STATISTIC ASSIGNMENT

1. Discrete Random Variable Class

Class Definition and Methods

```
python
Copy code
class DiscreteRandomVariable:
    def __init__(self, values, probabilities):
        self.values = values
        self.probabilities = probabilities

    def expected_value(self):
        return sum(v * p for v, p in zip(self.values, self.probabilities))

    def variance(self):
        mean = self.expected_value()
        return sum(p * (v - mean) ** 2 for v, p in zip(self.values,
self.probabilities))

# Example usage
values = [1, 2, 3, 4, 5, 6]
probabilities = [1/6] * 6
die = DiscreteRandomVariable(values, probabilities)

print("Expected Value:", die.expected_value())
print("Variance:", die.variance())
```

2. Simulate Rolling a Die and Calculate Expected Value and Variance

```
python
Copy code
import numpy as np

# Simulate rolling a die 1000 times
rolls = np.random.randint(1, 7, size=1000)
expected_value = np.mean(rolls)
variance = np.var(rolls)

print("Expected Value:", expected_value)
print("Variance:", variance)
```

3. Generate Random Samples from a Distribution and Calculate Mean and Variance

Binomial Distribution

```
python
Copy code
```

```
def binomial_distribution(n, p, size):
    samples = np.random.binomial(n, p, size)
    mean = np.mean(samples)
    variance = np.var(samples)
    return mean, variance

mean, variance = binomial_distribution(10, 0.5, 1000)
print("Binomial Distribution Mean:", mean)
print("Binomial Distribution Variance:", variance)
```

Poisson Distribution

```
python
Copy code
def poisson_distribution(lam, size):
    samples = np.random.poisson(lam, size)
    mean = np.mean(samples)
    variance = np.var(samples)
    return mean, variance

mean, variance = poisson_distribution(5, 1000)
print("Poisson Distribution Mean:", mean)
print("Poisson Distribution Variance:", variance)
```

4. Generate Random Numbers from Gaussian Distribution and Compute Statistics

```
python
Copy code
samples = np.random.normal(loc=0, scale=1, size=1000)
mean = np.mean(samples)
variance = np.var(samples)
std_dev = np.std(samples)

print("Gaussian Distribution Mean:", mean)
print("Gaussian Distribution Variance:", variance)
print("Gaussian Distribution Standard Deviation:", std_dev)
```

5. Analysis of `tips` Dataset using Seaborn

```
python
Copy code
import seaborn as sns
import pandas as pd

# Load dataset
tips = sns.load_dataset("tips")

# Define functions for calculations
def calculate_skewness(series):
    return series.skew()

def covariance(series1, series2):
    return series1.cov(series2)
```

```
def pearson_correlation(series1, series2):
    return series1.corr(series2)

# Calculate skewness
total_bill_skewness = calculate_skewness(tips['total_bill'])
tip_skewness = calculate_skewness(tips['tip'])

print("Total Bill Skewness:", total_bill_skewness)
print("Tip Skewness:", tip_skewness)

# Determine skewness type
def skewness_type(skewness):
    if skewness > 0:
        return "Positive Skewness"
    elif skewness < 0:
        return "Negative Skewness"
    else:
        return "Approximately Symmetric"

print("Total Bill Skewness Type:", skewness_type(total_bill_skewness))
print("Tip Skewness Type:", skewness_type(tip_skewness))

# Calculate covariance and correlation
cov = covariance(tips['total_bill'], tips['tip'])
corr = pearson_correlation(tips['total_bill'], tips['tip'])

print("Covariance:", cov)
print("Pearson Correlation Coefficient:", corr)

# Visualize correlation
sns.scatterplot(x='total_bill', y='tip', data=tips)
plt.title('Scatter Plot of Total Bill vs Tip')
plt.show()
```

6. Probability Density Function of Normal Distribution

```
python
Copy code
from scipy.stats import norm

def pdf_normal(x, mean, std_dev):
    return norm.pdf(x, mean, std_dev)

print("PDF of Normal Distribution:", pdf_normal(0, 0, 1))
```

7. Cumulative Distribution Function of Exponential Distribution

```
python
Copy code
from scipy.stats import expon

def cdf_exponential(x, scale):
    return expon.cdf(x, scale=scale)
```

```
print("CDF of Exponential Distribution:", cdf_exponential(1, 1))
```

8. Probability Mass Function of Poisson Distribution

```
python
Copy code
from scipy.stats import poisson

def pmf_poisson(k, lam):
    return poisson.pmf(k, lam)

print("PMF of Poisson Distribution:", pmf_poisson(3, 2))
```

9. Z-Test for Layout Conversion Rate

```
python
Copy code
from statsmodels.stats.proportion import proportions_ztest

old_layout = np.array([1] * 50 + [0] * 950)
new_layout = np.array([1] * 70 + [0] * 930)

count = np.array([old_layout.sum(), new_layout.sum()])
nobs = np.array([len(old_layout), len(new_layout)])

z_stat, p_val = proportions_ztest(count, nobs)
print("Z-Statistic:", z_stat)
print("P-Value:", p_val)
```

10. Z-Test for Tutoring Program Effectiveness

```
python
Copy code
from scipy.stats import ttest_rel

before_program = np.array([75, 80, 85, 70, 90, 78, 92, 88, 82, 87])
after_program = np.array([80, 85, 90, 80, 92, 80, 95, 90, 85, 88])

z_stat, p_val = ttest_rel(after_program, before_program)
print("Z-Statistic:", z_stat)
print("P-Value:", p_val)
```

11. Z-Test for Drug Effectiveness

```
python
Copy code
before_drug = np.array([145, 150, 140, 135, 155, 160, 152, 148, 130, 138])
after_drug = np.array([130, 140, 132, 128, 145, 148, 138, 136, 125, 130])

z_stat, p_val = ttest_rel(after_drug, before_drug)
print("Z-Statistic:", z_stat)
print("P-Value:", p_val)
```

12. Z-Test for Customer Service Response Time

```
python
Copy code
response_times = np.array([4.3, 3.8, 5.1, 4.9, 4.7, 4.2, 5.2, 4.5, 4.6, 4.4])

z_stat, p_val = ttest_1samp(response_times, 5)
print("Z-Statistic:", z_stat)
print("P-Value:", p_val)
```

13. A/B Test for Website Click-Through Rates

```
python
Copy code
def ab_test_analysis(data1, data2):
    t_stat, p_val = ttest_ind(data1, data2)
    dof = len(data1) + len(data2) - 2
    return t_stat, dof, p_val

layout_a_clicks = [28, 32, 33, 29, 31, 34, 30, 35, 36, 37]
layout_b_clicks = [40, 41, 38, 42, 39, 44, 43, 41, 45, 47]

t_stat, dof, p_val = ab_test_analysis(layout_a_clicks, layout_b_clicks)
print("T-Statistic:", t_stat)
print("Degrees of Freedom:", dof)
print("P-Value:", p_val)
```

14. T-Test for Drug Effectiveness on Cholesterol Levels

```
python
Copy code
existing_drug_levels = [180, 182, 175, 185, 178, 176, 172, 184, 179, 183]
new_drug_levels = [170, 172, 165, 168, 175, 173, 170, 178, 172, 176]

t_stat, p_val = ttest_ind(new_drug_levels, existing_drug_levels)
print("T-Statistic:", t_stat)
print("P-Value:", p_val)
```

15. T-Test for Educational Intervention Effectiveness

```
python
Copy code
pre_intervention_scores = [80, 85, 90, 75, 88, 82, 92, 78, 85, 87]
post_intervention_scores = [90, 92, 88, 92, 95, 91, 96, 93, 89, 93]

t_stat, p_val = ttest_rel(post_intervention_scores, pre_intervention_scores)
print("T-Statistic:", t_stat)
print("P-Value:", p_val)
```

16. T-Test for Gender-Based Salary Gap

```
python
Copy code
```

```
np.random.seed(0)  # For reproducibility
male_salaries = np.random.normal(loc=50000, scale=10000, size=20)
female_salaries = np.random.normal(loc=55000, scale=9000, size=20)

t_stat, p_val = ttest_ind(male_salaries, female_salaries)
print("T-Statistic:", t_stat)
print("P-Value:", p_val)
```

17. T-Test for Product Quality Comparison

```
python
Copy code
version1_scores = [85, 88, 82, 89, 87, 84, 90, 88, 85, 86, 91, 83, 87, 84,
89, 86, 84, 88, 85, 86, 89, 90, 87, 88, 85]
version2_scores = [80, 78, 83, 81, 79, 82, 76, 80, 78, 81, 77, 82, 80, 79,
82, 79, 80, 81, 79, 82, 79, 78, 80, 81, 82]

t_stat, p_val = ttest_ind(version1_scores, version2_scores)
print("T-Statistic:", t_stat)
print("P-Value:", p_val)
```

18. T-Test for Customer Satisfaction Scores

```
python
Copy code
branch_a_scores = [4, 5, 3, 4, 5, 4, 5, 3, 4, 4, 5, 4, 4, 3, 4, 5, 5, 4, 3,
4, 5, 4, 3, 5, 4, 4, 5, 3, 4, 5, 4]
branch_b_scores = [3, 4, 2, 3, 4, 3, 4, 2, 3, 3, 4, 3, 3, 2, 3, 4, 4, 3, 2,
3, 4, 3, 2, 4, 3, 3, 4, 2, 3, 4, 3]

t_stat, p_val = ttest_ind(branch_a_scores, branch_b_scores)
print("T-Statistic:", t_stat)
print("P-Value:", p_val)
```

19. Chi-Square Test for Voter Preferences

```
python
Copy code
from scipy.stats import chi2_contingency

age_groups = np.random.choice(['18-30', '31-50', '51+'], size=500)
voter_preferences = np.random.choice(['Candidate A', 'Candidate B'],
size=500)

contingency_table = pd.crosstab(age_groups, voter_preferences)
chi2, p_val, dof, expected = chi2_contingency(contingency_table)

print("Chi-Square Statistic:", chi2)
print("P-Value:", p_val)
print("Degrees of Freedom:", dof)
print("Expected Frequencies:", expected)
```

20. Chi-Square Test for Customer Satisfaction and Region

```
python
Copy code
data = np.array([[50, 30, 40, 20], [30, 40, 30, 50], [20, 30, 40, 30]])
chi2, p_val, dof, expected = chi2_contingency(data)

print("Chi-Square Statistic:", chi2)
print("P-Value:", p_val)
print("Degrees of Freedom:", dof)
print("Expected Frequencies:", expected)
```

21. Chi-Square Test for Job Performance Before and After Training

```
python
Copy code
data = np.array([[50, 30, 20], [30, 40, 30], [20, 30, 40]])
chi2, p_val, dof, expected = chi2_contingency(data)

print("Chi-Square Statistic:", chi2)
print("P-Value:", p_val)
print("Degrees of Freedom:", dof)
print("Expected Frequencies:", expected)
```

22. ANOVA Test for Customer Satisfaction Scores

```
python
Copy code
from scipy.stats import f_oneway

standard_scores = [80, 85, 90, 78, 88, 82, 92, 78, 85, 87]
premium_scores = [90, 92, 88, 92, 95, 91, 96, 93, 89, 93]
deluxe_scores = [95, 98, 92, 97, 96, 94, 98, 97, 92, 99]

f_stat, p_val = f_oneway(standard_scores, premium_scores, deluxe_scores)
print("F-Statistic:", f_stat)
print("P-Value:", p_val)
```

23. Chi-Square Test for Job Performance Before and After Training

The Chi-Square test can be used to determine if there is a significant association between the job performance levels before and after the training. We will use the given sample data for this test.

Sample Data

```
python
Copy code
import numpy as np
from scipy.stats import chi2_contingency

# Job performance levels before (rows) and after (columns) training
```

```
data = np.array([[50, 30, 20], [30, 40, 30], [20, 30, 40]])

# Perform Chi-Square test
chi2, p_val, dof, expected = chi2_contingency(data)

print("Chi-Square Statistic:", chi2)
print("P-Value:", p_val)
print("Degrees of Freedom:", dof)
print("Expected Frequencies:", expected)
```

24. ANOVA Test for Customer Satisfaction Scores

The ANOVA test (Analysis of Variance) can be used to determine if there is a significant difference in customer satisfaction scores among the three product versions: Standard, Premium, and Deluxe.

Sample Data

```
python
Copy code
from scipy.stats import f_oneway

# Customer satisfaction scores for each product version
standard_scores = [80, 85, 90, 78, 88, 82, 92, 78, 85, 87]
premium_scores = [90, 92, 88, 92, 95, 91, 96, 93, 89, 93]
deluxe_scores = [95, 98, 92, 97, 96, 94, 98, 97, 92, 99]

# Perform ANOVA test
f_stat, p_val = f_oneway(standard_scores, premium_scores, deluxe_scores)

print("F-Statistic:", f_stat)
print("P-Value:", p_val)
```

Explanation of Results

For the Chi-Square Test:

- **Chi-Square Statistic**: This tells us how much the observed frequencies deviate from the expected frequencies.
- **P-Value**: This tells us the probability of observing such a deviation (or more extreme) by chance if there is no association between the variables.
- **Degrees of Freedom (dof)**: This is calculated based on the number of categories in each variable.
- **Expected Frequencies**: These are the frequencies we would expect if there is no association between the variables.

For the ANOVA Test:

- **F-Statistic**: This tells us the ratio of the variance between the group means to the variance within the groups.

- **P-Value**: This tells us the probability of observing such a ratio (or more extreme) by chance if there is no difference between the group means.

Interpretation

- If the **P-Value** from the Chi-Square Test is less than the significance level (usually 0.05), we reject the null hypothesis and conclude that there is a significant association between job performance levels before and after the training.
- If the **P-Value** from the ANOVA Test is less than the significance level (usually 0.05), we reject the null hypothesis and conclude that there is a significant difference in customer satisfaction scores among the three product versions.