1155175123 Tsoi Ming Hon

IERG 3050 Assignment 3 **Due: 11 April 2025**

- Submit a single .pdf file containing all your answers to the Blackboard before the due date.
- Answer all questions.
- Type or write your work neatly.

---

1. We want to predict the weight (y) of whitefishes from their heights (h) and widths (w). Here are the data obtained from a fish market:

y	h	w
270	8.3804	4.2476
270	8.1454	4.2485
306	8.778	4.6816
540	10.744	6.562
800	11.7612	6.5736
1000	12.354	6.525

Find the plane $y = a_0 + a_1 h + a_2 w$ by multiple linear regression. What is the predicted weight of a whitefish of height 10 and width 5.5?

2. Apply Monte Carlo simulation with importance sampling to compute $\int_0^2 \frac{dx}{\sqrt{x}}$.

3. Generate 300 samples of $U(0, 1)$ by any programming language or software you like. Perform the uniformity test of these samples by chi-square goodness-of-fit test with 5 equal-length bins and at 5% significance level. Also show the histogram of the bins.

4. Generate 5 samples from a Student's t-distribution with 30 degrees of freedom, and then standardize them. Sort the values in ascending order, and let the values be the 0.1-, 0.3-, 0.5-, 0.7- and 0.9-quantiles. Plot the Q-Q plot of these values against the standard normal distribution. What is the conclusion of the Q-Q plot?

1. We want to predict the weight (y) of whitefishes from their heights (h) and widths (w). Here are the data obtained from a fish market:

y	h	w
270	8.3804	4.2476
270	8.1454	4.2485
306	8.778	4.6816
540	10.744	6.562
800	11.7612	6.5736
1000	12.354	6.525

Find the plane $y = a_0 + a_1 h + a_2 w$ by multiple linear regression. What is the predicted weight of a whitefish of height 10 and width 5.5?

q1.py > ...

```python
import pandas as pd

data = {
    'y': [270, 270, 306, 540, 800, 1000],
    'h': [8.3804, 8.1454, 8.778, 10.744, 11.7612, 12.354],
    'w': [4.2476, 4.2485, 4.6816, 6.562, 6.5736, 6.525]
}
df = pd.DataFrame(data)
from sklearn.linear_model import LinearRegression

X = df[['h', 'w']]
Y = df['y']

regr = LinearRegression()
regr.fit(X, Y)

print('Intercept:', regr.intercept_)
print('Coefficients:', regr.coef_)

new_h = 10
new_w = 5.5
predicted_y = regr.predict([[new_h, new_w]])
print('Predicted Weight:', predicted_y[0])
```

```
(.venv) lifehater@LifedeMacBook-Pro temp %
win-arm64/bundled/libs/debugpy/adapter/../..
Intercept: -1263.789283808669
Coefficients: [ 275.77358715 -177.31218183]
Predicted Weight: 518.7295875821146
```

2. Apply Monte Carlo simulation with importance sampling to compute $\int_0^2 \frac{dx}{\sqrt{x}}$.

For $I = \int_0^2 \frac{1}{\sqrt{x}}\,dx$

Exact analytical solution:

$$I = \int_0^2 x^{-\frac{1}{2}}\,dx = \left[2x^{\frac{1}{2}}\right]_0^2 = 2\sqrt{2}$$

For $f(x) = \frac{1}{\sqrt{x}} \Rightarrow$ we select $p(x) = \frac{1}{2\sqrt{x}}$ for $x \in (0,2]$

Importance sampling estimate: $I \approx \frac{1}{N}\sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)} = \frac{1}{N}\sum_{i=1}^{N} \frac{\frac{1}{\sqrt{x_i}}}{\frac{1}{2\sqrt{x_i}}} = \frac{1}{N}\sum_{i=1}^{N} 2 = 2$ $(\times)$

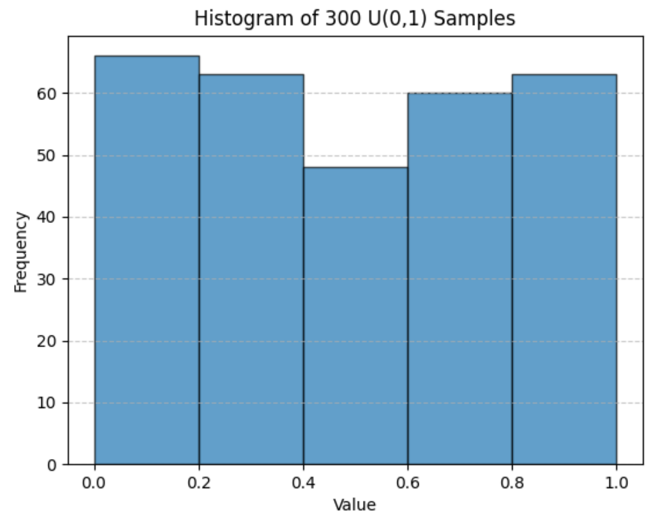
Correct normalized distribution: $p(x) = \frac{1}{2\sqrt{2}\,x}$

$$I \approx \frac{1}{N}\sum_{i=1}^{N} \frac{\frac{1}{\sqrt{x_i}}}{\frac{1}{2\sqrt{2}\sqrt{x_i}}} = \frac{1}{N}\sum_{i=1}^{N} 2\sqrt{2} = 2\sqrt{2} \;//$$

3. Generate 300 samples of $U(0,1)$ by any programming language or software you like. Perform the uniformity test of these samples by chi-square goodness-of-fit test with 5 equal-length bins and at 5% significance level. Also show the histogram of the bins.

q3.py > ...

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

# 1. Generate 300 samples from U(0, 1)
np.random.seed(42)  # For reproducibility
samples = np.random.uniform(0, 1, 300)

# 2. Define 5 equal-length bins between 0 and 1
bin_edges = np.linspace(0, 1, 6)  # 6 edges for 5 bins
observed_counts, _ = np.histogram(samples, bins=bin_edges)

# 3. Expected counts for each bin (uniform): 300 / 5 = 60
expected_counts = np.array([60] * 5)

# 4. Perform chi-square goodness-of-fit test
chi2_stat, p_value = chisquare(f_obs=observed_counts, f_exp=expected_counts)

# 5. Output results
print("Observed counts:", observed_counts)
print("Expected counts:", expected_counts)
print(f"Chi-square statistic: {chi2_stat:.4f}")
print(f"P-value: {p_value:.4f}")

if p_value < 0.05:
    print("Reject the null hypothesis: Data is not uniformly distributed.")
else:
    print("Fail to reject the null hypothesis: Data may be uniformly distributed.")

# 6. Plot histogram
plt.hist(samples, bins=bin_edges, edgecolor='black', alpha=0.7)
plt.title("Histogram of 300 U(0,1) Samples")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.xticks(bin_edges)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

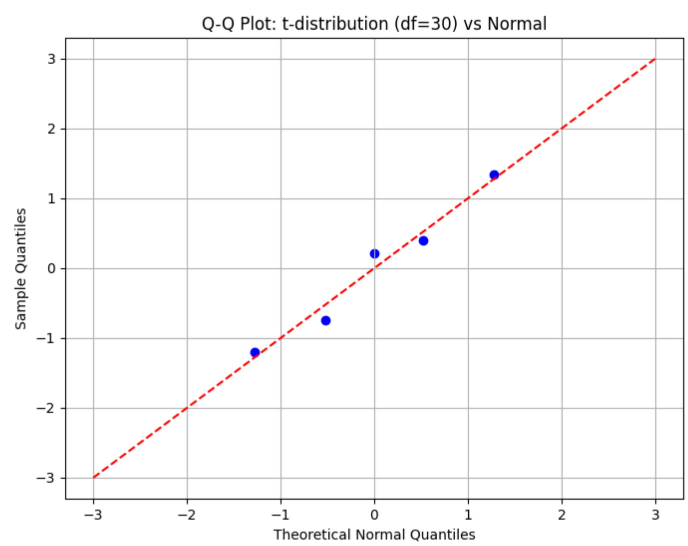



```
(.venv) lifehater@LifedeMacBook-Pro temp %
win-arm64/bundled/libs/debugpy/adapter/../
Observed counts: [66 63 48 60 63]
Expected counts: [60 60 60 60 60]
Chi-square statistic: 3.3000
P-value: 0.5089
```

4. Generate 5 samples from a Student's t-distribution with 30 degrees of freedom, and then standardize them. Sort the values in ascending order, and let the values be the 0.1-, 0.3-, 0.5-, 0.7- and 0.9-quantiles. Plot the Q-Q plot of these values against the standard normal distribution. What is the conclusion of the Q-Q plot?

q4.py > ...

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

# Set random seed for reproducibility
np.random.seed(42)

# 1. Generate 5 samples from t-distribution with 30 df
t_samples = stats.t.rvs(df=30, size=5)

# 2. Standardize the samples (though t with high df is already ~N(0,1))
standardized = (t_samples - np.mean(t_samples)) / np.std(t_samples, ddof=1)

# 3. Sort the values
sorted_samples = np.sort(standardized)

# 4. Define desired quantiles
quantiles = np.array([0.1, 0.3, 0.5, 0.7, 0.9])

# 5. Get theoretical normal quantiles
normal_quantiles = stats.norm.ppf(quantiles)

# 6. Create Q-Q plot
plt.figure(figsize=(8, 6))
plt.scatter(normal_quantiles, sorted_samples, color='blue')
plt.plot([-3, 3], [-3, 3], 'r--')  # y=x reference line
plt.title('Q-Q Plot: t-distribution (df=30) vs Normal')
plt.xlabel('Theoretical Normal Quantiles')
plt.ylabel('Sample Quantiles')
plt.grid(True)
plt.show()

# 7. Calculate correlation for additional insight
correlation = np.corrcoef(normal_quantiles, sorted_samples)[0, 1]
print(f"Correlation between theoretical and sample quantiles: {correlation:.4f}")
```




```
Correlation between theoretical and sample quantiles: 0.9850
```