Untitled

October 3, 2024

[ ]:
```
#question 1

#The standard error of the mean (SEM): described as the
#"standard deviation of the distribution of bootstrapped means,
#" captures the variability of the sample mean when repeatedly drawing samples.
#It reflects how much the sample mean would fluctuate if you repeatedly
#resampled and calculated new means, indicating the precision of the sample␣
↪mean as an estimate of the true population mean.

#the standard deviation (SD) of the original data measures
#the spread of individual data points around the mean, reflecting
#the variability within the dataset itself. It shows how much
#the values differ from the average.
```

[ ]:
```
#question 2
#1. Start with the sample mean: Calculate the mean of your sample data.
#2. Use the SEM: Multiply the SEM by 1.96 (the approximate z-value for 95%␣
↪confidence under a normal distribution).
#3. Calculate the interval:
#Lower bound = Sample Mean - (1.96 × SEM)
#Upper bound = Sample Mean + (1.96 × SEM)
```

[ ]:
```
#question 3
import numpy as np

# Assuming original dataset
data = np.array([...])

# Number of bootstrap samples
B = 1000

# Initialize an array to store bootstrapped means
bootstrapped_means = np.zeros(B)

# Generate bootstrapped samples and calculate means
for i in range(B):
    sample = np.random.choice(data, size=len(data), replace=True)
```

```
    bootstrapped_means[i] = np.mean(sample)

# Calculate the 95% confidence interval (2.5th and 97.5th quantiles)
ci_lower = np.quantile(bootstrapped_means, 0.025)
ci_upper = np.quantile(bootstrapped_means, 0.975)

print(f"95% Confidence Interval: ({ci_lower}, {ci_upper})")

#explaination
#To calculate a 95% confidence interval (CI) for the mean using bootstrapping,
#you repeatedly sample from your dataset with replacement to create many
#new datasets (bootstrap samples). For each sample, you calculate the
#mean, then store all the means in an array. To find the 95% CI,
#you use `np.quantile()` to get the 2.5th and 97.5th percentiles
#of the bootstrapped means. These percentiles give the lower and
#upper bounds of the CI, meaning that 95% of the time, the true
#mean will lie within this range in repeated samples.

#Using np.quantile() to calculate the 2.5th and 97.5th percentiles
#of bootstrapped means gives us a 95% confidence interval,
#making it a robust method for non-parametric confidence
#interval estimation.
```

[1]:
```
#question 4

# step 1: collect data
import plotly.express as px
import plotly.graph_objects as go
import pandas as pd
import numpy as np
df = pd.read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/
↪2e9bd5a67e09b14d01f616b00f7f7e0931515d24/data/2020/2020-07-07/coffee_ratings.
↪csv")
df = df.rename(columns={'country_of_origin': 'origin', 'total_cup_points':␣
↪'points'})

df = df[df['points']>65] # ignore some very low scores
df = df[~df['origin'].isna()] # remove rows with unknown origin

df['origin'] = df['origin'].str.replace("?","'") # fix character encoding issue
df['origin_original'] = df.origin.copy().values # save original (corrected)␣
↪names
df.shape
```

[1]: (1335, 44)

[5]:

```python
# step2: loop
N = 1000
boot_median = np.zeros(N)

for i in range(N):
    boot_sample = np.random.choice(df['points'], size=len(df['points']),␣
 ↪replace=True)
    boot_median[i] = np.median(boot_sample)
```

[7]:

```python
#step3: draw histagram

import plotly.express as px
import pandas as pd

# Assuming boot_median has been calculated correctly
fig = px.histogram(pd.DataFrame({"x": boot_median}), x="x")
fig.show()
```

[8]:

```python
#stepcalculate

# Calculate the 2.5th and 97.5th percentiles
quantiles = np.quantile(boot_median, [0.025, 0.975])

# Extract lower and upper bounds from the result
ci_lower, ci_upper = quantiles

print(f"95% Confidence Interval: ({ci_lower}, {ci_upper})")
```

```
95% Confidence Interval: (82.42, 82.67)
```

[ ]:

```python
# question 5
#Distinguishing between the population parameter and the sample
#statistic is essential because the population parameter represents
#the true value we want to estimate, while the sample statistic is
#derived from our sample data. Confidence intervals provide a
#range of plausible values for the population parameter based
#on the sample statistic, reflecting the uncertainty due to
#sampling variability. This distinction helps us understand
#the reliability of our estimates and the potential error in
#inferring population characteristics from a sample.
```

[ ]:

```python
# question 6
# question:What is the process of bootstrapping?
#Bootstrapping is a method where you repeatedly take random
#samples from your existing data (with replacement) to create many
#"mini-samples." By calculating statistics like averages from these
#mini-samples, you can estimate the variability and distribution
```

```
#of the statistic.

# question: What is the main purpose of bootstrapping?
#The main purpose of bootstrapping is to assess the reliability
#of your estimates, like averages or medians. It helps you create
#confidence intervals, showing a range of values where the true
#population parameter might lie.

#question: How could you use bootstrapping to assess whether your guess about␣
↪the average is plausible?
#If you have a guess about the average (like 10),
#you can take a sample from your data and use bootstrapping
#to generate many averages. If most of these averages are close
#to your guess, it suggests your guess is plausible; if not,
#it may need reconsideration.
```

[ ]:
```
#question 7
#When a confidence interval (CI) includes zero, it means there is
#a plausible range of values for the effect, including no effect
#(zero). This suggests that the observed data is consistent with
#the possibility that the drug has no effect on average. As a result,
#we "fail to reject the null hypothesis," which assumes no effect.

#To reject the null hypothesis, the CI must not include zero.
#This would indicate that the observed sample mean is significantly
#different from zero, suggesting that the drug likely has an effect.
```

[14]:
```
# question 8

#problem introduction:

#An explaination of the meaning of a Null Hypothesis of "no effect" in this␣
↪context
# null hypotheses: there is no different, no relationship, or impact in the␣
↪context being studied
# alternative hypothesis: there is different, relationship or impact in the␣
↪context being studied

#Data Visualization
# we can ilustrate the comparison of initial and final heath scores, a box plot␣
↪or bar graph can be used.
import pandas as pd

# Define the data as a list of dictionaries
data = {
    "PatientID": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
```

```
    "Age": [45, 34, 29, 52, 37, 41, 33, 48, 26, 39],
    "Gender": ['M', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F'],
    "InitialHealthScore": [84, 78, 83, 81, 81, 80, 79, 85, 76, 83],
    "FinalHealthScore": [86, 86, 80, 86, 84, 86, 86, 82, 83, 84]
}

# Create the DataFrame
df = pd.DataFrame(data)

# Display the DataFrame
df["d"]=df.FinalHealthScore-df.InitialHealthScore


print(df)
```

```
   PatientID  Age Gender  InitialHealthScore  FinalHealthScore  d
0          1   45      M                  84                86  2
1          2   34      F                  78                86  8
2          3   29      M                  83                80 -3
3          4   52      F                  81                86  5
4          5   37      M                  81                84  3
5          6   41      F                  80                86  6
6          7   33      M                  79                86  7
7          8   48      F                  85                82 -3
8          9   26      M                  76                83  7
9         10   39      F                  83                84  1
```

[3]:
```
import numpy as np
import pandas as pd
import plotly.express as px

# Re-create the DataFrame for completeness
data = {
    "PatientID": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    "Age": [45, 34, 29, 52, 37, 41, 33, 48, 26, 39],
    "Gender": ['M', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F'],
    "InitialHealthScore": [84, 78, 83, 81, 81, 80, 79, 85, 76, 83],
    "FinalHealthScore": [86, 86, 80, 86, 84, 86, 86, 82, 83, 84]
}
df = pd.DataFrame(data)
df["d"] = df["FinalHealthScore"] - df["InitialHealthScore"]

# Bootstrapping
N = 1000
boot_mean = np.zeros(N)

for i in range(N):
```

```
    boot_sample = np.random.choice(df["d"], size=len(df), replace=True)  #␣
↪Corrected here
    boot_mean[i] = np.mean(boot_sample)

# Calculate confidence intervals
ci_lower = np.percentile(boot_mean, 2.5)
ci_upper = np.percentile(boot_mean, 97.5)

# Creating histogram
fig = px.histogram(pd.DataFrame({"x": boot_mean}), x="x", title="Bootstrapped␣
↪Mean Differences")
fig.show()

# Print confidence intervals
print(f"95% Confidence Interval for the Mean Difference: [{ci_lower:.2f},␣
↪{ci_upper:.2f}]")

# Step 4: Supporting Visualization - Histogram of bootstrapped means
fig = px.histogram(pd.DataFrame({"Mean Difference": boot_mean}),
                   x="Mean Difference",
                   title="Bootstrapped Mean Differences",
                   labels={'Mean Difference': 'Mean Difference in Health␣
↪Scores'})
fig.add_vline(x=ci_lower, line_color='red', line_dash='dash',␣
↪annotation_text='Lower CI', annotation_position="bottom right")
fig.add_vline(x=ci_upper, line_color='green', line_dash='dash',␣
↪annotation_text='Upper CI', annotation_position="top right")
fig.show()
```

```
95% Confidence Interval for the Mean Difference: [0.90, 5.60]
```

[21]:

```
# discussion
# The confidence interval does include zero,
#we fail to reject the null hypothesis, which indicating
#that the vaccine not have a significant effect on health outcomes.

#Further Considerations
#Sample Size: Future studies should aim for larger samples for more
#reliable results.
#Longitudinal Studies: Assessing long-term effects could provide
#deeper insights.
#Additional Variables: Investigating other factors
#(e.g., comorbidities) may help understand the vaccine's impact better.
```

[ ]:

```
#chatbox: https://chatgpt.com/share/66ff1b1d-5974-8002-9e50-347bbe4a25ed
```