**E**stimation **A**nd **C**onfidence **I**ntervals

**Background**

In quality control processes, especially when dealing with high-value items, destructive sampling is a necessary but costly method to ensure product quality. The test to determine whether an item meets the quality standards destroys the item, leading to the requirement of small sample sizes due to cost constraints.

**Scenario**

A manufacturer of print-heads for personal computers is interested in estimating the mean durability of their print-heads in terms of the number of characters printed before failure. To assess this, the manufacturer conducts a study on a small sample of print-heads due to the destructive nature of the testing process.

**Data**

A total of 15 print-heads were randomly selected and tested until failure. The durability of each print-head (in millions of characters) was recorded as follows:

1.13, 1.55, 1.43, 0.92, 1.25, 1.36, 1.32, 0.85, 1.07, 1.48, 1.20, 1.33, 1.18, 1.22, 1.29

**Assignment Tasks**

**a. Build 99% Confidence Interval Using Sample Standard Deviation**

Assuming the sample is representative of the population, construct a 99% confidence interval for the mean number of characters printed before the print-head fails using the sample standard deviation. Explain the steps you take and the rationale behind using the t-distribution for this task.

**Answer:**

**Steps:**

1. Calculate the sample mean and sample standard deviation from the provided data.

2. Determine the degrees of freedom (n-1), which is used to find the appropriate t-critical value.

3. Determine the t-critical value using the desired confidence level (99%) and degrees of freedom. Use the t.ppf function from the scipy.stats module to find this value.

4. Calculate the margin of error: this is the product of t\_critical and the standard error of the mean. The standard error is the sample standard deviation divided by the square root of the sample size.

5. Construct the confidence interval: The interval is formed by subtracting the margin of error from the sample mean (lower bound) and adding the margin of error to the sample mean (upper bound).

import numpy as np

from scipy.stats import t

data = [1.13, 1.55, 1.43, 0.92, 1.25, 1.36, 1.32, 0.85, 1.07, 1.48, 1.20, 1.33, 1.18, 1.22, 1.29]

sample\_mean = np.mean(data)

sample\_std = np.std(data, ddof=1)

# Sample size

n = len(data)

# Degrees of freedom

df = n - 1

confidence\_level = 0.99

# Find the critical t-value

alpha = 1 - confidence\_level

t\_critical = t.ppf(1 - alpha / 2, df)

# Calculate the margin of error

margin\_of\_error = t\_critical \* (sample\_std / np.sqrt(n))

# Calculate the confidence interval

lower\_bound = sample\_mean - margin\_of\_error

upper\_bound = sample\_mean + margin\_of\_error

# Print the results

print(f"Sample Mean: {sample\_mean:.4f}")

print(f"Sample Standard Deviation: {sample\_std:.4f}")

print(f"Sample Size: {n}")

print(f"Degrees of Freedom: {df}")

print(f"t-critical value: {t\_critical:.4f}")

print(f"Margin of Error: {margin\_of\_error:.4f}")

print(f"99% Confidence Interval: ({lower\_bound:.4f}, {upper\_bound:.4f})")

**Sample Mean: 1.2387**

**Sample Standard Deviation: 0.1932**

**Sample Size: 15**

**Degrees of Freedom: 14**

**t-critical value: 2.9768**

**Margin of Error: 0.1485**

**99% Confidence Interval: (1.0902, 1.3871)**

Explanation

We use the t-distribution because the population standard deviation is unknown, and the sample size is relatively small (n=15). The t-distribution is more appropriate for small sample sizes when the population standard deviation is not known.

**b. Build 99% Confidence Interval Using Known Population Standard Deviation**

If it were known that the population standard deviation is 0.2 million characters, construct a 99% confidence interval for the mean number of characters printed before failure.

import numpy as np

from scipy.stats import norm

data = [1.13, 1.55, 1.43, 0.92, 1.25, 1.36, 1.32, 0.85, 1.07, 1.48, 1.20, 1.33, 1.18, 1.22, 1.29]

# Population standard deviation

population\_std = 0.2

# Calculate the sample mean

sample\_mean = np.mean(data)

# Sample size

n = len(data)

# Confidence level

confidence\_level = 0.99

# Calculate the critical z-value

alpha = 1 - confidence\_level

z\_critical = norm.ppf(1 - alpha / 2)

# Calculate the margin of error

margin\_of\_error = z\_critical \* (population\_std / np.sqrt(n))

# Calculate the confidence interval

lower\_bound = sample\_mean - margin\_of\_error

upper\_bound = sample\_mean + margin\_of\_error

# Print the results

print(f"Sample Mean: {sample\_mean:.4f}")

print(f"Population Standard Deviation: {population\_std:.4f}")

print(f"Sample Size: {n}")

print(f"z-critical value: {z\_critical:.4f}")

print(f"Margin of Error: {margin\_of\_error:.4f}")

print(f"99% Confidence Interval: ({lower\_bound:.4f}, {upper\_bound:.4f})")

**Sample Mean: 1.2387**

**Population Standard Deviation: 0.2000**

**Sample Size: 15**

**z-critical value: 2.5758**

**Margin of Error: 0.1330**

**99% Confidence Interval: (1.1057, 1.3717)**