Lab Assignment: Probability Distributions (Coding)

Q1. Write a Python program to simulate 10,000 random numbers from a Uniform distribution U(2,5). Plot the histogram of the simulated data and compare it with the theoretical probability density function of U(2,5).

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


np.random.seed(42)

n = 10000
a, b = 2, 5
samples = np.random.uniform(a, b, size=n)



plt.figure(figsize=(7,4))
plt.hist(samples, bins=30, density=True, alpha=0.6,
label='Simulated')


x = np.linspace(a-0.5, b+0.5, 500)
pdf = np.where((x>=a) & (x<=b), 1.0/(b-a), 0.0)
plt.plot(x, pdf, 'r-', lw=2, label='Theoretical U(2,5)')

plt.title('Question 1: Uniform(2,5)')
plt.xlabel('x')
plt.ylabel('Density')
plt.legend()
plt.grid(alpha=0.25)
plt.tight_layout()
plt.show()
```

Q2. The waiting time (in minutes) for a bus is uniformly distributed between 0 and 20 minutes. Write a Python program to simulate 10,000 such waiting times and address the following:

- Estimate the probability that waiting time exceeds 15 minutes.
- Estimate the probability that the waiting time lies between 5 and 15 minutes.
- Calculate mean and variance of the simulated data and compare with theoretical values.
- Plot histogram of simulated data.

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


np.random.seed(42)

n = 10000
a, b = 0, 20
samples = np.random.uniform(a, b, size=n)

p_gt_15_emp = np.mean(samples > 15)
p_between_5_15_emp = np.mean((samples >= 5) & (samples <= 15))
mean_emp = samples.mean()
var_emp = samples.var(ddof=0)

p_gt_15_th = (b - 15) / (b - a)
p_between_5_15_th = (15 - 5) / (b - a)
mean_th = (a + b) / 2
var_th = (b - a) ** 2 / 12

print({
    'P(T>15)': {'empirical': p_gt_15_emp, 'theoretical':
p_gt_15_th},
    'P(5<=T<=15)': {'empirical': p_between_5_15_emp, 'theoretical':
p_between_5_15_th},
    'mean': {'empirical': mean_emp, 'theoretical': mean_th},
    'variance': {'empirical': var_emp, 'theoretical': var_th},
})


plt.figure(figsize=(7,4))
plt.hist(samples, bins=30, density=True, alpha=0.7,
label='Simulated')

x = np.linspace(a-2, b+2, 400)
pdf = np.where((x>=a) & (x<=b), 1/(b-a), 0.0)
plt.plot(x, pdf, 'r-', lw=2, label='Theoretical  U(0,20)')

plt.title('Question 2: Waiting Time U(0,20)')
plt.xlabel('Minutes')
plt.ylabel('Density')
plt.legend()
plt.grid(alpha=0.25)
plt.tight_layout()
plt.show()
```

Q3. Write a Python program to generate 10,000 random values from a Normal distribution with mean 0 and standard deviation 1. Plot histogram of the generated values and overlay the theoretical density curve. Also, compute the percentage of values lying between -2 and 2 and compare with theoretical probability.

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


np.random.seed(42)

n = 10000
samples = np.random.normal(0, 1, size=n)

pct_emp = np.mean((samples >= -2) & (samples <= 2))
pct_th = norm.cdf(2) - norm.cdf(-2)

print({'pct_between_-2_2': {'empirical': pct_emp, 'theoretical':
pct_th}})

x = np.linspace(-4, 4, 600)
pdf = norm.pdf(x, 0, 1)

plt.figure(figsize=(7,4))
plt.hist(samples, bins=40, density=True, alpha=0.6,
label='Simulated')
plt.plot(x, pdf, 'r-', lw=2, label='N(0,1) PDF')
plt.title('Question 3: Standard Normal')
plt.xlabel('x')
plt.ylabel('Density')
plt.legend()
plt.grid(alpha=0.25)
plt.tight_layout()
plt.show()
```

Q4. In an intelligence test, IQ scores are modeled as N(100,15). Write a Python program that simulates IQ scores of 1,000 individuals and then:
- Calculate how many have IQ above 130.
- Calculate how many have IQ below 85.

- Compare counts with theoretical expectations.
- Plot histogram of simulated IQ scores.

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


np.random.seed(42)

n = 1000
mu, sigma = 100, 15
samples = np.random.normal(mu, sigma, size=n)

count_above_130 = int(np.sum(samples > 130))
count_below_85 = int(np.sum(samples < 85))

p_above_130 = 1 - norm.cdf(130, loc=mu, scale=sigma)
p_below_85 = norm.cdf(85, loc=mu, scale=sigma)
exp_above_130 = n * p_above_130
exp_below_85 = n * p_below_85

print({
    'count_above_130': count_above_130,
    'count_below_85': count_below_85,
    'expected_above_130': exp_above_130,
    'expected_below_85': exp_below_85,
    'p_above_130': p_above_130,
    'p_below_85': p_below_85
})

plt.figure(figsize=(7,4))
plt.hist(samples, bins=30, density=True, alpha=0.6,
label='Simulated')
x = np.linspace(mu-4*sigma, mu+4*sigma, 600)
pdf = norm.pdf(x, mu, sigma)
plt.plot(x, pdf, 'r-', lw=2, label='N(100,15) PDF')
plt.title('Question 4: IQ Distribution')
plt.xlabel('IQ')
plt.ylabel('Density')
plt.legend()
plt.grid(alpha=0.25)
plt.tight_layout()
plt.show()
```

Q5. Write a Python program to simulate 10,000 flips of a fair coin (Bernoulli trial, p=0.5). After each flip, compute the running proportion of heads and plot its evolution, showing convergence to 0.5 (Law of Large Numbers).

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


np.random.seed(42)

n = 10000
p = 0.5
flips = np.random.binomial(1, p, size=n)
running_prop = np.cumsum(flips) / (np.arange(n) + 1)

plt.figure(figsize=(7,4))
plt.plot(running_prop, label='Running proportion of heads')
plt.axhline(p, color='r', linestyle='--', label='Theoretical p=0.5')
plt.title('Question 5: Law of Large Numbers (Fair Coin)')
plt.xlabel('Trial')
plt.ylabel('Proportion')
plt.legend()
plt.grid(alpha=0.25)
plt.tight_layout()
plt.show()
```

Q6. Consider a biased coin with p=0.3. Write a Python program to simulate 1,000 independent flips and:
- Compute sample proportion of heads and sample variance.
- Compare with theoretical mean and variance.
- Plot bar chart of relative frequencies of outcomes (0s vs 1s).

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


np.random.seed(42)

n = 1000
p = 0.3
flips = np.random.binomial(1, p, size=n)
prop_emp = flips.mean()
```

```
var_emp = flips.var(ddof=0)

mean_th = p
var_th = p * (1 - p)

print({
    'sample_proportion_heads': prop_emp,
    'sample_variance': var_emp,
    'theoretical_mean': mean_th,
    'theoretical_variance': var_th
})

vals, counts = np.unique(flips, return_counts=True)
rel_freq = counts / n

plt.figure(figsize=(6,4))
plt.bar(vals, rel_freq, width=0.5, alpha=0.8, tick_label=['0','1'],
label='Relative freq')
plt.axhline(1-p, color='gray', linestyle='--', alpha=0.7,
label='Theoretical P(0)')
plt.axhline(p, color='r', linestyle='--', alpha=0.7,
label='Theoretical P(1)')
plt.title('Question 6: Biased Coin p=0.3')
plt.xlabel('Outcome')
plt.ylabel('Relative Frequency')
plt.legend()
plt.grid(alpha=0.25)
plt.tight_layout()
plt.show()
```

Q7. Write a Python program that simulates 10,000 repetitions of an experiment of 10 independent coin flips (p=0.5 each). Record number of heads per experiment. Plot histogram of outcomes (0–10 heads) and overlay theoretical Binomial(10,0.5) PMF.

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


np.random.seed(42)

n_experiments = 10000
n = 10
p = 0.5
```

```
heads_counts = np.random.binomial(n, p, size=n_experiments)

k = np.arange(0, n+1)
pmf = binom.pmf(k, n, p)

plt.figure(figsize=(7,4))
plt.hist(heads_counts, bins=np.arange(-0.5, n+1.5, 1), density=True,
alpha=0.6, label='Empirical')
plt.plot(k, pmf, 'ro-', label='Binomial(10,0.5) PMF')
plt.title('Question 7: Heads in 10 Fair Flips')
plt.xlabel('Number of Heads')
plt.ylabel('Probability')
plt.legend()
plt.grid(alpha=0.25)
plt.tight_layout()
plt.show()
```

Q8. A multiple-choice quiz has 10 questions with 4 options each (p=0.25 for correct guess). Simulate scores of 5,000 students guessing randomly. Estimate probability of scoring at least 5 and compare with theoretical Binomial(10,0.25). Plot bar chart of score distribution (0–10).

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


np.random.seed(42)

n_students = 5000
n = 10
p = 0.25

scores = np.random.binomial(n, p, size=n_students)

prob_emp = np.mean(scores >= 5)

k = np.arange(0, n+1)
pmf = binom.pmf(k, n, p)
prob_th = pmf[k>=5].sum()

print({
    'P(score>=5)_empirical': prob_emp,
```

```
    'P(score>=5)_theoretical': prob_th
})

plt.figure(figsize=(7,4))
bins = np.arange(-0.5, n+1.5, 1)
plt.hist(scores, bins=bins, density=True, alpha=0.7,
label='Empirical')
plt.plot(k, pmf, 'r-o', label='Binomial(10,0.25) PMF')
plt.title('Question 8: Quiz Score Distribution')
plt.xlabel('Score (0-10)')
plt.ylabel('Probability')
plt.legend()
plt.grid(alpha=0.25)
plt.tight_layout()
plt.show()
```