**Assignment Code: DA-AG-007**

# Statistics Advanced - 2| **Assignment**

**Instructions:** Carefully read each question. Use Google Docs, Microsoft Word, or a similar tool to create a document where you type out each question along with its answer. Save the document as a PDF, and then upload it to the LMS. Please do not zip or archive the files before uploading them. Each question carries 20 marks.

**Total Marks**: 180

**Question 1:** What is hypothesis testing in statistics?

**Answer:**

 It's a **statistical inference** procedure. You take a sample from a population, then use the data to test whether a specific claim (hypothesis) about the population parameter is plausible.
 Usually you define two competing hypotheses:

1. **Null hypothesis** (denoted H0H_0H0) — this is a default claim, often saying "no effect" or "no difference" or that a parameter equals some value.
2. **Alternative hypothesis** (denoted HaH_aHa or H1H_1H1) — what you suspect might be true instead of the null (e.g. there *is* a difference, or the parameter is greater/less than some value).

 Based on sample data, you compute a **test statistic**. This statistic has a known probability distribution under the assumption that H0H_0H0 is true. Then you see how extreme your observed statistic is relative to that distribution.
 You decide whether to *reject* H0H_0H0 in favor of HaH_aHa, or *fail to reject* H0H_0H0 (i.e. data do not give strong enough evidence against H0H_0H0). Note: failing to reject doesn't mean you accept the null as true; just that there isn't strong enough evidence to discard it.

**Question 2:** What is the null hypothesis, and how does it differ from the alternative hypothesis?

**Answer:**

| Aspect | Null Hypothesis H0H_0 | Alternative Hypothesis HaH_a / H1H_1 |
|---|---|---|
| What it asserts | No effect / no difference / status quo | Some effect / difference / what researcher wants to show |
| Relationship to equality / inequality | Usually involves equality (e.g. ==, ≥, ≤) | Usually involves inequality (≠, >, or <) unless a specific type of test is used (directional) |
| Assumption start point | Assumed true at the start of test | Not assumed true; you test whether data provide enough evidence in its favor |
| Outcome roles | You either “fail to reject” or “reject” H0H_0 based on data | You either support HaH_a (by rejecting H0H_0) or fail to support it (if H0H_0 is not rejected) |
| Error implications | Risk of **Type I error** if you incorrectly reject H0H_0 when it's true | Risk of **Type II error** if you fail to reject H0H_0 when HaH_a is actually true |

**Question 3:** Explain the significance level in hypothesis testing and its role in deciding the outcome of a test.

**Answer:**

 The **significance level**, symbolized by α (alpha), is a threshold set *before* collecting data. It’s the maximum probability the researcher is willing to accept of making a **Type I error** — that is, rejecting the null hypothesis H0H_0H0 *when in fact it’s true*.
 It defines how “extreme” the sample evidence must be in order to reject H0H_0H0. In other words, it determines how much risk of a false positive (thinking there is a real effect when there isn’t) the researcher accepts.

**Question 4:** What are Type I and Type II errors? Give examples of each.

**Answer:**

 **Type I error** (also called *false positive* or error of the first kind): rejecting the null hypothesis H0H_0H0 when it is actually true.
 **Type II error** (also called *false negative* or error of the second kind): failing to reject the null hypothesis when it is actually false.

**Question 5:** What is the difference between a Z-test and a T-test? Explain when to use each.

**Answer:**

| Feature | Z-test | T-test |
|---|---|---|
| Knowledge of population standard deviation (σ) | **Known** or treated as known. If it's unknown but sample is large, sometimes approximate. | **Unknown**; we estimate σ using sample standard deviation s. |
| Sample size | Usually **large** (typical rule of thumb n≥30n \ge 30, though context matters) ensures by the central limit theorem that sampling distribution approximates normal. | Often used when the sample is **small** (less than ~30) so that we need to account for the extra variability from estimating σ). |
| Distribution of test statistic | Standard normal (Z) distribution under H0H_0 | Student's t-distribution with appropriate degrees of freedom (often n−1n − 1) |
| Assumption about data | Data (or sample mean) approximately normally distributed (especially important when n is small); independent observations; in many cases known σ. | Needs normality (or approximate) of the underlying population especially with small n; independence; variance assumed unknown. |

**Question 6:** Write a Python program to generate a binomial distribution with n=10 and p=0.5, then plot its histogram.

(*Include your Python code and output in the code box below.*)

Hint: Generate random number using random function.

**Answer:**

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

def simulate_binomial(n, p, num_samples):
    """
    Generate `num_samples` samples from a Binomial(n, p) distribution.
    """
    # Use NumPy's random.binomial to do this efficiently
    samples = np.random.binomial(n, p, size=num_samples)
    return samples

def plot_histogram(samples, n, p):
    """
    Plot a histogram of the binomial samples and overlay the theoretical PMF.
    """
    # Compute counts for each possible number of successes (0 to n)
    possible_outcomes = np.arange(0, n+1)

    # Histogram of simulated data
    # Use bins aligned to integers so each possible count is its own bar
    bins = np.arange(-0.5, n + 1.5, 1)  # to center bins on integers
    plt.hist(samples, bins=bins, density=True, alpha=0.7, edgecolor='black',
             label='Simulated frequency')

    # Theoretical PMF
    # Probability Mass Function: P(X = k) for k = 0,1,...,n
    from scipy.stats import binom
    pmf_values = binom.pmf(possible_outcomes, n, p)

    # Overlay the PMF as points (and optionally lines)
    plt.scatter(possible_outcomes, pmf_values, color='red', label='Theoretical PMF')
    plt.vlines(possible_outcomes, 0, pmf_values, colors='red', linestyles='dashed', alpha=0.6)

    # Labels & title
    plt.xlabel('Number of successes')
    plt.ylabel('Probability')
    plt.title(f'Binomial Distribution (n={n}, p={p})')
    plt.legend()
    plt.grid(axis='y')
    plt.show()

def main():
    n = 10      # number of trials per sample
```

```
    p = 0.5     # probability of success in each trial
    num_samples = 10000  # how many independent binomial draws

    # Simulate
    samples = simulate_binomial(n, p, num_samples)

    # Plot
    plot_histogram(samples, n, p)

if __name__ == "__main__":
    main()
```

**Question 7**: Implement hypothesis testing using Z-statistics for a sample dataset in Python. Show the Python code and interpret the results.

sample_data = [49.1, 50.2, 51.0, 48.7, 50.5, 49.8, 50.3, 50.7, 50.2, 49.6,
50.1, 49.9, 50.8, 50.4, 48.9, 50.6, 50.0, 49.7, 50.2, 49.5,
50.1, 50.3, 50.4, 50.5, 50.0, 50.7, 49.3, 49.8, 50.2, 50.9,
50.3, 50.4, 50.0, 49.7, 50.5, 49.9]

(*Include your Python code and output in the code box below.*)

**Answer:**

```
 import numpy as np
from statsmodels.stats.weightstats import ztest
import scipy.stats as stats

# your sample data
sample_data = [49.1, 50.2, 51.0, 48.7, 50.5, 49.8, 50.3, 50.7, 50.2, 49.6,
               50.1, 49.9, 50.8, 50.4, 48.9, 50.6, 50.0, 49.7, 50.2, 49.5,
               50.1, 50.3, 50.4, 50.5, 50.0, 50.7, 49.3, 49.8, 50.2, 50.9,
               50.3, 50.4, 50.0, 49.7, 50.5, 49.9]

# Parameters for null hypothesis
mu0 = 50.0  # population mean under H0
sigma = 0.5 # assume known population standard deviation (just for illustration)

# Compute sample mean, sample size
x_bar = np.mean(sample_data)
n = len(sample_data)

# Compute the z-statistic manually
z_stat = (x_bar - mu0) / (sigma / np.sqrt(n))
# Two-tailed p-value from z
```

```
p_value = 2 * (1 - stats.norm.cdf(abs(z_stat)))

print("Sample mean:", x_bar)
print("Z-statistic:", z_stat)
print("P-value (two-tailed):", p_value)

# Decision
alpha = 0.05
if p_value < alpha:
    print("Reject null hypothesis: sample mean is significantly different from", mu0)
else:
    print("Fail to reject null hypothesis: no significant difference from", mu0)
```

**Question 8**: Write a Python script to simulate data from a normal distribution and calculate the 95% confidence interval for its mean. Plot the data using Matplotlib.

(*Include your Python code and output in the code box below.*) **Answer:**

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

def simulate_normal_data(mu=0.0, sigma=1.0, n=30, seed=None):
    if seed is not None:
        np.random.seed(seed)
    data = np.random.normal(loc=mu, scale=sigma, size=n)
    return data

def confidence_interval_known_sigma(data, sigma, confidence=0.95):
    """
    CI when population sigma is known.
    """
    n = len(data)
    mean = np.mean(data)
    z = stats.norm.ppf( (1 + confidence) / 2 )
    se = sigma / np.sqrt(n)
    lower = mean - z * se
    upper = mean + z * se
    return mean, (lower, upper)

def confidence_interval_unknown_sigma(data, confidence=0.95):
    """
    CI when population sigma is unknown: use t-distribution.
    """
```

```
    n = len(data)
    mean = np.mean(data)
    s = np.std(data, ddof=1)  # sample standard deviation
    se = s / np.sqrt(n)
    t_crit = stats.t.ppf( (1 + confidence) / 2, df=n-1 )
    lower = mean - t_crit * se
    upper = mean + t_crit * se
    return mean, (lower, upper)

def plot_data_with_ci(data, ci, title='Data with 95% CI for mean'):
    """
    Plot histogram of data, sample mean, and confidence interval for mean.
    """
    mean = np.mean(data)
    lower, upper = ci
    plt.figure(figsize=(8,5))
    # Histogram
    plt.hist(data, bins='auto', alpha=0.7, color='skyblue', edgecolor='black', density=True)

    # Plot mean line
    plt.axvline(mean, color='red', linestyle='--', linewidth=2,
```

**Question 9:** Write a Python function to calculate the Z-scores from a dataset and visualize the standardized data using a histogram. Explain what the Z-scores represent in terms of standard deviations from the mean.

(*Include your Python code and output in the code box below.*) **Answer:**

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

def compute_z_scores(data):
    """
    Given a list or array of numeric data, return the Z-scores:
    (X - mean) / standard_deviation
    Uses sample standard deviation (ddof=1).
    """
    arr = np.array(data, dtype=float)
    mean = np.mean(arr)
    std = np.std(arr, ddof=1)  # ddof=1 for sample SD
    z_scores = (arr - mean) / std
    return z_scores, mean, std
```

```
def plot_standardized_data(z_scores, nbins=30):
    """
    Plot histogram of Z-scores.
    Also overlay a standard normal (mean=0, std=1) curve for comparison.
    """
    plt.figure(figsize=(8, 5))
    # Histogram of z-scores
    count, bins, ignored = plt.hist(z_scores, bins=nbins, density=True, alpha=0.7,
                                    color='skyblue', edgecolor='black',
                                    label='
```