1. What is hypothesis testing in statistics?

Hypothesis testing is a statistical procedure used to decide whether there is enough evidence in a sample of data to support a particular claim about a population parameter. The logic is: assume the claim of no effect (the null hypothesis) is true, then see how likely the observed data are under that assumption; if they're very unlikely, you reject the null in favour of an alternative hypothesis. BCcampus Open Publishing+2EIMT+2
In simpler words: you collect data, you test whether what you observe is consistent with a "no-effect/no-difference" scenario and if it's not, you conclude something else is happening.

---

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

- The **null hypothesis**, often noted H0H_0H0, is a statement of no effect, no difference, or no change in the population. It sets up a baseline or status-quo assumption. BCcampus Open Publishing+1

- The **alternative hypothesis**, often noted H1H_1H1 or HaH_aHa, is what you want to show: that there *is* an effect, difference or change. Scribbr+1
  Difference: the null is what you assume to be true unless evidence shows otherwise; the alternative is what you suspect might be true instead of the null. In hypothesis testing you evaluate whether data provide sufficient evidence to reject the null in favour of the alternative.

---

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

The significance level, denoted by α\alphaα, is the threshold probability you set *before* doing the test to determine how extreme the data must be (under the null hypothesis) in order to reject H0H_0H0. EIMT+1
In practice:

- You compute the test statistic and then a *p-value* which is the probability of observing data as extreme or more extreme assuming H0H_0H0 is true. PennState: Statistics Online Courses

- If the p-value ≤ α, you reject the null hypothesis; if p-value > α, you do not reject it. PennState: Statistics Online Courses+1
  So significance level sets how strong the evidence must be to claim the alternative.

---

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

- **Type I error (false positive)**: rejecting the null hypothesis when it is in fact true. The probability of a Type I error is α\alphaα, the significance level. Scribbr+1
  Example: You test a new drug and conclude it *does* improve outcomes (so you reject H0H_0H0: “no improvement”), but actually the drug has no effect (so you made a Type I error).

- **Type II error (false negative)**: failing to reject the null hypothesis when the alternative hypothesis is in fact true. Denoted by β\betaβ. EIMT+1
  Example: You test the same drug and conclude it *doesn’t* improve outcomes (so you keep H0H_0H0), but in truth the drug does work — that is a Type II error.

---

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

Difference / When to use:

- A **Z-test** uses the standard normal distribution (Z-distribution). It is appropriate when the population standard deviation is known *or* when the sample size is large enough that the sampling distribution is approximately normal and you can treat the sample standard deviation as a good estimate. DataCamp+1

- A **T-test** uses the Student’s t-distribution, which has “heavier tails” than the normal distribution, and is used when the population standard deviation is unknown *and* typically the sample size is small. study.com+1
  In summary:

- Use Z-test: when population variance is known or sample is large (often n ≥ 30) and assumptions of normality hold.

- Use T-test: when variance is unknown (so you estimate it from the sample) and/or sample size is small, so you need the extra caution of the t-distribution. Testbook

---

6. Python program to generate a binomial distribution (n = 10, p = 0.5) and plot its histogram.

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

# Parameters
n = 10
```

```
p = 0.5
num_samples = 10000

# Generate binomial data
data = np.random.binomial(n, p, size=num_samples)

# Plot histogram
plt.hist(data, bins=range(n+2), align='left', edgecolor='black')
plt.title(f'Binomial distribution (n={n}, p={p})')
plt.xlabel('Number of successes')
plt.ylabel('Frequency')
plt.xticks(range(n+1))
plt.show()
```

Output explanation:
 This code will produce a histogram of how often 0, 1, 2, … up to 10 successes occurred in the 10-trial binomial experiments, across 10,000 simulations. The distribution should roughly peak around 5 successes (since n · p=10×0.5=5n \cdot p = 10 \times 0.5 = 5n · p=10×0.5=5).

---

7. Implement hypothesis testing using Z-statistics for a sample dataset in Python. Interpret results.

Here is Python code and a simple interpretation.

```
import numpy as np
import scipy.stats as stats

# 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]

# Hypothesised population mean
mu0 = 50

# Sample statistics
xbar = np.mean(sample_data)
```

```
n = len(sample_data)
# Here we assume we *know* population standard deviation sigma
# For demonstration we'll estimate it anyway (but in true Z-test
sigma should be known)
sigma = np.std(sample_data, ddof=0)  # using population sd estimate

# Compute Z statistic
z_stat = (xbar - mu0) / (sigma / np.sqrt(n))
# Compute two-tailed p-value
p_value = 2 * (1 - stats.norm.cdf(abs(z_stat)))

print(f"Sample mean = {xbar:.3f}")
print(f"Z statistic = {z_stat:.3f}")
print(f"P-value (two-tailed) = {p_value:.4f}")

# Interpretation
alpha = 0.05
if p_value <= alpha:
    print("Reject the null hypothesis: sample mean significantly
different from 50")
else:
    print("Fail to reject the null hypothesis: no significant
evidence that the mean differs from 50")
```

Interpretation:

- The code computes a Z-statistic for the hypothesis H0:μ=50H_0: \mu = 50H0:μ=50 vs Ha:μ≠50H_a: \mu \neq 50Ha:μ=50.
- Based on the p-value and chosen α (say 0.05), you decide whether to reject H0H_0H0.
- If p-value is less than or equal to 0.05 you conclude the mean **is** significantly different from 50. If p > 0.05 you conclude you don't have sufficient evidence to say it differs. Because here we actually used the sample's standard deviation (violating the ideal "known population sigma" assumption for Z-test) it's a bit of a hack—but it demonstrates the process. In real practice if sigma is unknown you would use a T-test.

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

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

# Simulate data
np.random.seed(0)
mu_true = 100
sigma_true = 15
n = 200
data = np.random.normal(mu_true, sigma_true, size=n)

# Sample mean and standard error
xbar = np.mean(data)
se = sigma_true / np.sqrt(n)  # assume known sigma; if unknown use
sample sd

# For 95% CI: z* ≈ 1.96
z_star = stats.norm.ppf(0.975)
ci_lower = xbar - z_star * se
ci_upper = xbar + z_star * se

print(f"Sample mean = {xbar:.3f}")
print(f"95% confidence interval for mean = ({ci_lower:.3f},
{ci_upper:.3f})")

# Plot data histogram
plt.hist(data, bins=30, edgecolor='black', alpha=0.7)
plt.axvline(x=ci_lower, color='red', linestyle='--', label='95% CI
lower')
plt.axvline(x=ci_upper, color='red', linestyle='--', label='95% CI
upper')
plt.title('Histogram of simulated normal data with 95% CI lines')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

What this does / output:

- It simulates 200 draws from a normal distribution with true mean 100 and standard deviation 15.
- It computes the sample mean, then computes the 95% confidence interval for the population mean (given known sigma).
- It then plots the histogram of the data, and marks the lower and upper bounds of the 95% CI on the plot.
- You will see something like:
    - Sample mean maybe ~100 (close to true 100)
    - Confidence interval something like (≈ 96 – 104) depending on the random draw
    - The histogram showing the distribution of the 200 simulated values.

---

9. Python function to calculate the Z-scores from a dataset and visualize the standardized data using a histogram. Explain what the Z-scores represent.

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

def compute_z_scores(data):
    mu = np.mean(data)
    sigma = np.std(data, ddof=0)
    z_scores = (data - mu) / sigma
    return z_scores

# Example data
data = np.array([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])

z = compute_z_scores(data)
```

```
print("First 10 Z-scores:", z[:10])

plt.hist(z, bins=20, edgecolor='black', alpha=0.7)
plt.title('Histogram of Z-scores (standardized data)')
plt.xlabel('Z-score')
plt.ylabel('Frequency')
plt.axvline(x=0, color='red', linestyle='--', label='Mean = 0')
plt.legend()
plt.show()
```

What Z-scores represent:
A Z-score indicates how many standard deviations a given data point is from the mean of the data set.

- If a point has Z-score +2, then it is 2 standard deviations *above* the mean.

- If a point has Z-score −1.5, it is 1.5 standard deviations *below* the mean.
  Standardizing data (converting to Z-scores) allows you to compare values from different distributions or to see how extreme values are relative to their own distribution.
  In the histogram you will see the distribution of these standardized values, centered around 0 (mean) with spread such that roughly 68% lie within ±1 (if roughly normally distributed) and ~95% within ±2.