Statistical Interference

February 25, 2024

URK22AI1085

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..

[1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
```

```
/opt/anaconda3/lib/python3.11/site-packages/pandas/core/arrays/masked.py:60:
UserWarning: Pandas requires version '1.3.6' or newer of 'bottleneck' (version
'1.3.5' currently installed).
  from pandas.core import (
```

[3]:
```python
df = pd.read_csv('supermarket - supermarket.csv')
ff = df[df['Gender'] == 'Female']
```

[7]:
```python
# 1. Calculate the sample mean for 'Unit price' column with n=500
sample_mean_500 = ff['Unit price'].sample(n=500).mean()
sample_mean_500
```

[7]: 55.04540000000001

[9]:
```python
# 2. 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]: 55.295289999999994

[11]:
```python
# 3. Calculate the population mean for 'Unit price' column
population_mean = ff['Unit price'].mean()
population_mean
```

[11]: 55.263952095808385

[12]:
```python
# 4. 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))
```

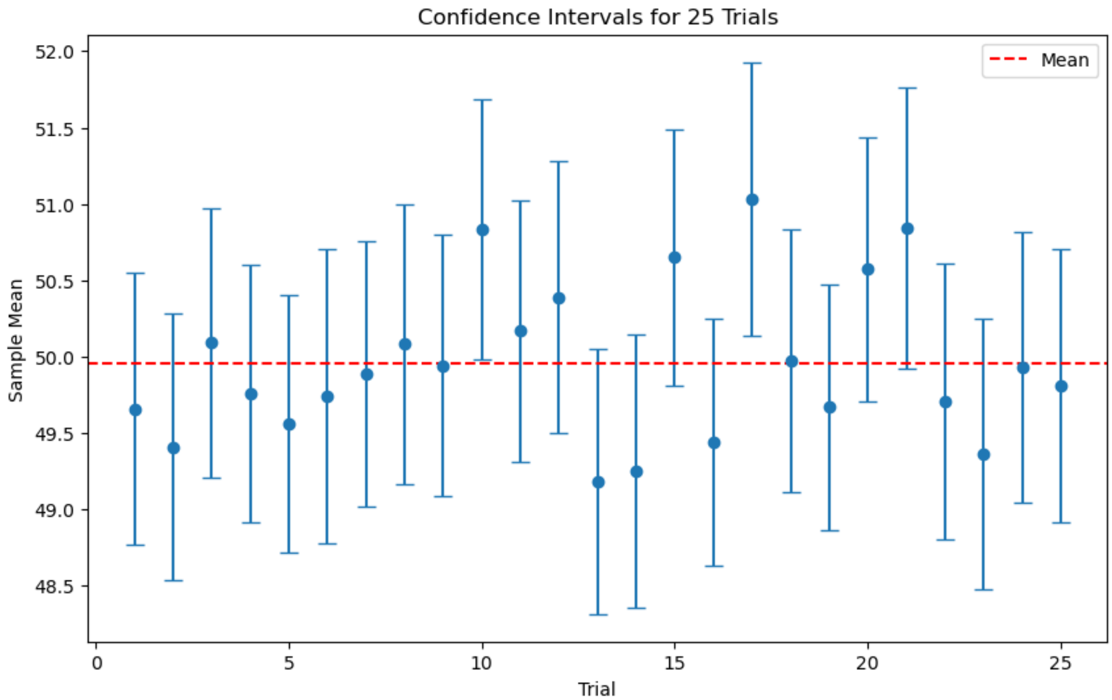
```
Confidence Interval: [52.03, 56.80]
```

[21]:
```python
#5. 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: [51.88, 58.25]
```

[24]:
```python
# 6. Calculate and plot the Confidence Intervals for 25 Trials with n=500 and␣
↪CI=95% for 'Unit price' column
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: [51.88, 58.25]
```



[31]:
```
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
```

[29]:
```
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
```

[30]:

```
#9
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.]]
```

[35]:

```
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)
```

```python
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: Thus, the program to demonstrate the statistical interferences used for data science application using python language was executed successfully.