T-distribution is used when the sample size is small (n 30) and/or the population standard deviation is unknown. It has heavier tails than the normal distribution, which helps account for increased variability in smaller samples.

2.0.11 11) What is the difference between a Z-test and a T-test ?

A Z-test is used when the population variance is known, or the sample size is large.

A T-test is used when the population variance is unknown and the sample size is small.

2.0.12 12) What is the T-test, and how is it used in hypothesis testing ?

The T-test is used to compare the means of a sample to the population mean (or to compare two sample means) when the sample size is small or the population standard deviation is unknown

2.0.13 13) What is the relationship between Z-test and T-test in hypothesis testing ?

The Z-test and T-test are similar in that both test hypotheses about means. The key difference is that the Z-test assumes the population standard deviation is known or the sample size is large, whereas the T-test is used when the standard deviation is unknown, and the sample size is small

2.0.14 14) What is a confidence interval, and how is it used to interpret statistical results ?

A confidence interval is a range of values that is likely to contain the true population parameter with a certain level of confidence (e.g., 95%). It provides an estimate of the uncertainty around the sample statistic.

2.0.15 15) What is the margin of error, and how does it affect the confidence interval ?

The margin of error is the range within which the true population parameter is expected to lie. It is calculated as:

Margin of error=Z /2 $\times$ (/n)

where /2 is the Z-score corresponding to the confidence level. A larger margin of error makes the confidence interval wider, indicating more uncertainty.

2.0.16 16) How is Bayes' Theorem used in statistics, and what is its significance ?

Bayes' Theorem is used to update the probability of a hypothesis based on new evidence or data. It is significant because it allows for the incorporation of prior knowledge into statistical analysis, providing more accurate predictions or estimations.

2.0.17 17) What is the Chi-square distribution, and when is it used ?

The Chi-square distribution is used in tests of independence and goodness-of-fit tests. It is applied when comparing observed data to expected data, and when dealing with categorical variables.

2.0.18 18) What is the Chi-square goodness of fit test, and how is it applied ?

The Chi-square goodness of fit test compares the observed frequencies of categories to expected frequencies to determine whether there is a significant difference. It is commonly used in categorical data analysis.

2.0.19 19) What is the F-distribution, and when is it used in hypothesis testing ?

The F-distribution is used in the context of variance analysis and comparing two or more variances. It's most commonly used in ANOVA and regression analysis to test if variances between groups are significantly different.

2.0.20 20) What is an ANOVA test, and what are its assumptions ?

The ANOVA (Analysis of Variance) test compares means across multiple groups to see if at least one group mean is different. Its assumptions include:

(i)Independence of observations

(ii) Normally distributed populations

(iii)Homogeneity of variances (equal variances across groups)

2.0.21 21) What are the different types of ANOVA tests ?

(i) One-way ANOVA: Compares means of three or more independent groups based on one factor.

(ii) Two-way ANOVA: Compares means across groups based on two factors, and it can also assess interaction effects between the factors.

(iii)Repeated measures ANOVA: Used when the same subjects are measured multiple times.

2.0.22 22) What is the F-test, and how does it relate to hypothesis testing?

The F-test is used to compare two variances to determine if they are significantly different. In the context of ANOVA, the F-test is used to test if there are significant differences between the group means.

This overview should help you grasp the main concepts in hypothesis testing and related statistical tests! Let me know if you'd like further details on any topic

[ ]:

3 Practical

3.0.1 1) Write a Python program to perform a Z-test for comparing a sample mean to a known population mean and interpret the results ?

[24]:
```
import numpy as np
import scipy.stats as stats

# Sample data (random data generation for example)
```

```
sample_data = np.random.normal(loc=100, scale=15, size=50)  # mean=100, std=15,␣
 ↪sample size=50
population_mean = 100  # Known population mean
sample_mean = np.mean(sample_data)
sample_std = np.std(sample_data, ddof=1)  # Sample standard deviation
n = len(sample_data)

# Z-test calculation
z_score = (sample_mean - population_mean) / (sample_std / np.sqrt(n))
p_value = 2 * (1 - stats.norm.cdf(abs(z_score)))  # Two-tailed test

print(f"Z-score: {z_score}")
print(f"P-value: {p_value}")

# Interpretation
if p_value < 0.05:
    print("Reject the null hypothesis: There is a significant difference.")
else:
    print("Fail to reject the null hypothesis: No significant difference.")
```

```
Z-score: -0.3745165217260789
P-value: 0.7080200669559742
Fail to reject the null hypothesis: No significant difference.
```

3.0.2 2) Simulate random data to perform hypothesis testing and calculate the corresponding P-value using Python .

[26]:
```
# Simulate data
np.random.seed(0)
sample_data = np.random.normal(loc=100, scale=15, size=100)

# Population mean (null hypothesis)
population_mean = 100

# One-sample Z-test
sample_mean = np.mean(sample_data)
sample_std = np.std(sample_data, ddof=1)
n = len(sample_data)

z_score = (sample_mean - population_mean) / (sample_std / np.sqrt(n))
p_value = 2 * (1 - stats.norm.cdf(abs(z_score)))

print(f"Z-score: {z_score}")
print(f"P-value: {p_value}")
```

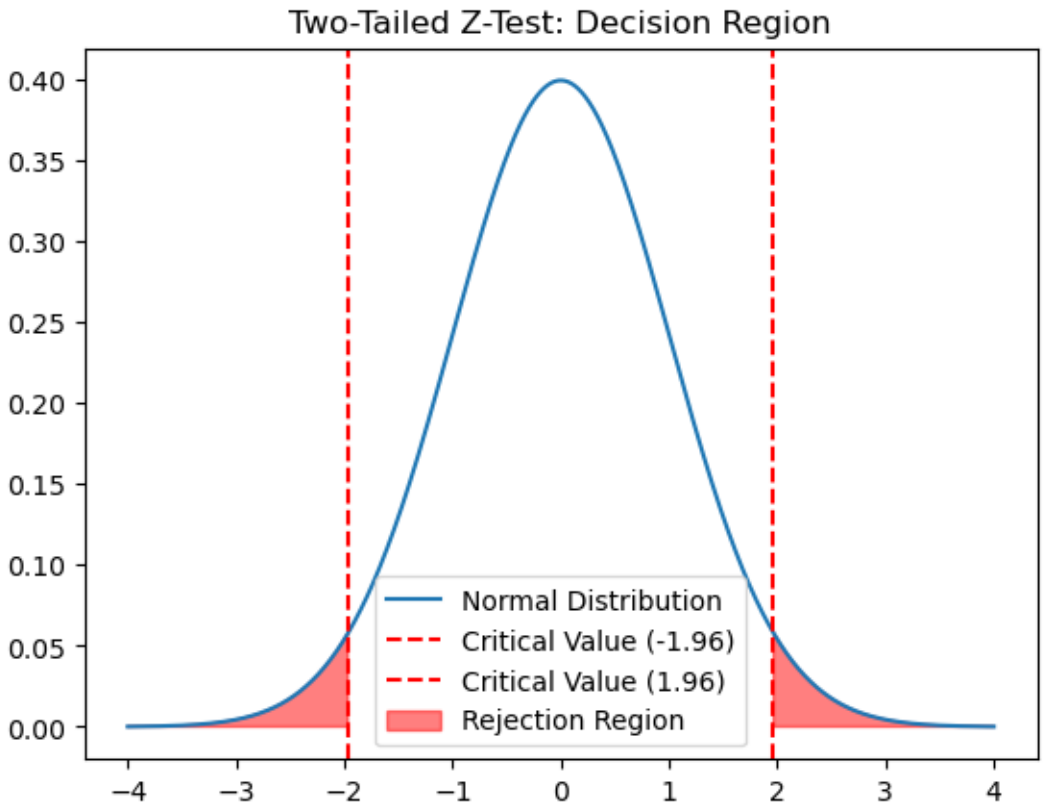
```
Z-score: 0.5904283402851795
P-value: 0.5549035151647161
```

3.0.3 3) Implement a one-sample Z-test using Python to compare the sample mean with the population mean.

[ ]:

3.0.4 4) Perform a two-tailed Z-test using Python and visualize the decision region on a plot.

[28]:
```
import matplotlib.pyplot as plt

# Two-tailed Z-test
z_critical = stats.norm.ppf(1 - 0.025)  # Critical value for two-tailed test␣
 ↪(alpha=0.05)

# Plotting decision region
x = np.linspace(-4, 4, 1000)
y = stats.norm.pdf(x)

plt.plot(x, y, label='Normal Distribution')
plt.axvline(x=-z_critical, color='r', linestyle='--', label=f'Critical Value␣
 ↪(-{z_critical:.2f})')
plt.axvline(x=z_critical, color='r', linestyle='--', label=f'Critical Value␣
 ↪({z_critical:.2f})')

# Highlight the rejection regions
plt.fill_between(x, y, where=(x <= -z_critical) | (x >= z_critical),␣
 ↪color='red', alpha=0.5, label="Rejection Region")
plt.title("Two-Tailed Z-Test: Decision Region")
plt.legend()
plt.show()

print(f"Z-Score: {z_score}")
print(f"P-Value: {p_value}")
```

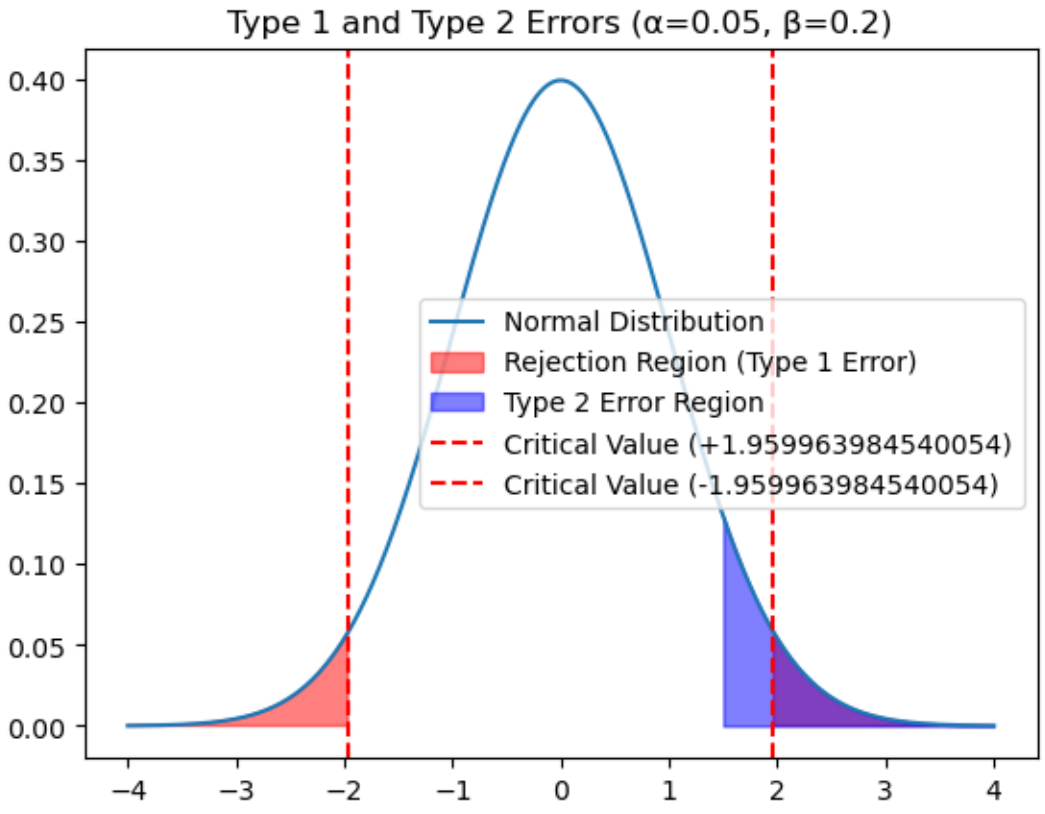



```
Z-Score: 0.5904283402851795
P-Value: 0.5549035151647161
```

3.0.5 5) Create a Python function that calculates and visualizes Type 1 and Type 2 errors during hypothesis testing.

[30]:
```python
def plot_errors(alpha, beta, z_critical, z_score):
    # Type 1 error (alpha): Rejecting H0 when it's true
    # Type 2 error (beta): Failing to reject H0 when it's false

    x = np.linspace(-4, 4, 1000)
    y = stats.norm.pdf(x)

    # Plot the normal distribution
    plt.plot(x, y, label='Normal Distribution')

    # Decision regions for Type 1 and Type 2 errors
    plt.fill_between(x, y, where=(x <= -z_critical) | (x >= z_critical),␣
↪color='red', alpha=0.5, label="Rejection Region (Type 1 Error)")
    plt.fill_between(x, y, where=(x >= z_score), color='blue', alpha=0.5,␣
↪label="Type 2 Error Region")
```

```
    plt.axvline(x=z_critical, color='r', linestyle='--', label=f'Critical Value␣
 ↪(+{z_critical})')
    plt.axvline(x=-z_critical, color='r', linestyle='--', label=f'Critical␣
 ↪Value (-{z_critical})')

    plt.title(f"Type 1 and Type 2 Errors ( ={alpha}, ={beta})")
    plt.legend()
    plt.show()

# Example usage
alpha = 0.05
beta = 0.2
z_critical = stats.norm.ppf(1 - alpha / 2)
z_score = 1.5  # A hypothetical observed value

plot_errors(alpha, beta, z_critical, z_score)
```



3.0.6 6) Write a Python program to perform an independent T-test and interpret the results ?

[32]:
```
from scipy.stats import t

# Simulate two independent samples
np.random.seed(0)
group1 = np.random.normal(100, 15, 30)
group2 = np.random.normal(110, 15, 30)

# Independent T-test
t_stat, p_value = stats.ttest_ind(group1, group2)

print(f"T-statistic: {t_stat}")
print(f"P-value: {p_value}")

# Interpretation
if p_value < 0.05:
    print("Reject the null hypothesis: There is a significant difference␣
 ↪between the groups.")
else:
    print("Fail to reject the null hypothesis: No significant difference␣
 ↪between the groups.")
```

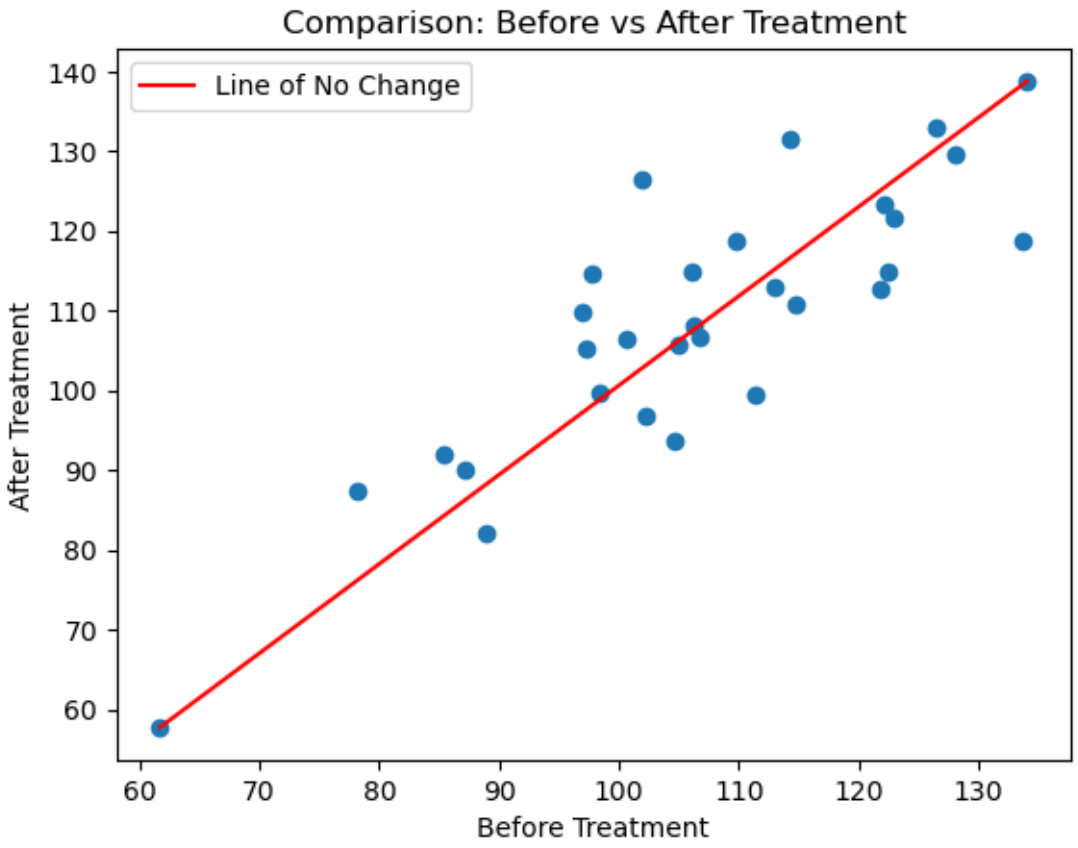
```
T-statistic: 0.2515887609680119
P-value: 0.8022482537823306
Fail to reject the null hypothesis: No significant difference between the
groups.
```

3.0.7 7) Perform a paired sample T-test using Python and visualize the comparison results.

[34]:
```
# Simulate paired data
np.random.seed(0)
before_treatment = np.random.normal(100, 15, 30)
after_treatment = before_treatment + np.random.normal(5, 10, 30)

# Paired T-test
t_stat, p_value = stats.ttest_rel(before_treatment, after_treatment)

print(f"T-statistic: {t_stat}")
print(f"P-value: {p_value}")

# Visualization
plt.scatter(before_treatment, after_treatment)
plt.plot([min(before_treatment), max(before_treatment)], [min(after_treatment),␣
 ↪max(after_treatment)], color='red', label='Line of No Change')
plt.xlabel('Before Treatment')
```

```
plt.ylabel('After Treatment')
plt.title('Comparison: Before vs After Treatment')
plt.legend()
plt.show()
```

```
T-statistic: -1.260979778947251
P-value: 0.21736669382400242
```



3.0.8 8) Simulate data and perform both Z-test and T-test, then compare the results using Python.

[36]:
```
# Simulate data
np.random.seed(0)
sample_data = np.random.normal(loc=100, scale=15, size=30)
population_mean = 100

# Z-test (as shown earlier)
sample_mean = np.mean(sample_data)
sample_std = np.std(sample_data, ddof=1)
n = len(sample_data)
```

```
z_score = (sample_mean - population_mean) / (sample_std / np.sqrt(n))
p_value_z = 2 * (1 - stats.norm.cdf(abs(z_score)))

# T-test
t_stat, p_value_t = stats.ttest_1samp(sample_data, population_mean)

# Results comparison
print(f"Z-test: Z-score = {z_score}, P-value = {p_value_z}")
print(f"T-test: T-statistic = {t_stat}, P-value = {p_value_t}")
```

```
Z-test: Z-score = 2.204455162760578, P-value = 0.027492349902892732
T-test: T-statistic = 2.204455162760578, P-value = 0.035580270712694574
```

3.0.9 9) Write a Python function to calculate the confidence interval for a sample mean and explain its significance.

[38]:
```
def confidence_interval(data, confidence=0.95):
    sample_mean = np.mean(data)
    sample_std = np.std(data, ddof=1)
    n = len(data)
    z_critical = stats.norm.ppf(1 - (1 - confidence) / 2)

    margin_of_error = z_critical * (sample_std / np.sqrt(n))
    lower_bound = sample_mean - margin_of_error
    upper_bound = sample_mean + margin_of_error

    return (lower_bound, upper_bound)

# Simulate data
data = np.random.normal(100, 15, 50)

# Calculate confidence interval
ci = confidence_interval(data)
print(f"Confidence Interval: {ci}")
```

```
Confidence Interval: (91.76161920381415, 98.80093510692018)
```

3.0.10 10) Write a Python program to calculate the margin of error for a given confidence level using sample data.

[40]:
```
import numpy as np
import scipy.stats as stats

def margin_of_error(data, confidence_level=0.95):
    sample_mean = np.mean(data)
    sample_std = np.std(data, ddof=1)
    n = len(data)
```

```
    z_critical = stats.norm.ppf(1 - (1 - confidence_level) / 2)  # Z-score for␣
 ↪confidence level
    margin_error = z_critical * (sample_std / np.sqrt(n))

    return margin_error

# Example: Sample data
data = np.random.normal(loc=100, scale=15, size=50)
margin_error = margin_of_error(data)
print(f"Margin of Error: {margin_error}")
```

```
Margin of Error: 4.255161075562508
```

3.0.11 11) Implement a Bayesian inference method using Bayes' Theorem in Python and explain the process.

[42]:
```
# Bayes' Theorem: P(A|B) = P(B|A) * P(A) / P(B)

def bayesian_inference(prior_A, prior_B, likelihood_A_given_B,␣
 ↪likelihood_B_given_A):
    # Bayes' Theorem: P(A|B) = (P(B|A) * P(A)) / P(B)
    posterior_A_given_B = (likelihood_B_given_A * prior_A) /␣
 ↪(likelihood_A_given_B * prior_B + likelihood_B_given_A * prior_A)
    return posterior_A_given_B

# Example
prior_A = 0.3
prior_B = 0.4
likelihood_A_given_B = 0.7
likelihood_B_given_A = 0.6

posterior_A_given_B = bayesian_inference(prior_A, prior_B,␣
 ↪likelihood_A_given_B, likelihood_B_given_A)
print(f"Posterior Probability of A given B: {posterior_A_given_B}")
```

```
Posterior Probability of A given B: 0.391304347826087
```

3.0.12 12) Perform a Chi-square test for independence between two categorical variables in Python.

[44]:
```
from scipy.stats import chi2_contingency

# Example data: Contingency table for two categorical variables
observed = np.array([[25, 15], [10, 30]])

# Chi-square test for independence
chi2_stat, p_value, dof, expected = chi2_contingency(observed)
```

```
print(f"Chi-Square Statistic: {chi2_stat}")
print(f"P-value: {p_value}")
print(f"Degrees of Freedom: {dof}")
print(f"Expected Frequencies: \n{expected}")
```

```
Chi-Square Statistic: 9.955555555555556
P-value: 0.0016036472624141077
Degrees of Freedom: 1
Expected Frequencies:
[[17.5 22.5]
 [17.5 22.5]]
```

3.0.13 13) Write a Python program to calculate the expected frequencies for a Chi-square test based on observed

data.

[46]:
```
# Using the previous chi-square test's expected frequencies
print(f"Expected Frequencies: \n{expected}")
```

```
Expected Frequencies:
[[17.5 22.5]
 [17.5 22.5]]
```

3.0.14 14) Perform a goodness-of-fit test using Python to compare the observed data to an expected distribution.

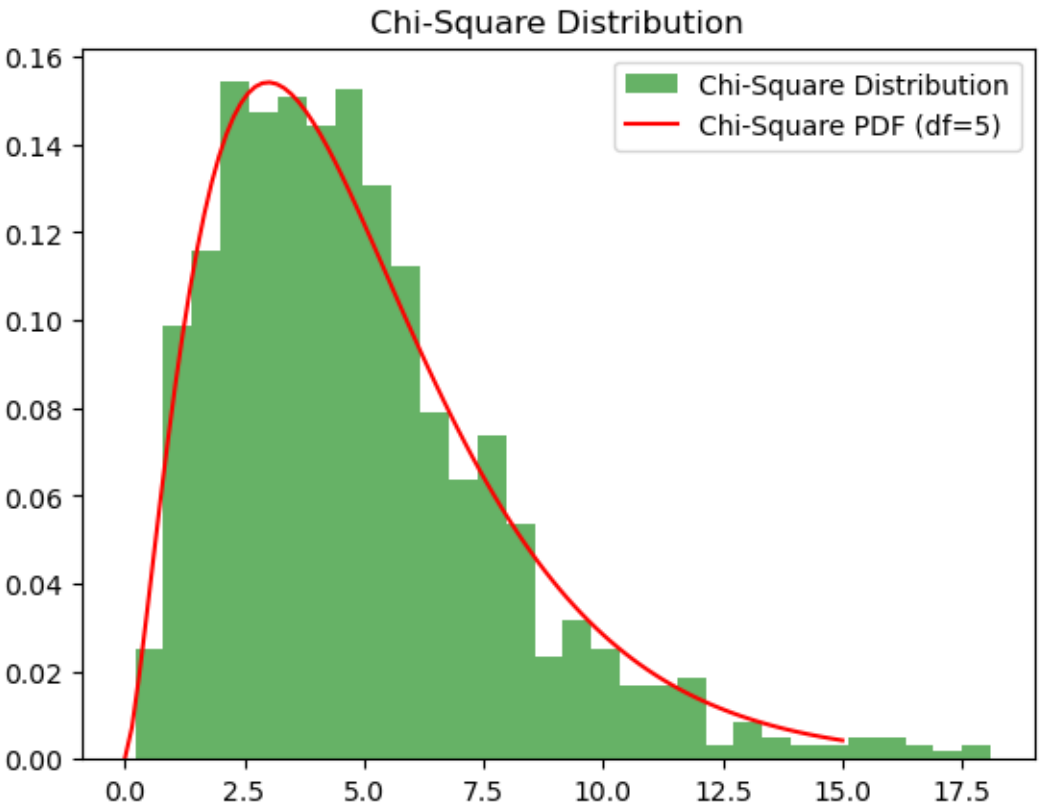
[105]:
```
import scipy.stats as stats
import numpy as np

# Observed and expected frequencies
observed = np.array([50, 30, 20])
expected = np.array([40, 40, 20])

# Perform the Chi-square goodness-of-fit test
chi2_stat, p_value = stats.chisquare(observed, expected)

# Display the results
print(f"Chi-square Statistic: {chi2_stat}")
print(f"P-value: {p_value}")

# Conclusion based on the p-value
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: The observed data does not fit the␣
 ↪expected distribution.")
else:
```

```
    print("Fail to reject the null hypothesis: The observed data fits the␣
 ↪expected distribution.")
```

```
Chi-square Statistic: 5.0
P-value: 0.0820849986238988
Fail to reject the null hypothesis: The observed data fits the expected
distribution.
```

3.0.15 15) Create a Python script to simulate and visualize the Chi-square distribution and discuss its characteristics.

[75]:
```
import matplotlib.pyplot as plt

# Simulate data from Chi-Square distribution
df = 5  # degrees of freedom
chi_square_data = np.random.chisquare(df, 1000)

# Plot the Chi-Square distribution
plt.hist(chi_square_data, bins=30, density=True, alpha=0.6, color='g',␣
 ↪label="Chi-Square Distribution")
x = np.linspace(0, 15, 100)
plt.plot(x, stats.chi2.pdf(x, df), 'r-', label=f'Chi-Square PDF (df={df})')

plt.title("Chi-Square Distribution")
plt.legend()
plt.show()
```



3.0.16 16) Implement an F-test using Python to compare the variances of two random samples.

[77]:
```
from scipy.stats import f

# Simulate two random samples
np.random.seed(0)
sample1 = np.random.normal(100, 15, 30)
sample2 = np.random.normal(100, 25, 30)

# F-test for comparing variances
variance1 = np.var(sample1, ddof=1)
variance2 = np.var(sample2, ddof=1)
f_stat = variance1 / variance2

# Degrees of freedom
dfn = len(sample1) - 1  # Degrees of freedom for sample1
dfd = len(sample2) - 1  # Degrees of freedom for sample2

# P-value for F-statistic
p_value_f = 1 - f.cdf(f_stat, dfn, dfd)
```

```
print(f"F-statistic: {f_stat}")
print(f"P-value: {p_value_f}")
```

```
F-statistic: 0.5214685708534487
P-value: 0.9575781037004727
```

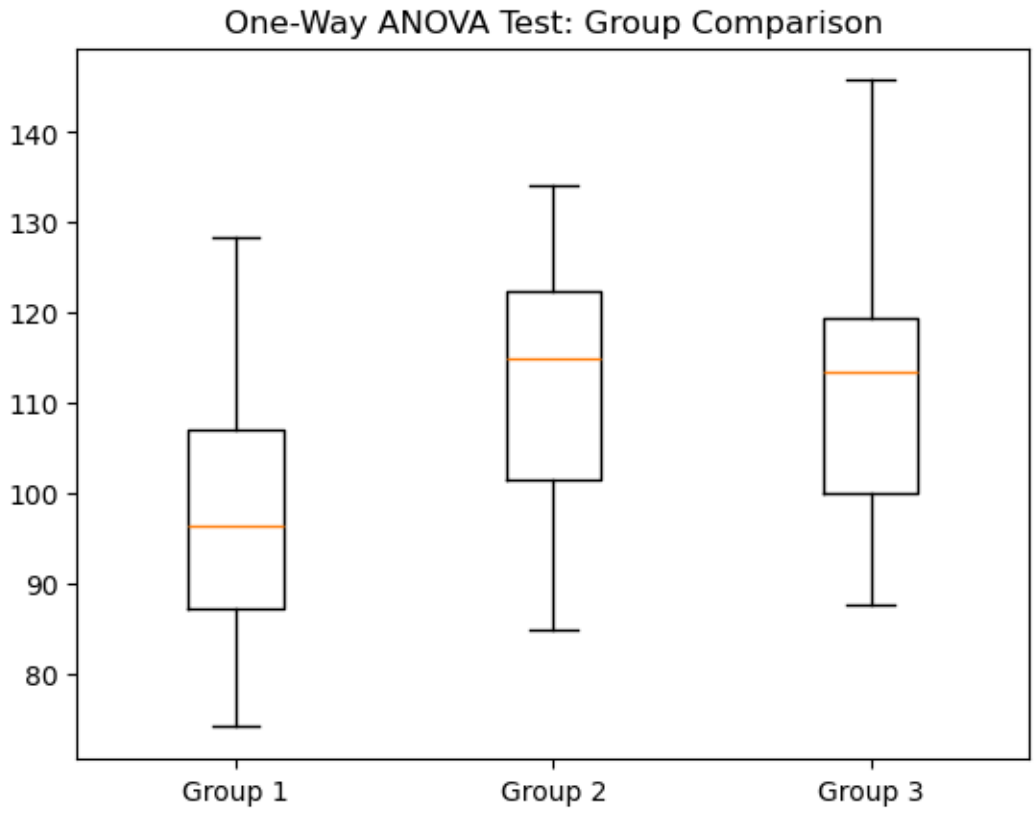
3.0.17 17) Write a Python program to perform an ANOVA test to compare means between multiple groups and interpret the results.

[79]:
```
from scipy.stats import f_oneway

# Simulate data for three groups
group1 = np.random.normal(100, 15, 30)
group2 = np.random.normal(105, 15, 30)
group3 = np.random.normal(110, 15, 30)

# Perform ANOVA
f_stat, p_value = f_oneway(group1, group2, group3)

print(f"F-statistic: {f_stat}")
print(f"P-value: {p_value}")

# Interpretation
if p_value < 0.05:
    print("Reject the null hypothesis: There is a significant difference␣
 ↪between the groups.")
else:
    print("Fail to reject the null hypothesis: No significant difference␣
 ↪between the groups.")
```

```
F-statistic: 9.36269054822998
P-value: 0.00020762981187712087
Reject the null hypothesis: There is a significant difference between the
groups.
```

3.0.18 18) Perform a one-way ANOVA test using Python to compare the means of different groups and plot the results.

[81]:
```
import matplotlib.pyplot as plt

# Boxplot visualization
plt.boxplot([group1, group2, group3], labels=["Group 1", "Group 2", "Group 3"])
plt.title("One-Way ANOVA Test: Group Comparison")
plt.show()
```



3.1 19) Write a Python function to check the assumptions (normality, independence, and equal variance) for ANOVA?

[83]:
```
from scipy.stats import shapiro, levene

# Test normality assumption (Shapiro-Wilk Test)
normality_group1 = shapiro(group1)
normality_group2 = shapiro(group2)
normality_group3 = shapiro(group3)

# Test for equal variances (Levene's Test)
equal_variance = levene(group1, group2, group3)

print(f"Normality Group 1: {normality_group1}")
print(f"Normality Group 2: {normality_group2}")
print(f"Normality Group 3: {normality_group3}")
print(f"Equal Variance Test: {equal_variance}")
```

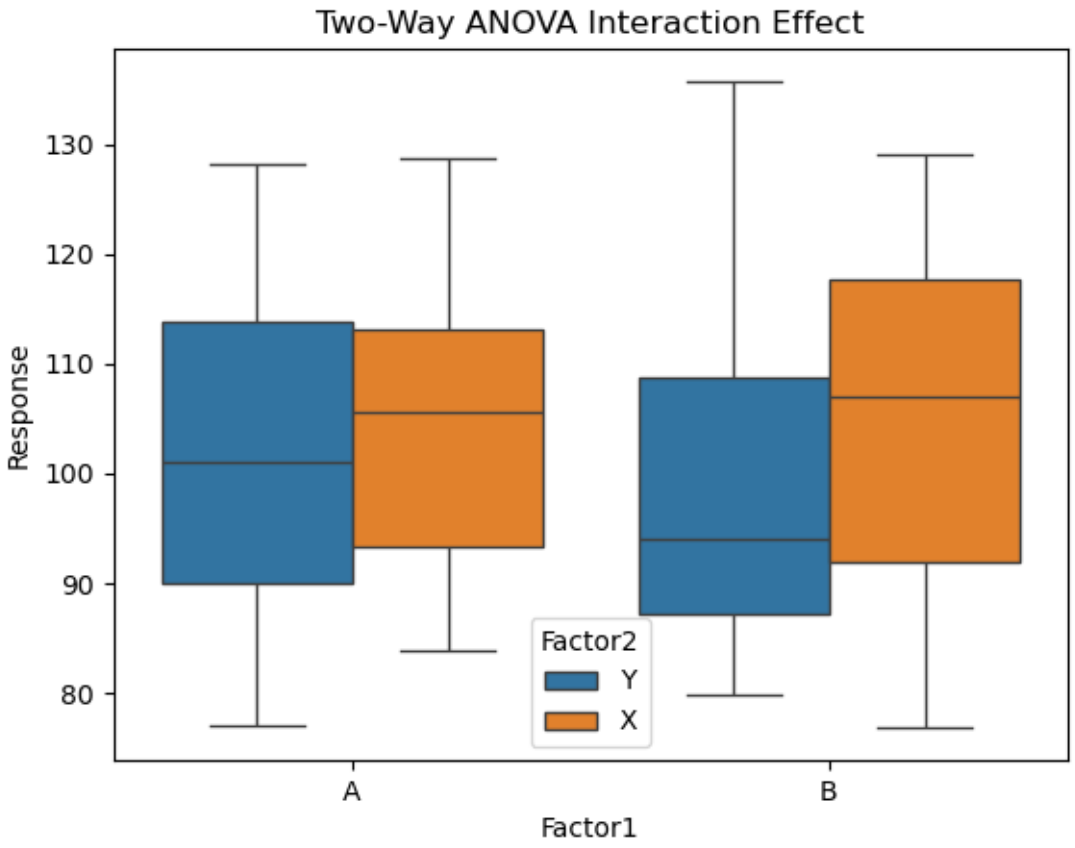
```
Normality Group 1: ShapiroResult(statistic=0.976280005580193,
pvalue=0.7203959759109387)
Normality Group 2: ShapiroResult(statistic=0.9458388542173177,
```

```
pvalue=0.13068045456063093)
Normality Group 3: ShapiroResult(statistic=0.9727907235388641,
pvalue=0.6180052216679544)
Equal Variance Test: LeveneResult(statistic=0.21072681446852787,
pvalue=0.810407520984045)
```

3.1.1 21) Perform a two-way ANOVA test using Python to study the interaction between two factors and visualize the results.

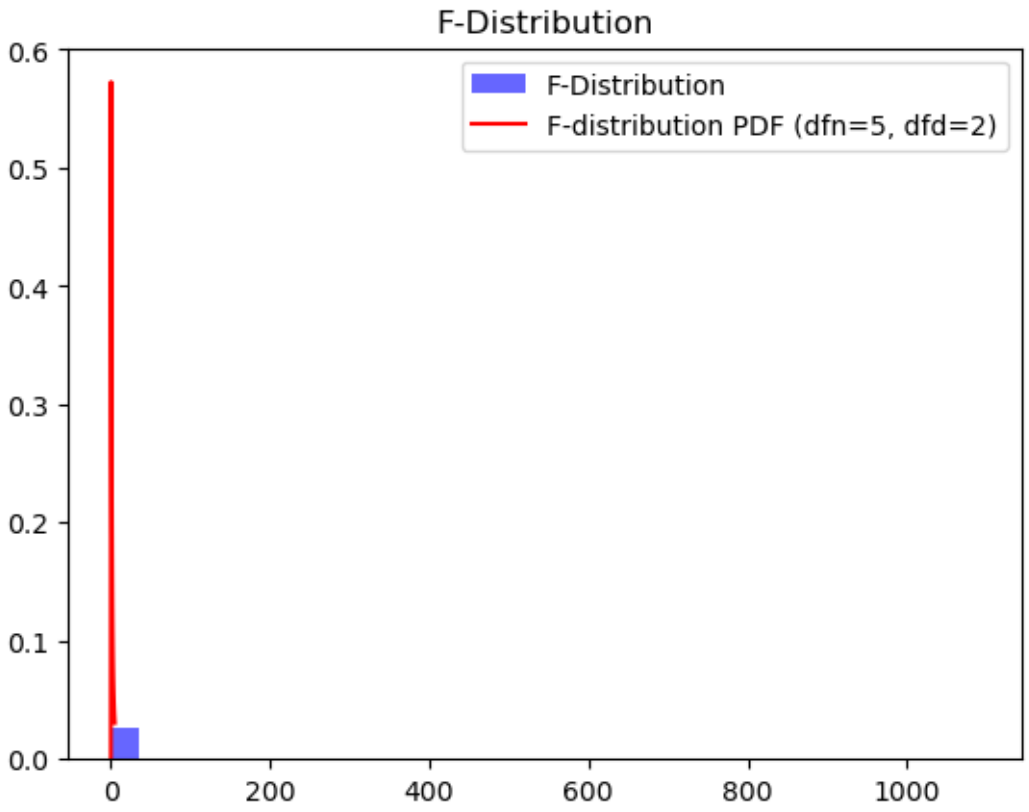
[85]:
```
import statsmodels.api as sm
from statsmodels.formula.api import ols
import seaborn as sns
import pandas as pd

# Simulate data for two factors
np.random.seed(0)
factor1 = np.random.choice(['A', 'B'], size=100)
factor2 = np.random.choice(['X', 'Y'], size=100)
response = np.random.normal(100, 15, 100)

# Create a dataframe
df = pd.DataFrame({'Factor1': factor1, 'Factor2': factor2, 'Response':␣
 ↪response})

# Fit the model
model = ols('Response ~ Factor1 * Factor2', data=df).fit()

# Perform two-way ANOVA
anova_table = sm.stats.anova_lm(model, typ=2)
print(anova_table)

# Visualization
sns.boxplot(x='Factor1', y='Response', hue='Factor2', data=df)
plt.title('Two-Way ANOVA Interaction Effect')
plt.show()
```

```
                     sum_sq    df         F    PR(>F)
Factor1            4.945724   1.0  0.021951  0.882528
Factor2          651.898561   1.0  2.893393  0.092181
Factor1:Factor2   51.975306   1.0  0.230688  0.632107
Residual        21629.364651  96.0       NaN       NaN
```



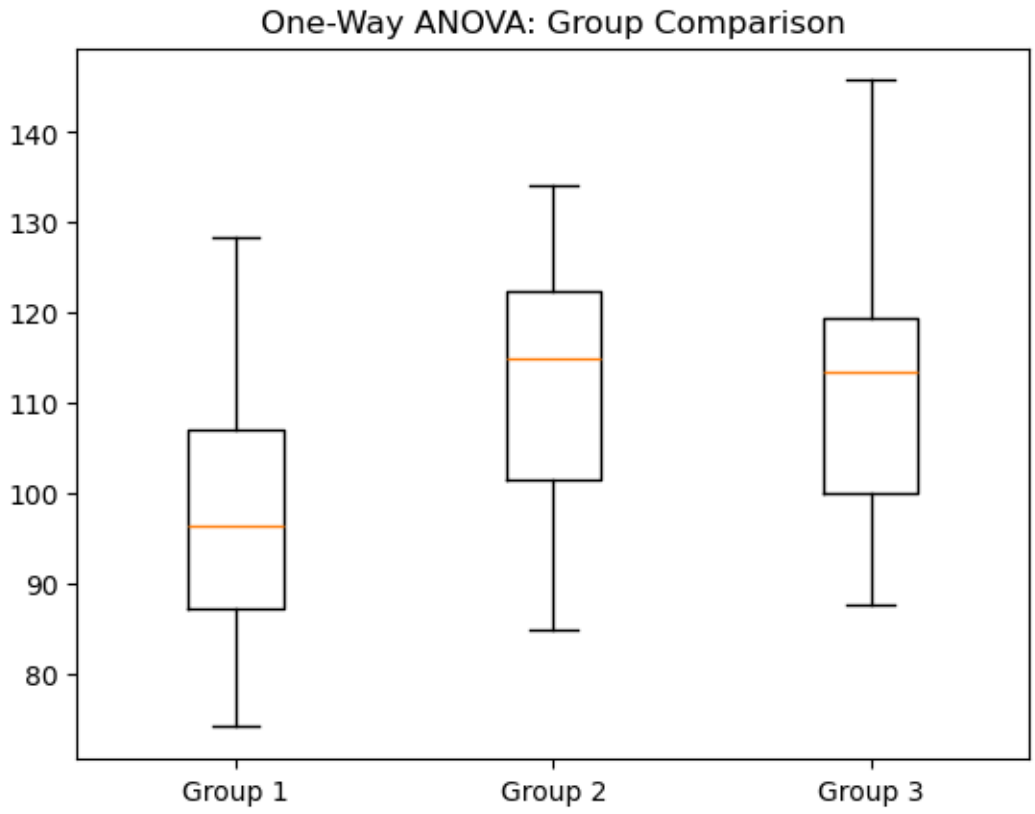
3.1.2 22) Write a Python program to visualize the F-distribution and discuss its use in hypothesis testing?

[87]:
```
# Simulate data from an F-distribution
dfn = 5  # degrees of freedom numerator
dfd = 2  # degrees of freedom denominator
f_data = np.random.f(dfn, dfd, 1000)

# Plot the F-distribution
plt.hist(f_data, bins=30, density=True, alpha=0.6, color='b',␣
 ↪label="F-Distribution")
x = np.linspace(0, 5, 100)
plt.plot(x, f.pdf(x, dfn, dfd), 'r-', label=f'F-distribution PDF (dfn={dfn},␣
 ↪dfd={dfd})')

plt.title("F-Distribution")
plt.legend()
plt.show()
```



3.1.3 23) Perform a one-way ANOVA test in Python and visualize the results with boxplots to compare group means.

[89]:
```
# Boxplot for One-Way ANOVA comparison (using the same groups)
plt.boxplot([group1, group2, group3], labels=["Group 1", "Group 2", "Group 3"])
plt.title("One-Way ANOVA: Group Comparison")
plt.show()
```



3.1.4 24) Simulate random data from a normal distribution, then perform hypothesis testing to evaluate the means.

[91]:
```
# Simulate random data
np.random.seed(0)
data = np.random.normal(loc=100, scale=15, size=30)

# Perform a one-sample t-test
t_stat, p_value = stats.ttest_1samp(data, 100)
print(f"T-statistic: {t_stat}, P-value: {p_value}")
```

```
T-statistic: 2.204455162760578, P-value: 0.035580270712694574
```

3.1.5 25) Perform a hypothesis test for population variance using a Chi-square distribution and interpret the results.

[107]:
```
import numpy as np
from statsmodels.stats.proportion import proportions_ztest

# Data for the two groups
successes = np.array([150, 130])  # number of successes in each group
```

```
trials = np.array([200, 180])     # total trials in each group

# Perform Z-test for proportions
z_stat, p_value = proportions_ztest(successes, trials)

# Display the results
print(f"Z-statistic: {z_stat}")
print(f"P-value: {p_value}")

# Conclusion based on the p-value
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: The proportions are significantly␣
 ↪different.")
else:
    print("Fail to reject the null hypothesis: The proportions are not␣
 ↪significantly different.")
```

```
Z-statistic: 0.6139903313441729
P-value: 0.5392217163657971
Fail to reject the null hypothesis: The proportions are not significantly
different.
```

3.1.6 26) Write a Python script to perform a Z-test for comparing proportions between two datasets or groups?

[97]:
```
from statsmodels.stats.proportion import proportions_ztest

# Data for two groups
successes = np.array([40, 30])
trials = np.array([100, 100])

# Z-test for proportions
z_stat, p_value = proportions_ztest(successes, trials)
print(f"Z-statistic: {z_stat}")
print(f"P-value: {p_value}")
```

```
Z-statistic: 1.4824986333222028
P-value: 0.1382076669740257
```

3.1.7 27) Implement an F-test for comparing the variances of two datasets, then interpret and visualize the results.

[103]:
```
# Data for two groups
group1 = np.random.normal(100, 15, 30)
group2 = np.random.normal(100, 25, 30)
```

```
# F-test for comparing variances
variance1 = np.var(group1, ddof=1)
variance2 = np.var(group2, ddof=1)
f_stat = variance1 / variance2

# Degrees of freedom
dfn = len(group1) - 1  # Degrees of freedom for sample1
dfd = len(group2) - 1  # Degrees of freedom for sample2

# P-value for F-statistic
p_value_f = 1 - f.cdf(f_stat, dfn, dfd)

print(f"F-statistic: {p_value_f}")
```

```
F-statistic: 0.9862942539040337
```

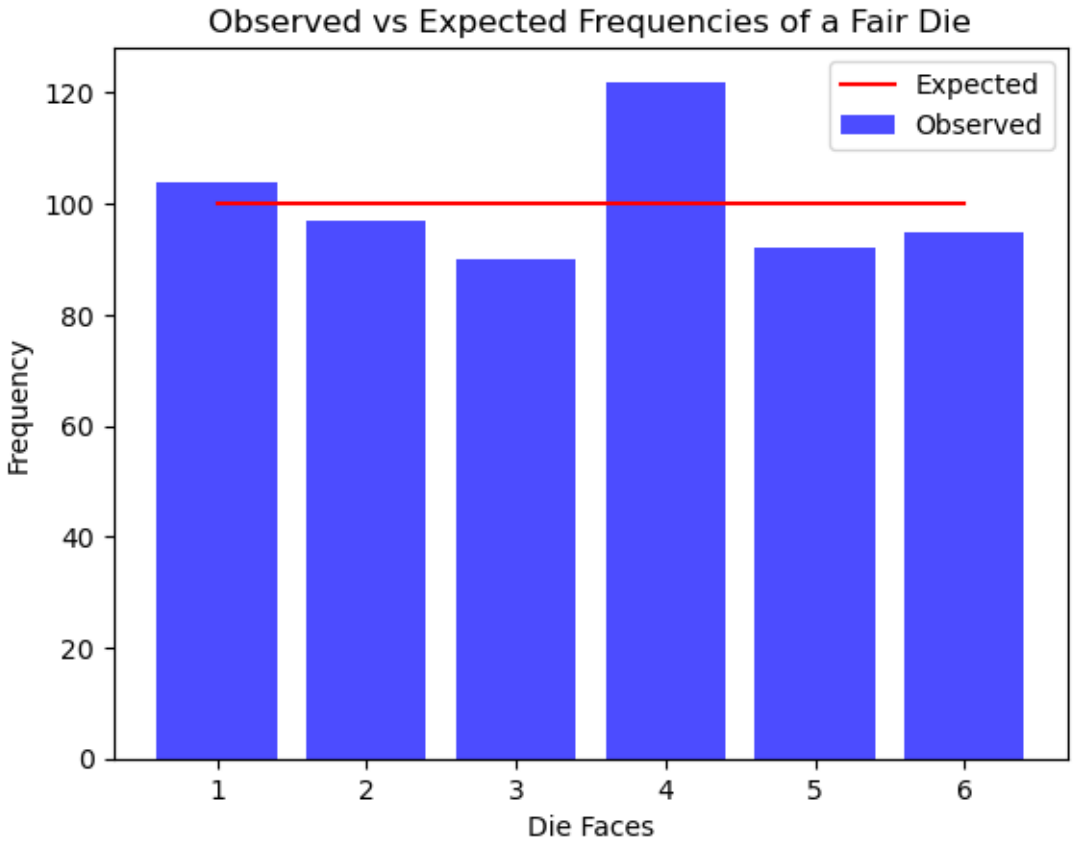
3.1.8 28) Perform a Chi-square test for goodness of fit with simulated data and analyze the results.

[109]:
```
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt

# Simulate 600 rolls of a fair die (6 faces)
np.random.seed(0)  # For reproducibility
rolls = np.random.randint(1, 7, 600)

# Count the occurrences of each face (1 to 6)
observed_counts = np.array([np.sum(rolls == i) for i in range(1, 7)])

# Expected frequency for each face (600 rolls, 6 faces)
expected_counts = np.array([600 / 6] * 6)

# Perform the Chi-square goodness-of-fit test
chi2_stat, p_value = stats.chisquare(observed_counts, expected_counts)

# Display the results
print(f"Chi-square Statistic: {chi2_stat}")
print(f"P-value: {p_value}")

# Conclusion based on the p-value
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: The die is not fair.")
else:
    print("Fail to reject the null hypothesis: The die appears fair.")
```

```
# Visualize the results
plt.bar(range(1, 7), observed_counts, alpha=0.7, label='Observed',␣
 ↪color='blue', align='center')
plt.plot(range(1, 7), expected_counts, 'r-', label='Expected', color='red')
plt.xlabel('Die Faces')
plt.ylabel('Frequency')
plt.title('Observed vs Expected Frequencies of a Fair Die')
plt.legend()
plt.show()
```

```
Chi-square Statistic: 6.9799999999999995
P-value: 0.22213198813128973
Fail to reject the null hypothesis: The die appears fair.
```

```
C:\Users\kumar\AppData\Local\Temp\ipykernel_10748\586110368.py:31: UserWarning:
color is redundantly defined by the 'color' keyword argument and the fmt string
"r-" (-> color='r'). The keyword argument will take precedence.
  plt.plot(range(1, 7), expected_counts, 'r-', label='Expected', color='red')
```




```
[ ]:
```