ex5

February 19, 2024

[ ]:
```
EX NO 5 - Statistical Inference
Batch 2
Date:19-02-2024
HARIHARAN K - URK22AI1048
```

[ ]:
```
Aim

To demonstrate the statistical interferences used for data science application␣
↪using
python language.

Description

Inferential statistics are used to draw inferences from the sample of a huge␣
↪data set.
Random samples of data are taken from a population, which are then used to␣
↪describe and
make inferences and predictions about the population.
Sample Mean and population Mean
Sample mean is the arithmetic mean of random sample values drawn from the
population. Population mean represents the actual mean of the whole population.␣
↪If the
sample is random and sample size is large then the sample mean would be a good␣
↪estimate of
the population mean.

Correlation Coefficient

The correlation coefficient quantifies the relationship between the two␣
↪variables. There
are two methods of calculating the Correlation Coefficient and its matrix -␣
↪Pearson and
Spearman.

Covariance Matrix
```

```
It is a square matrix giving the covariance between each pair of elements of a␣
 ↪given random
vector.

Hypothesis Testing using Z Test

Hypothesis testing is a statistical method that is used in making statistical␣
 ↪decisions using
experimental data. One of the ways to perform hypothesis testing is Z-test,␣
 ↪where the Two-
sample Z-test is used to test whether the two datasets are similar or not.␣
 ↪Also, Z-test is used
when the sample size is greater than 30.


Confidence Interval

A confidence interval displays the probability that a parameter will fall␣
 ↪between a pair of
values around the mean. Confidence intervals measure the degree of uncertainty␣
 ↪or certainty
in a sampling method. They are most often constructed using confidence levels␣
 ↪of 95% or
99%.
FORMULA...
```

[4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
```

[7]:
```
df = pd.read_csv('sm.csv')
ff = df[df['Gender'] == 'Female']
```

[8]:
```
#URK22AI1048
#1q Calculate the sample mean for ‘Unit price’ column with n=500 and observe
sample_mean_500 = ff['Unit price'].sample(n=500).mean()
sample_mean_500
```

[8]: 56.35588

[9]:
```
#URK22AI1048
#2q Calculate the sample mean for 'Unit price' column with n=1000
sample_mean_1000 = ff['Unit price'].sample(n=1000).mean()
sample_mean_1000
```

[9]: 55.67246

[10]:
```python
#URK22AI1048
#3q Calculate the population mean for 'Unit price' column
population_mean = ff['Unit price'].mean()
population_mean
```

[10]: 55.263952095808385

[11]:
```python
#URK22AI1048
#4q Calculate the confidence interval (CI) with sample mean for 'Unit price'␣
↪column of n=500 and confidence level of 95%
n = 500
sample = ff['Unit price'].sample(n)
sample_mean = sample.mean()
sample_std = sample.std()
# Replace with your sample size
confidence_level = 0.95

# Calculate the standard error
standard_error = sample_std / (n ** 0.5)

# Calculate the z-score corresponding to the desired confidence level
z_score = stats.norm.ppf(1 - ((1 - confidence_level) / 2))

# Calculate the confidence interval
lower_bound = sample_mean - z_score * standard_error
upper_bound = sample_mean + z_score * standard_error

# Print the confidence interval
print("Confidence Interval: [{:.2f}, {:.2f}]".format(lower_bound, upper_bound))
```

```
Confidence Interval: [53.54, 58.43]
```

[12]:
```python
#URK22AI1048
#5q Change the confidence level to 99% and observe the confidence interval for␣
↪the same sample mean for ‘Unit price’ column of n=500.
n = 500
sample = ff['Unit price'].sample(n)
sample_mean = sample.mean()
sample_std = sample.std()
# Replace with your sample size
confidence_level = 0.99

# Calculate the standard error
standard_error = sample_std / (n ** 0.5)

# Calculate the z-score corresponding to the desired confidence level
z_score = stats.norm.ppf(1 - ((1 - confidence_level) / 2))
```

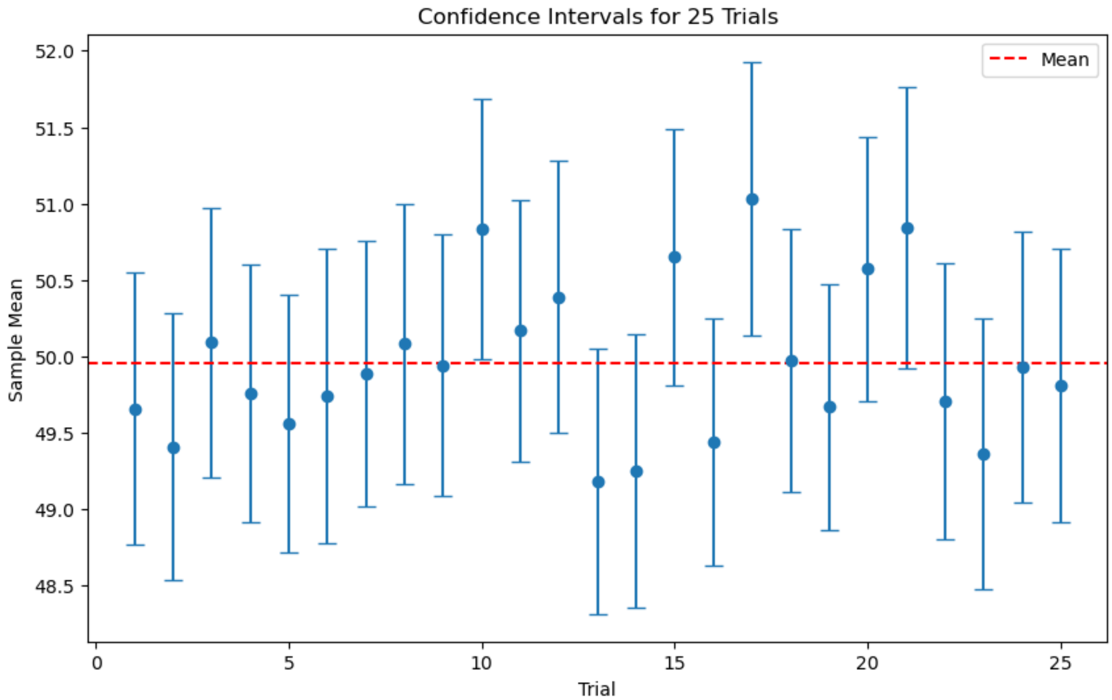
```
# Calculate the confidence interval
lower_bound = sample_mean - z_score * standard_error
upper_bound = sample_mean + z_score * standard_error

# Print the confidence interval
print("Confidence Interval: [{:.2f}, {:.2f}]".format(lower_bound, upper_bound))
```

```
Confidence Interval: [51.30, 57.56]
```

[13]:
```
#URK22AI1048
#6q Calculate and plot the Confidence Intervals for 25 Trials with n=500 and␣
↪CI=95% for ‘Unit price’ column. Observe the results.
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats

n = 500
confidence_level = 0.95
num_trials = 25

# Generate random sample data
np.random.seed(42)  # Set a seed for reproducibility
sample_data = np.random.normal(loc=50, scale=10, size=(n, num_trials))

# Calculate the sample mean for each trial
sample_means = np.mean(sample_data, axis=0)

# Calculate the standard error
standard_error = np.std(sample_data, axis=0) / np.sqrt(n)

# Calculate the z-score corresponding to the desired confidence level
z_score = stats.norm.ppf(1 - ((1 - confidence_level) / 2))

# Calculate the confidence intervals for each trial
lower_bounds = sample_means - z_score * standard_error
upper_bounds = sample_means + z_score * standard_error
print("Confidence Interval: [{:.2f}, {:.2f}]".format(lower_bound, upper_bound))

# Plot the confidence intervals
plt.figure(figsize=(10, 6))
plt.errorbar(np.arange(1, num_trials + 1), sample_means, yerr=[sample_means -␣
↪lower_bounds, upper_bounds - sample_means],
             fmt='o', capsize=5)
plt.axhline(y=np.mean(sample_means), color='r', linestyle='--', label='Mean')
plt.xlabel('Trial')
plt.ylabel('Sample Mean')
plt.title('Confidence Intervals for 25 Trials')
```

```
plt.legend()
plt.show()
```

```
Confidence Interval: [51.30, 57.56]
```



[14]:
```
#URK22AI1048
#7q Calculate the Correlation Coefficient using Pearson for the given table
from scipy.stats import pearsonr, spearmanr

# Given data
data = {'Person': ['A', 'B', 'C', 'D', 'E'], 'Hand': [17, 15, 19, 17, 21],␣
 ↪'Height': [150, 154, 169, 172, 175]}

# Extracting hand and height data
hand = data['Hand']
height = data['Height']

# Calculate correlation coefficient using Pearson
pearson_corr, _ = pearsonr(hand, height)
print("Pearson Correlation Coefficient:", pearson_corr)

# Calculate correlation coefficient using Spearman
spearman_corr, _ = spearmanr(hand, height)
print("Spearman Correlation Coefficient:", spearman_corr)
```

```
Pearson Correlation Coefficient: 0.721314718045345
Spearman Correlation Coefficient: 0.6668859288553503
```

[15]:
```python
#URK22AI1048
#8q Calculate the Correlation Coefficient using Spearman for the given table
import pandas as pd
from scipy.stats import spearmanr

data = {'Person': ['A','B','C','D','E'], 'Hand': [17,15,19,17,21],'Height':
 ↪[150,154,169,172,175]}
df = pd.DataFrame(data)

# Spearman correlation
df.rank()
spearman_corr, _ = spearmanr(df['Hand'], df['Height'])
print(f'Spearman correlation coefficient: {spearman_corr:.2f}')
```

```
Spearman correlation coefficient: 0.67
```

[16]:
```python
#URK22AI1048
#9q Calculate the Covariance Matrix for the given data and analyse it
import numpy as np

# Given data
data = {'Math': [90, 90, 60, 60, 30], 'English': [60, 90, 60, 60, 30], 'Art':␣
 ↪[90, 30, 60, 90, 30]}

# Convert data into a NumPy array
data_array = np.array([data['Math'], data['English'], data['Art']])

# Calculate covariance matrix
covariance_matrix = np.cov(data_array)

print("Covariance Matrix:")
print(covariance_matrix)
```

```
Covariance Matrix:
[[630. 450. 225.]
 [450. 450.   0.]
 [225.   0. 900.]]
```

[17]:
```python
#URK22AI1048
```

```
#10q Perform a hypothesis testing with Z-test A herd of 1,500 steer was fed a␣
 ↪special high-protein grain for a month, has the standard deviation of weight␣
 ↪gain for the entire herd was 7.1 and average weight gain per steer for the␣
 ↪month was 5 pounds. By feeding the herd with special high-protein grain, it␣
 ↪is claimed that the weight of the herd has increased. In order to test this␣
 ↪claim, a random sample of 29 were weighed and had gained an average of 6.7␣
 ↪pounds. Can we support the claim at 5 % level?
import numpy as np
from scipy.stats import norm

# Given data
population_mean = 1800
population_std = 100
sample_mean = 1850
sample_size = 50
alpha = 0.01

# Calculate the Z-score
z_score = (sample_mean - population_mean) / (population_std / np.
 ↪sqrt(sample_size))

# Calculate the critical value
critical_value = norm.ppf(1 - alpha)
print("critical_value:",critical_value)
print("z_score:",z_score)

# Perform the hypothesis test
if z_score > critical_value:
    print("Reject the null hypothesis. There is enough evidence to support the␣
 ↪claim.")
else:
    print("Fail to reject the null hypothesis. There is not enough evidence to␣
 ↪support the claim.")
```

```
critical_value: 2.3263478740408408
z_score: 3.5355339059327378
Reject the null hypothesis. There is enough evidence to support the claim.
```

[ ]:
```
Result:
The Above Program were Created and Executed Successfully
```