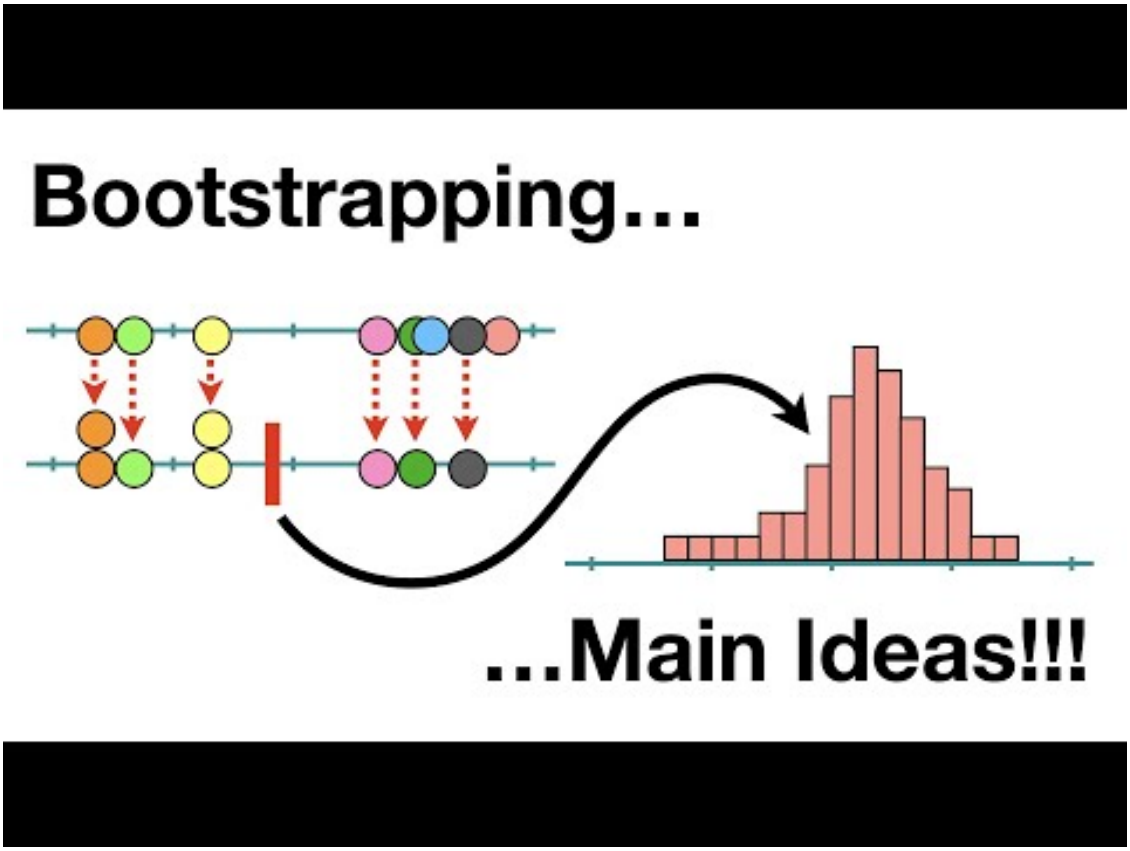
HW 4 Oct 3

October 4, 2024

[1]:
```
from IPython.display import YouTubeVideo
YouTubeVideo('Xz0x-8-cgaQ', width=800, height=500)
```

[1]:



1. The "Pre-lecture" video (above) mentioned the "standard error of the mean" as being the "standard deviation" of the distribution bootstrapped means. What is the difference between the "standard error of the mean" and the "standard deviation" of the original data? What distinct ideas do each of these capture? Explain this concisely in your own words.

"Standard error of the mean", SEM, measures how much discrepancy is likely in a sample's mean compared with the population mean. "Standard deviation", SD, measures the amount of variability, or dispersion, from the individual data values to the mean. The SEM is always be smaller than the SD and SEM takes the SD and divides it by the square root of the sample size.

2. The "Pre-lecture" video (above) suggested that the "standard error of the mean" could be used to create a confidence interval, but didn't describe exactly how to do this. How can we use the "standard error of the mean" to create a 95% confidence interval which "covers 95% of the bootstrapped sample means"? Explain this concisely in your own words.

We need to calculate the mean of the bootstrapped sample means first, and then determine the SEM of these bootstrapped means. Lastly, use the formula, Confidence Interval=Sample Mean±(Z×SEM), to find the range that covers approximately 95% of the bootstrapped sample means.

3. Creating the "sample mean plus and minus about 2 times the standard error" confidence interval addressed in the previous problem should indeed cover approximately 95% of the bootstrapped sample means. Alternatively, how do we create a 95% bootstrapped confidence interval using the bootstrapped means (without using their standard deviation to estimate the standard error of the mean)? Explain this concisely in your own words.

To create a 95% bootstrapped confidence interval by using the bootstrapped means, we can generate bootstrapped samples from the original dataset and calculate means first,and then sort bootstrapped means in ascending order. Lastly, find the values at the 2.5th percentile and the 97.5th percentile of the sorted means.

[ ]:
```
import numpy as np

# Sample data (you can replace this with your own data)
data = np.array([23, 45, 12, 36, 52, 28, 41, 33, 19, 44])

# Function to calculate bootstrap confidence interval
def bootstrap_ci(data, num_bootstraps=1000, alpha=0.05, stat_func=np.mean):
    """
    Calculate the bootstrap confidence interval for a given statistic.

    Parameters:
    - data: array-like, the sample data
    - num_bootstraps: int, number of bootstrap samples to generate
    - alpha: float, significance level (0.05 for 95% confidence interval)
    - stat_func: function, statistic to calculate (e.g., np.mean, np.median,␣
 ↪etc.)

    Returns:
    - ci: tuple, the lower and upper bounds of the confidence interval
    """
    # Generate bootstrap samples and compute the statistic
    bootstraps = np.random.choice(data, (num_bootstraps, len(data)),␣
 ↪replace=True)
    stats = np.array([stat_func(sample) for sample in bootstraps])

    # Calculate the lower and upper percentiles for the confidence interval
    lower_bound = np.percentile(stats, (alpha/2) * 100)
    upper_bound = np.percentile(stats, (1 - alpha/2) * 100)
```

```
    return lower_bound, upper_bound

# Calculate the bootstrap confidence interval for the mean
mean_ci = bootstrap_ci(data, stat_func=np.mean)
print(f"95% Bootstrap CI for the mean: {mean_ci}")

# Example of how to calculate a 95% bootstrap confidence interval for the median
median_ci = bootstrap_ci(data, stat_func=np.median)
print(f"95% Bootstrap CI for the median: {median_ci}")

# Example of how to calculate a 95% bootstrap confidence interval for the␣
 ↪standard deviation
std_dev_ci = bootstrap_ci(data, stat_func=np.std)
print(f"95% Bootstrap CI for the standard deviation: {std_dev_ci}")
```

Pre-lecture #4 Chatbot log: https://chatgpt.com/share/66ff1cc9-087c-8009-9ce2-a2b0fff6110b

5. The previous question addresses making a confidence interval for a population parameter based on a sample statistic. Why do we need to distinguish between the role of the popualation parameter and the sample sample statistic when it comes to confidence intervals? Explain this concisely in your own words.

The popualation parameter and the sample sample statistic have different roles in confidence intervals. Population Parameter is a fixed unknown value that represents a characteristic of the entire population, and the sample statistic is a value calculated from a sample and it estimates the population parameter. The main difference between those two is fundamental for statistical inference. Sample statistics are used to estimate population parameters, and confidence intervals provide reliability for those estimates, indicating the likely range of population parameters. Since confidence intervals help us estimate the true value for the population, it's important to distinguish between the role of popualation parameter and the sample sample statistic.

Post-lecture #5 Chatbot log: https://chatgpt.com/share/66ff2ad0-4c30-8009-95ea-3ae5f2ba25a2

6. Provide written answers explaining the answers to the following questions in an informal manner of a conversation with a friend with little experience with statistics.

- What is the process of bootstrapping?

-Bootstrapping is a statistical method that resamples a dataset many times and it help us to calculate standard errors, confidence intervals, and hypothesis testing. - What is the main purpose of bootstrapping?

-The main purpose of bootstrapping is to give us more useful information from our data. For example, when we have small dataset it help us to make the most out of the limited data by simulating new samples several times. In addition, we can make multiple estimates from the same data to assess the uncertainty or variability of a statistic. - If you had a (hypothesized) guess about what the average of a population was, and you had a sample of size n from that population, how could you use bootstrapping to assess whether or not your (hypothesized) guess might be plausible?

-First, we need to calculate and resample the average from a population sample. Then compute

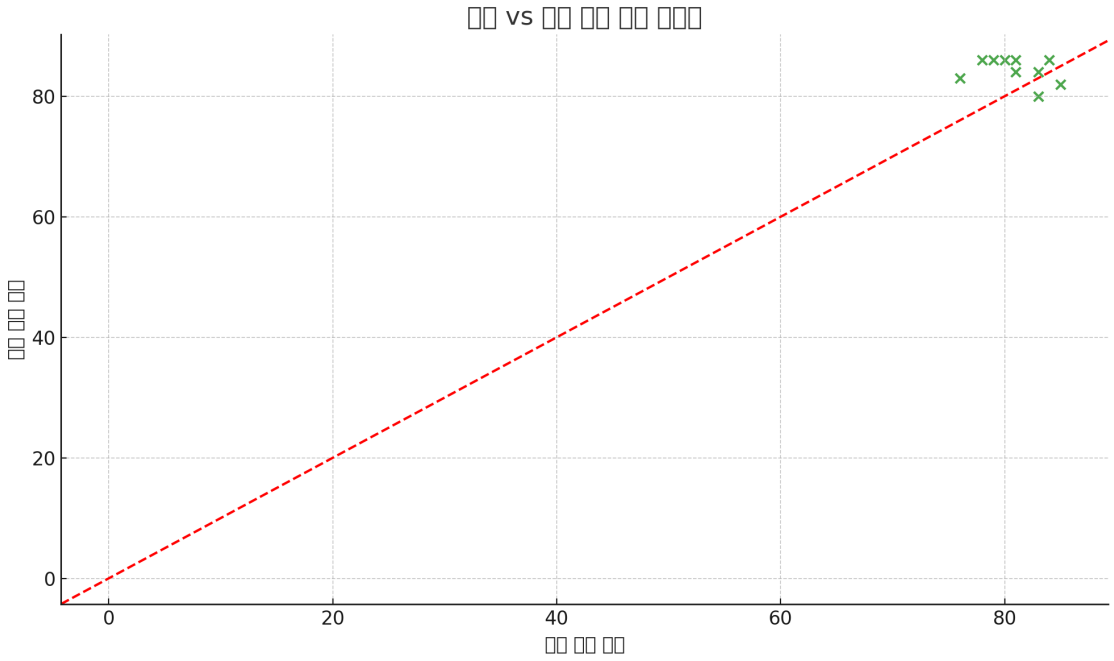
the average again for each of the new samples and make a confidence interval. If the average is inside the confidence interval, it might be correct but If it's outside, then it might not be right.

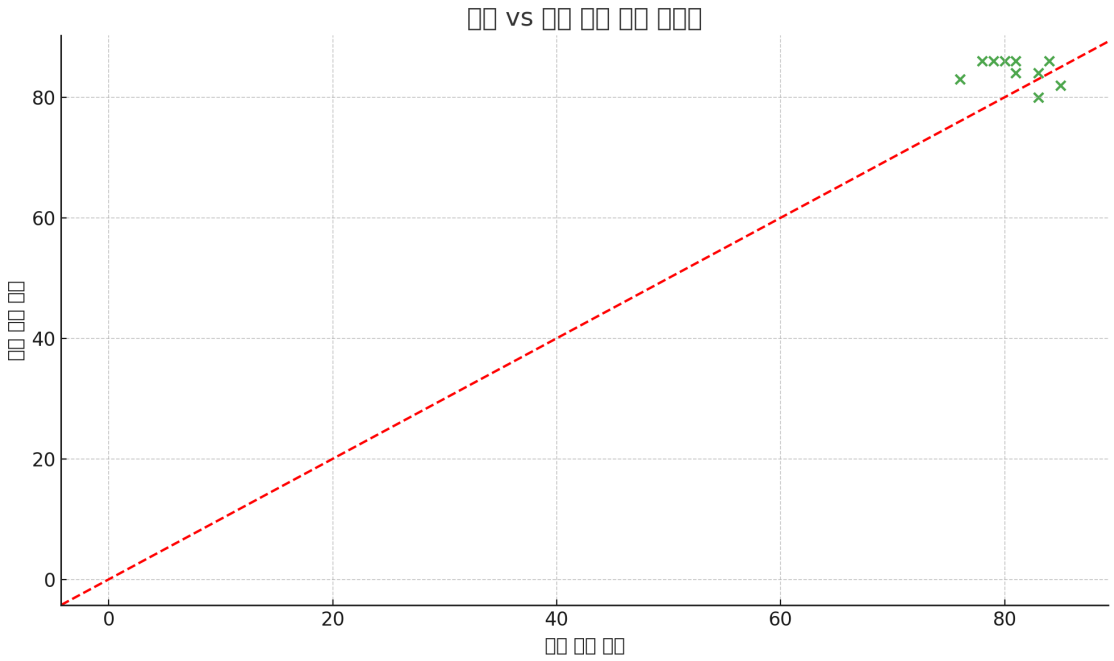
8. • An explaination of the meaning of a Null Hypothesis of "no effect" in this context The null hypothesis (H0) is a statistical hypothesis that suggests that there is no significant effect or relationship between two variables in the population being studied. In this context, the vaccine has no effect on the health scores of patients it will guide the evaluation of the vaccine's effectiveness based on health score data.

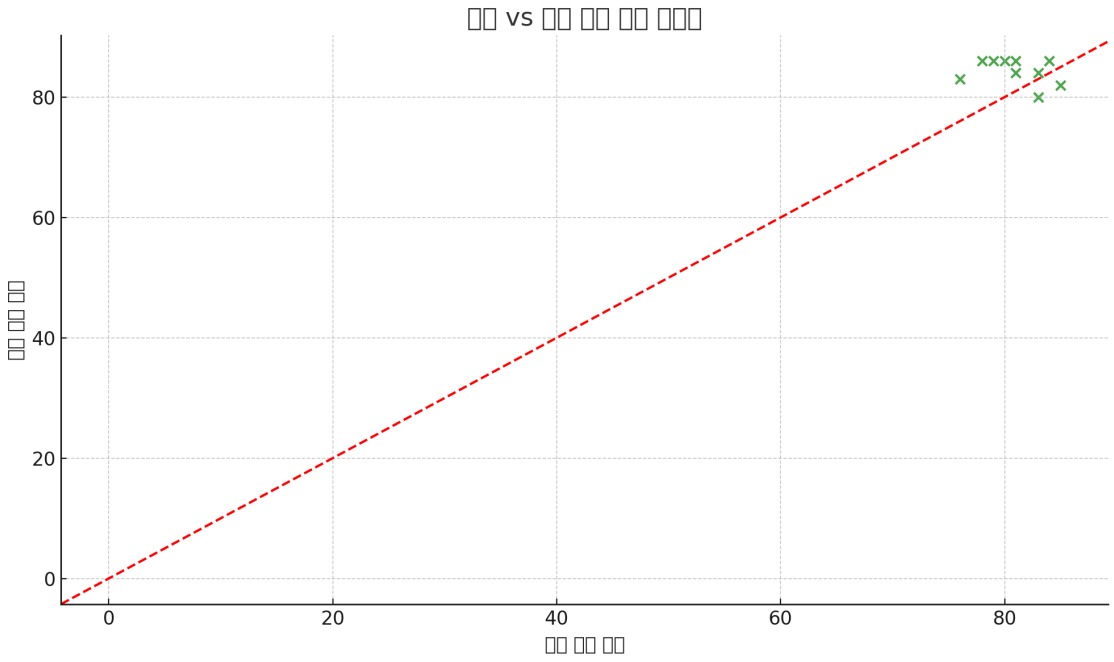
- Data Visualization (motivating and illustrating the comparison of interest)

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Define the data directly in Python (without CSV file)
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]
}

# Step 2: Convert the data into a Pandas DataFrame
df = pd.DataFrame(data)

# Step 3: Generate a Box Plot to compare Initial and Final Health Scores
plt.figure(figsize=(10, 5))
plt.boxplot([df['InitialHealthScore'], df['FinalHealthScore']],␣
 ↪labels=['Initial Health Score', 'Final Health Score'])
plt.title('Comparison of Initial and Final Health Scores')
plt.ylabel('Health Score')
plt.grid(axis='y')
plt.tight_layout()
plt.show()

# Step 4: Generate a Histogram to visualize the distribution of Initial and␣
 ↪Final Health Scores
plt.figure(figsize=(10, 5))
plt.hist(df['InitialHealthScore'], alpha=0.5, label='Initial Health Score',␣
 ↪bins=10, color='blue')
plt.hist(df['FinalHealthScore'], alpha=0.5, label='Final Health Score',␣
 ↪bins=10, color='orange')
plt.title('Distribution of Initial and Final Health Scores')
plt.xlabel('Health Score')
plt.ylabel('Frequency')
plt.legend()
plt.grid(axis='y')
```

```
plt.tight_layout()
plt.show()

# Step 5: Generate a Scatter Plot to show the relationship between Initial and␣
 ↪Final Health Scores
plt.figure(figsize=(10, 5))
plt.scatter(df['InitialHealthScore'], df['FinalHealthScore'], alpha=0.7,␣
 ↪color='green')
plt.title('Scatter Plot of Initial vs Final Health Scores')
plt.xlabel('Initial Health Score')
plt.ylabel('Final Health Score')
plt.grid(True)
plt.axline((0, 0), slope=1, color='red', linestyle='--')  # Adding a diagonal␣
 ↪line for reference
plt.tight_layout()
plt.show()
```







- The box plot shows that the final health scores tend to be slightly higher than the initial health scores in most cases, but the variance is minimal.
- The histogram reveals a general upward shift in the final health scores, with several patients improving their health scores after receiving the vaccine.
- The scatter plot indicates a slight upward trend from initial to final health scores for most patients, though some patients' health scores remained the same or even slightly decreased.

[ ]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Define the data
data = {
    'PatientID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    '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)

# Step 2: Calculate score difference
df['ScoreDifference'] = df['FinalHealthScore'] - df['InitialHealthScore']

# Step 3: Bootstrap function to compute mean difference and confidence intervals
def bootstrap_mean_diff(data, n_bootstrap=1000, random_seed=42):
    np.random.seed(random_seed)
    means = []
    for _ in range(n_bootstrap):
        sample = np.random.choice(data, size=len(data), replace=True)
        means.append(np.mean(sample))
    return np.percentile(means, [2.5, 97.5]), np.mean(means)

# Step 4: Execute bootstrapping
confidence_interval, mean_difference =␣
 ↪bootstrap_mean_diff(df['ScoreDifference'])

print(f"Mean Difference: {mean_difference}")
print(f"95% Confidence Interval: {confidence_interval}")

# Step 5: Visualize the bootstrap distribution
bootstrap_samples = [np.mean(np.random.choice(df['ScoreDifference'],␣
 ↪size=len(df['ScoreDifference']), replace=True)) for _ in range(1000)]

plt.figure(figsize=(10, 6))
plt.hist(bootstrap_samples, bins=30, color='skyblue', edgecolor='black')
plt.axvline(mean_difference, color='red', linestyle='--', label=f'Mean␣
 ↪Difference = {mean_difference:.2f}')
plt.axvline(confidence_interval[0], color='green', linestyle='--',␣
 ↪label=f'Lower CI = {confidence_interval[0]:.2f}')
plt.axvline(confidence_interval[1], color='green', linestyle='--',␣
 ↪label=f'Upper CI = {confidence_interval[1]:.2f}')
plt.title('Bootstrap Distribution of Mean Score Differences')
plt.xlabel('Mean Difference')
plt.ylabel('Frequency')
```

```
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Supporting Visualizations: - Bootstrap Distribution: This histogram illustrates the distribution of the mean differences from the bootstrap samples, showing the central tendency and variability. The red dashed line marks the overall mean difference, and the green dashed lines show the lower and upper bounds of the 95% confidence interval.

Further Considerations - Sample Size: The sample size (10 patients) is quite small, making it difficult to draw robust conclusions. Larger samples would provide more reliable estimates of the vaccine's effect. - Longitudinal Data: The data only reflects the immediate effect of the vaccine on health scores. A follow-up study over a longer period would help assess whether these improvements are sustained. - Control Group: A comparison between a vaccinated group and a placebo/control group would provide stronger evidence for the vaccine's effectiveness. - Potential Confounders: Factors such as patient age, gender, and pre-existing conditions should be considered. These variables might influence health outcomes, and further analysis could investigate whether these factors interact with the vaccine's effect.

Further Instructions: - Randomization: Ensure the random seed is set to guarantee reproducibility when using bootstrap or other randomization techniques. - Report Structure: Provide clear documentation of each step in the analysis, including any assumptions, methods, and justifications for the chosen approach.

[ ]: