Statistical Inference

Minerva University

FA50: Forma Analyses

Prof. Hadavand

December 10, 2023

Introduction[1]

The bigger population this study refers to the entirety of Cornerstone class sessions in 2021-2022. The sample (dataset) examines 30 class sessions from Cornerstone classes in the 2021-2022 academic year. In educational settings, understanding the differences in behavior (engagement during class sessions), if they exist, based on how many absences a student has during class sessions is crucial for enhancing learning outcomes, because measures can be put in place in schools, specifically Minerva, depending on the findings. Minerva can decide to be more strict with absences, less strict, or don't change anything at all, but the policy could potentially be changed and even improved by the end of the study (or we would have an idea of how student behavior would change if their number of absences). Studies have shown that higher attendance generally indicates greater engagement (Credé et al., 2010 & Thomes, 2022) This research aims to analyze if there's a correlation between how many times students miss class and how much they talk during their classes at Cornerstone in 2021-2022.

Dataset (Calculations can be found in Appendix A)

The dataset includes data from Cornerstone classes during the 2021-2022 academic year. It captures various metrics for each class session, including talk time, chat messages, hand raises, etc., gathered automatically from Forum. This paper based on the dataset, consisting of 240 data points (a random sample of 30 class sessions from each Cornerstone for each semester), aims to understand the relationship between student talk time (average seconds per student) and absenteeism.

[1] #organization: I effectively organized my paper by using bolded headings. I organized my code by using the Appendix and made sure my findings were connected.

The variables used for the analysis are:

Variable 1: num_class_absences

- Type: Discrete (represents distinct, separate values)
- Explanation: 'num_class_absences' indicates the count of absences recorded for a specific class session. This variable is discrete as it deals with counting the number of absences, which are whole, distinct values rather than continuous measurements. It tracks the instances when students were absent from the class session.
- Measurement Unit: Count (no specific unit, represents the number of absences)
- Data Collection: The count of absences is counted through an automated system in Forum.

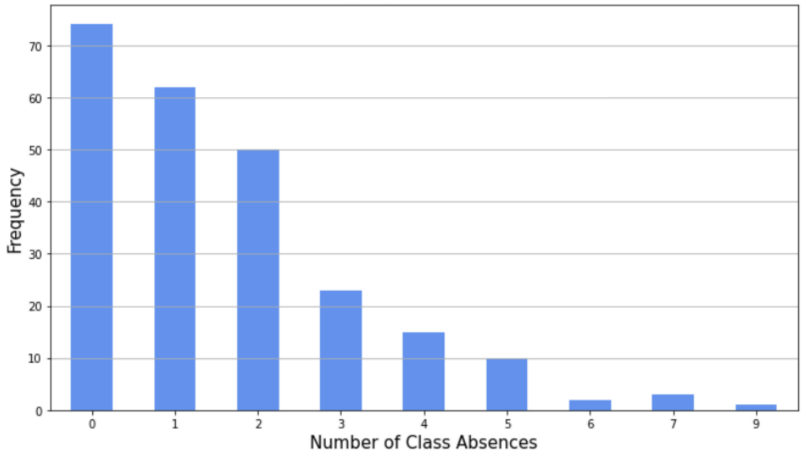

Figure 3: Histogram showing the spread of the number of class absences each student from the sample has

The average number of absences is 1.60. Since the number of class absences is discrete, I chose to approximate the average to 2 and chose it as my benchmark. I separated the sample into two subgroups: low number of absences (<= 2) and high number of absences (> 2) and analyzed

these two subgroups throughout my analysis. Although the two subgroups have a significant difference in the number of entries, I chose the average as opposed to the median because I believe it will provide more significant results (since I am analyzing the difference between multiple absences and a somewhat average or below-average number of absences).

Variable 2: student_talk_time_average

Type: Continuous (can take on a wide range of values within a specified interval)

- Explanation: 'student_talk_time_average' represents the average talk time per student during a class session, measured in seconds. This variable is continuous because it involves measuring time, which can span a range of values and is not restricted to distinct points.
- Measurement Unit: Seconds
- Data Collection: The talk time is recorded through the Forum and it tracks the time each

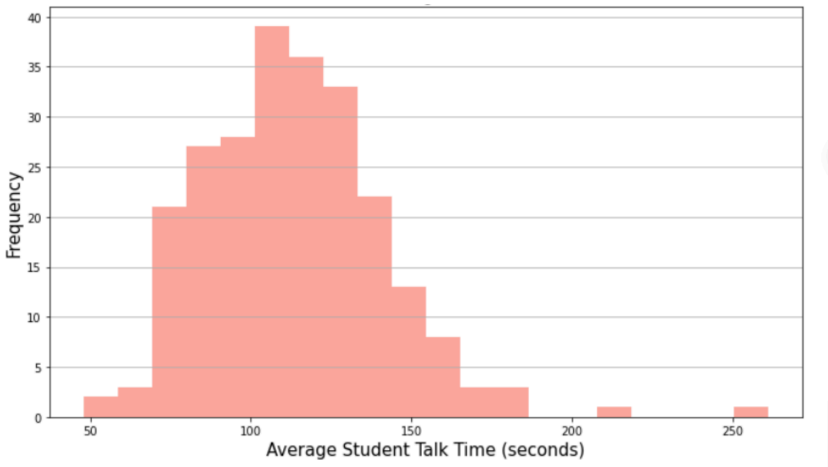

Figure 2: Histogram showing the distribution of average student talk time

For this analysis, I will assume that absence and student talk time are independent variables.student spends talking or engaging in discussions.[2]

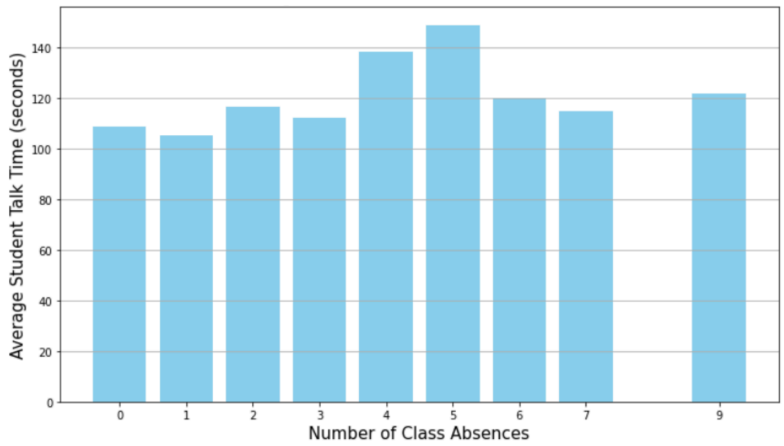

Figure 1: Histogram showing the average student talk time based on class absences

Analysis (Appendix A)

I used Phyton (the libraries: *pandas, numpy, scipy, statsmodels.stats.power*) to perform the calculations. The two hypotheses that I wrote to explore to see if there is a correlation between the variables are:

Null Hypothesis (H0): μ1 =μ2

The average student talk time in Cornerstone classes when the student has a small number of absences (<= 2) is equal to the average talk time when they have a big number of absences (> 2).

Alternative Hypothesis (H1): μ1 !=μ2

[2] #variables: I identified, classified and described my two variables. I detailedhow they are measured and added units. I included a detailed description of them and their roles in my question and understood why they are important for the study (they are the main two variables of the study)

There is a significant difference between the average student talk time in Cornerstone classes when the student has a small number of absences (<=2) compared to when they have a big number of absences (>2).

A significance level ($\alpha = 0.05$) guides the tests. If the calculated p-value is ≤ 0.05, it's considered significant, leading to rejecting the null hypothesis in favor of the alternative (Ha). The choice of α affects Type I and Type II errors: Higher α (e.g., 0.10) increases Type I but reduces Type II errors, while lower α (e.g., 0.01) decreases Type I but raises the chance of Type II errors.

The test is two-tailed because I am interested in seeing any difference whether positive or negative between the variables.

Summary Statistics (Appendix B)

Table 1: Statistics for student talk time average based on the amount of absences a student has (I separated them into low amount of absences which means less or equal to 2 and high amount of absences which means bigger than 2)

	Low amount of absences (<= 2)	High amount of absence (>2)
Count	n1=186	n2=54
Mean	x1=109.8	x2=126.9
Median	110	123.8
Mode	105, 111, 112	118, 73, 110, 114, 140
Standard Deviation	s1=23.9	s2=36.6
Range	127	197

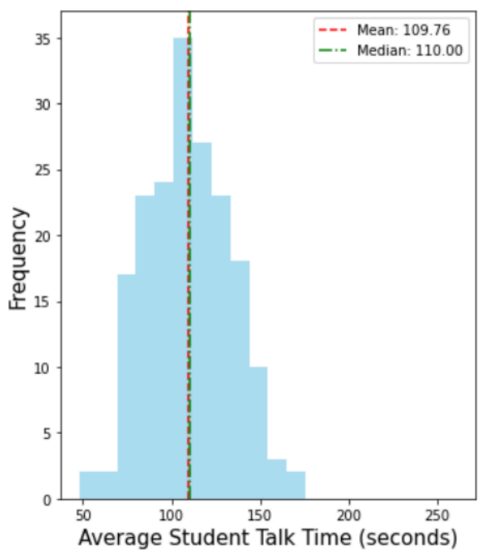

Figure 4: Histogram showing the spread of the number of seconds a student talks for when they have a low amount of absences (less or equal to 20)

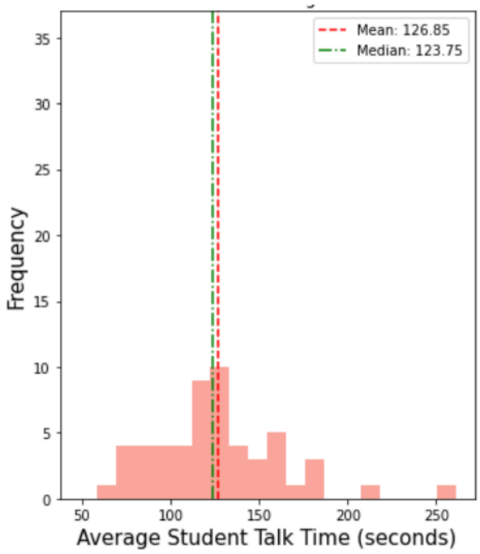

Figure 5: Histogram showing the spread of the number of seconds a student talks for when they have a high amount of absences (bigger than 20)

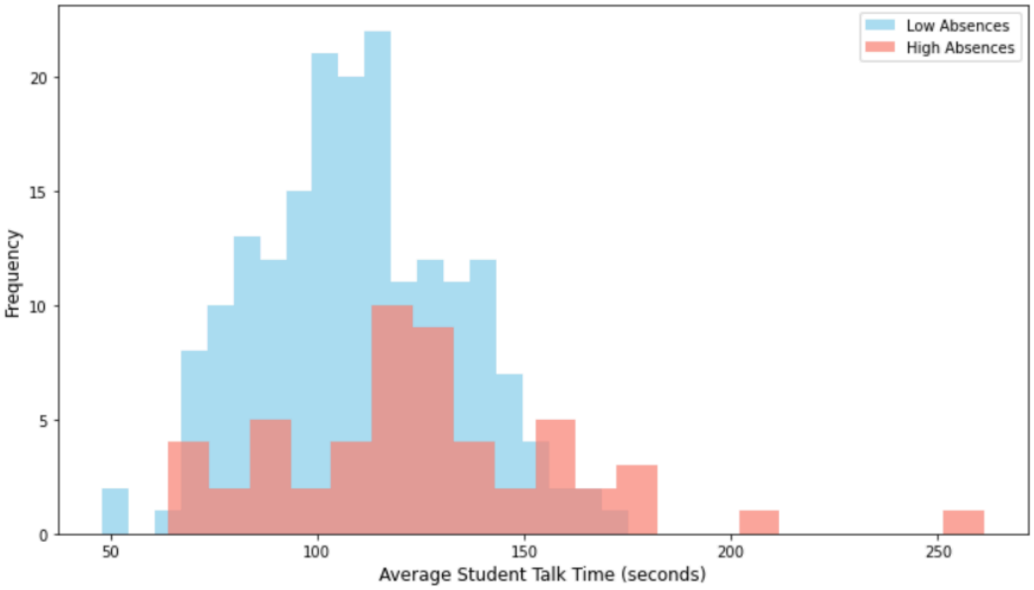

Figure 6: Histogram showing the spread of the number of seconds a student talks for when they have a low amount of absences (less or equal to 2) and high amount of absences (over 2)

I used a histogram because it is the best for visualizing frequency distributions[3]. The histograms display the distribution of the average student talk time for a small amount of absences and a big amount of absences subgroups. The data is almost normally distributed (it is neither significantly skewed or bimodal). When students have 2 or fewer absences in a class, their average talk time seems to range from around 45 to 175. When students have more than 2 absences in a class, their average talk time seems to range from around 60 to 275 (from only looking at the data visualization, I would assume the two values that are above 200 as outliers).

It seems that the students with less absences overall talked less in class than the students with more absences. The first group has a smaller mean and median than the second (as the data spread starts and ends lower). The range for the second group is also bigger than the first. Both distributions show variances, but the first group shows a higher frequency from 90 to 110, whereas the second is from 120 to 130 and the standard deviation for the second group is larger, which further shows that it takes more spread-out values. Based on the visualization and table alone, the first subgroup seems to have generally less talk-time compared to the second one.

There is no need for corrections because the sample size is large enough and I am not conducting multiple statistical tests (if it was the case I would use Bonferroni to control the inflation of the Type 1 error rate).[4]

---

[3] #dataviz: My vizualizations are clear and easy-to-read. I chose the appropriate visualization for the type of data I am showing: histogram which is better for showing frequencies. I included the essential elements of the graph and figure caption that is concise and tells us about what the graph is showing. I have analyzed and interpreted the data vizualization. I chose to remove any background lines for better ease in understanding, chose different colors for both subgroups. I also decided to place the two subgroups' spread one over the other and played with transparency for better comparison.

[4] #descriptivestats: I computed accurate and relevant statistics and included all measures in both my writing and calculations: measures of location, spread and skewness). I explained what insights I gained from the calculations and presented both the mean and mode in the histogram.

Conditions for Inference

I am using T-score because the population standard deviation is unknown (as well as the exact population size) and, for a more accurate estimation, I find it more helpful. It allows for more flexibility when the population parameters are not fully known or reliable.

To proceed with this choice, we must check whether these conditions are satisfied by examining the data displayed in Table 1 and Figures 4, 5, and 6:

- We see that the mean and median values for both groups are relatively close to each other, so the data is not extremely skewed.
- The sample size for both subgroups has at least 30 observations (186 and 54), so the sample size condition is met.
- Each observation in the sample must be independent of each other. Looking at the data set, this appears to be true and, based on the estimation from Table 2, the sample (30) is less than 10% of the estimated population size (2399), which is the benchmark condition.
- The sample is randomly selected. [5]

[5] #distributions: I identified the t-score and explained why aI used it. I applied the right scientific proprieties and evaluated the conditions. I have a single sample from the bigger population. I used the histograms and table above to guide my analysis.

Why?	Step	Lower Bound Estimation	Upper Bound Estimation
I assumed that, since my class has two classes per week per course, this is the case for every class.	Number of classes per week	2 classes	2 classes
I assumed that the semester length hadn't changed over the years.	Number of class sessions per student per semester	14 weeks × 2 classes/week = 28 class sessions	14 weeks × 2 classes/week = 28 class sessions
I assumed that the lower-bound number of students is 170, because I talked with some upper classmates and they told me most classes have had over 170 students in the past 3 years.	Total class sessions for all students per semester	170 students × 28 class sessions = 4760 class sessions	220 students × 28 class sessions = 6160 class sessions I assumed that 220 is the biggest possible number because M27's are the biggest and we had a little over 220 students enrolled.
I assumed Minerva always had 2 semesters	Total class sessions for both semesters	4760 class sessions × 2 = 9520 class sessions	6160 class sessions × 2 = 12320 class sessions
I assumed there have always been 4 Cournerstone classes	Total class sessions for all courses	9520 class sessions × 4 courses/student = 38080 sessions	12320 class sessions × 4 courses/student = 48280 sessions
I assumed that the average number of students per class are 18 because I remember most of my classes (if not all)have 18 students	Total class sessions considering students per class	38080 class sessions ÷ 18 students/class ≈ 2117 sessions	48280 class sessions ÷ 18 students/class ≈ 2683 sessions
	Approximate Geometric Mean: ((2.117 + 2.683)/2)*10^3 = 2399		

Table 2: Estimation using the technique of Estimation by Bounding and Bottom-Up Estimation to find the mean average of class sessions in the 2021-2022 class

Difference of Means Test (Appendix C)

The t-score measures the size of the difference relative to the variation in the sample data. A higher t-score indicates a larger difference between the means of the two groups. The t-score is

3.2364, which is relatively high (bigger than the critical value 0.05). It suggests that the difference between the means of the two groups (low absences and high absences) is statistically significant. The p-value is the probability of observing a result (or a more extreme result) under the assumption that the null hypothesis is true. A lower p-value suggests stronger evidence against the null hypothesis. The p-value obtained (0.0021) is less than the chosen significance level of 0.05 (5%), so there is enough evidence to reject the null hypothesis[6]. A Cohen's d value of 0.6277 indicates a large effect size (medium-to-large). This means the observed difference between the groups is also practically meaningful.[7]

Confidence Intervals (Appendix D)

Confidence interval for the first subgroup: (106.31, 113.21). This means with 95% confidence that the true mean student talk time for the 'Small Number of Absences' group falls within the range of approximately 106.31 to 113.21 seconds. If this study were replicated multiple times, in 95% of these studies, the calculated confidence intervals would contain the true mean of student talk time for the 'Small Number of Absences' group. There's a narrower range compared to the 'Large Number of Absences' group, suggesting more precision in estimating the mean (which makes sense because the sample size of this subgroup, 186, is bigger than the other, 54).

Confidence interval for the second subgroup: (116.86, 136.85). This indicates that with 95% confidence, the true mean student talk time for the “Large Number of Absences' group falls

[6] #probability: I calculated the appropriate probability with clear, detailed steps. I interpreted the application of probabilities on my research question. I ensured my probabilities are correct (sanity check - they are between 0 and 1).

[7] #significance: I correctly stated the null and alternative hypotheses and used symbols to write them. I evaluated why my test is two-tailed. I justified my significance levels nd the risks associated (Type 1 and 2 erros mentioned) I justified my choice of effect size for the practical significance.

within the range of approximately 116.86 to 136.85 seconds. The wider range suggests more variability or uncertainty in estimating the mean student talk time for the 'Large Number of Absences' group compared to the 'Small Number of Absences' group. There's no overlap between the confidence intervals of the two groups, indicating a statistically significant difference in the mean student talk time between 'Small Number of Absences' and 'Large Number of Absences' groups which means that there's a statistically significant difference in student talk time between the two groups. This is in agreement with the results of the p-value.[8]

Power (Appendix D)

The analysis showed a high statistical power of 0.981 so the study has a strong ability to detect a true difference between groups if it exists. It implies that in nearly 98.1% of cases where there is a genuine effect, the statistical test employed in the study is likely to correctly reject the null hypothesis (H0) and identify the difference as statistically significant. A high statistical power corresponds to a low probability of committing a Type II error (false negative). In other words, the chance of failing to detect a real effect when it's actually present is low. A power of 0.981 also may suggest that the study's sample size is likely sufficient to detect a meaningful difference between groups.

Results and Conclusions

The analysis assumes independence between absence frequency and student talk time, but potential correlation or causation might exist due to unexplored factors. The chosen significance level ($\alpha = 0.05$) impacts errors in the statistical tests. Limitations could include unconfirmed causation, limited variables considered, potential sample size constraints, Forum data capturing

[8] #confidenceintervals: I constructed the confidence intervals and explained which term is what. I interpreted the intervals and hinted at how it is important for my work. I explained the meaning of all of the terms , including the confidence level.

limitations, and the need for thorough validation of statistical assumptions. The study does not encompass broader contextual factor.

This said, the analysis provides strong evidence that there is a notable and statistically significant difference (p-value=0.0021, d=0.627) in student talk time between those with low and high absences in Cornerstone classes in 2021-2022. Overall, there is an increase in student talk time when the student has more absences.

If the sample size is big enough, the population is not heavily skewed, the sample size is less than 10% and the sample was randomly selected, based on this small sample of the population, the results are true for the whole population.[9]

Word Count: 1820[10]

[9] #induction: the application is in the realm of inductive reasoning and not deductive because they aren't properly observable. This argument aligns with inductive reasoning, particularly enumerative induction. It generalizes from specific observations (the sample) to make a broader conclusion about the entire population. Induction does not guarantee certainty. The strength and reliability of this argument depend on several factors.

[10] #professionalism: I followed all of the requirements and guidelines and presented my work in a professional manner.

References

Credé, M., Roch, S. G., & Kieszczynka, U. M. (2010). Class Attendance in College: A Meta-Analytic Review of the Relationship of Class Attendance With Grades and Student Characteristics. *Review of Educational Research*, *80*(2), 272–295. https://doi.org/10.3102/0034654310362998

Thomes, J. J. (2022, May 9). *Relationships Between Undergraduate Student Performance, Engagement, and Attendance in an Online Environment*. Frontiers. https://www.frontiersin.org/articles/10.3389/feduc.2022.906601/full

Reflection

To confirm the accuracy of t-scores, confidence intervals, and p-values, I employed cross-validation. This involved comparing outcomes across various statistical software. By conducting analyses in both Python and Excel, and ensuring consistency in computed values, I ensured the reliability and accuracy of the statistical results obtained. I also calculated by hand the easy to calculate values like mean, mode, etc. I also computed both by using Python predefined functions or not.

The feedback I got for #estimation in the last assignment highlighted the need to improve how I explain my estimation process. It advised me to rely less on medical literature and focus more on my own thinking and knowledge. So, I adjusted my approach by adding more of my own thoughts and understanding into the estimation process. This change helped me create a stronger and more customized analysis. I was also advised to look at the self-guided question, which helped me with choosing to use two different types of estimation - Bound Estimation and Bottom-Up Estimation.

Acknowledgements

Mulyn, Brady, Marta, Cristian helped me when I was in need and I did not understand the class material. I used websites like Study.com and Statistics How To when I needed help. I also used Professor’s Hadavand in-class and Slack advice and my TA’s, Enes, feedback. I only used Grammarly to make my sentences more cohesive. I adapted the code from the following classes: Session 21, Session 24. This feedback and help allowed me to deeply understand the course material and complete this assignment in-depth by myself and how to interpret my findings.

Appendix A: Import and Analyze Data

Code Cell 1 of 26

In [6]

```
# This line imports the Pandas library and names it 'pd' for convenience.
import pandas as pd

# This line stores the URL where the data is located in a variable called 'data_url'.
data_url = "https://course-resources.minerva.edu/uploaded_files/mu/00331768-5264/cornerstone-metrics-data-by-class-2021-2022--sample.csv"

# This line reads the data from the CSV file located at the URL and creates a DataFrame called 'df' to store it.
df = pd.read_csv(data_url)

# This line displays the first 10 rows of the DataFrame 'df' to show a preview of the data.
df.head(10)
```

Run Code Reset to Original Submission

Out [6]

	semester	course	course_id	section	section_id	class	class_id	num_
0	Fall 2021	Formal Analyses	1947	MW@09:00AM San Francisco	7754	CS50 Session 7 - (4.1) Fallacy Detection	57536	19
1	Fall 2021	Formal Analyses	1947	TTh@05:00PM San Francisco	7857	CS50 Session 15 - (8.2) Distributions of Discr...	58234	20
2	Fall 2021	Formal Analyses	1947	TTh@11:00AM San Francisco	7755	CS50 Session 24 - (13.2) Difference of Means T...	59700	18
3	Fall 2021	Formal Analyses	1947	MW@07:00AM San Francisco	7792	CS50 Session 1 - (1.1) Critical Thinking	56195	18
	Fall	Formal	1947	MW@07:00AM	7792	CS50 Session	57227	18

In [51]:

```
import pandas as pd
# This line imports the Pandas library and names it 'pd' for convenience.

# Assuming 'low_absences' and 'high_absences' are Series containing the data for low and high absences respectively.
# Assuming 'low_absences' and 'high_absences' are previously defined subsets of your data
low_absences = df[df['num_class_absences'] <= 2]['student_talk_time_average']
high_absences = df[df['num_class_absences'] > 2]['student_talk_time_average']

# Calculate the number of entries in each category
num_entries_low_absences = len(low_absences)
num_entries_high_absences = len(high_absences)

# Display the counts
print("Number of entries for low absences:", num_entries_low_absences)
print("Number of entries for high absences:", num_entries_high_absences)

# This line reads the data from the CSV file located at the URL and creates a DataFrame called 'df' to store it.
df = pd.read_csv(data_url)
# Get the absences and talk time data from the DataFrame and save them.
absences = df["num_class_absences"]
talk_time = df["student_talk_time_average"]

# Separating data into low and high amounts of absences based on specific criteria.
low_absences = df[df["num_class_absences"] <= 2]["student_talk_time_average"]
high_absences = df[df["num_class_absences"] > 2]["student_talk_time_average"]

# Function: Calculate the average of a group of values.
def calculate_average(data):
    total = sum(data)  # Sum up all the values.
    count = len(data)  # Count how many values there are (length of the data set).

    # Calculate the average by dividing the total sum by the number of values and give the result back.
    return total / count

# Calculate the average talk time for low and high absence groups.
avg_low_absences = calculate_average(low_absences)
avg_high_absences = calculate_average(high_absences)


no_absences= calculate_average(absences)
# Show the average talk times on the screen.
print("Average talk time for low absences (2 or less):", avg_low_absences)
print("Average talk time for high absences (more than 2):", avg_high_absences)
print("absences", no_absences)

```

Run Code Reset to Original Submission

Out [51]:

```
Number of entries for low absences: 186
Number of entries for high absences: 54
Average talk time for low absences (2 or less): 109.76075268817205
Average talk time for high absences (more than 2): 126.85185185185185
absences 1.5958333333333334
```

Code Cell 4 of 26

In [52]

```
# Function: Calculate the middle value (median) of a dataset.
def calculate_mid_value(data):
    if len(data) > 0:  # Check if there's data available.
        data_list = sorted(data)  # Sort the data in ascending order.
        n = len(data_list)  # Find the number of elements in the sorted list.
        if n % 2 == 0:  # If the number of elements is even:
            mid_val = (data_list[n // 2 - 1] + data_list[n // 2]) / 2  # Calculate the median.
            # 'data_list[n // 2 - 1]' and 'data_list[n // 2]' represent the two middle values in a sorted dataset.
            # For even-length datasets, the median is the average of these two middle values.
        else:  # If the number of elements is odd:
            mid_val = data_list[n // 2]  # The median is the middle value.
        return mid_val  # Return the calculated median value.
    else:  # If there's no data, return None.
        return None

# Calculate the median of absences and talk time data.
median_val1 = calculate_mid_value(low_absences)  # Calculate the median of the 'num_class_absences'.
median_val2 = calculate_mid_value(high_absences)  # Calculate the median of the 'student_talk_time_average'.
no_absences=calculate_mid_value(absences)
print("Median for absences:", median_val1)  # Display the calculated median value for absences.
print("Median for talk time:", median_val2)  # Display the calculated median value for talk time.

```

Run Code Reset to Original Submission

Out [52]

```
Median for absences: 110.0
Median for talk time: 123.75
absences 1.0
```

Code Cell 5 of 26

In [53]

```
from collections import Counter  # Import Counter from the collections module

# Function: Calculate the mode(s) of a dataset.
def calculate_modal(data):
    if len(data) > 0:  # Check if there's data available.
        counts = Counter(data)  # Count occurrences of each value in the dataset.
        modes = [k for k, v in counts.items() if v == max(counts.values())]
        # Create a list called 'modes' that contains items (keys) from the 'counts' dictionary.
        # Include only items where their count (value) matches the maximum count in the 'counts' dictionary.

        return modes  # Return the mode(s).
    else:  # If there's no data, return None.
        return None

# Calculate the mode(s) for subsets of low and high absences.

modes_low_absences = calculate_modal(low_absences)  # Calculate the mode(s) of the 'student_talk_time_average' for low absence group.
modes_high_absences = calculate_modal(high_absences)  # Calculate the mode(s) of the 'student_talk_time_average' for high absence group.

print("Mode for low absences:", modes_low_absences)  # Display the calculated mode(s) for low absences.
print("Mode for high absences:", modes_high_absences)  # Display the calculated mode(s) for high absences.

```

Run Code Reset to Original Submission

Out [53]

```
Mode for low absences: [105.0, 111.0, 112.0]
Mode for high absences: [118.0, 73.0, 110.0, 114.0, 140.0]
```

Code Cell 6 of 26

In [54]

```
# Function: Calculate the range of a dataset.
def calculate_data_range(data):
    if len(data) > 0:  # Check if there's data available.
        data_range = max(data) - min(data)  # Find the range by subtracting the smallest value from the largest.
        return data_range  # Return the calculated range.
    else:  # If there's no data, return None.
        return None

# Calculate the range of subsets related to low and high absences.
range_low_absences = calculate_data_range(low_absences)  # Calculate the range of 'student_talk_time_average' for low absence group.
range_high_absences = calculate_data_range(high_absences)  # Calculate the range of 'student_talk_time_average' for high absence group.

print("Range for low absences:", range_low_absences)  # Display the calculated range for low absences.
print("Range for high absences:", range_high_absences)  # Display the calculated range for high absences.

```

Run Code Reset to Original Submission

Out [54]

```
Range for low absences: 127.0
Range for high absences: 197.0
```

In [57]

```
# Function: Calculate the standard deviation of a dataset.
def calculate_std_dev(data):
    """
    Calculates the standard deviation of a given dataset.

    Parameters:
    data (list): The dataset for which the standard deviation needs to be calculated.

    Returns:
    float or None: The standard deviation of the dataset if data is available, otherwise returns None.
    """
    if len(data) > 0:
        # Calculate the mean of the dataset.
        mean_value = calculate_average(data)
        # Find the number of elements in the dataset.
        n = len(data)
        # For each element 'x' in the dataset 'data', calculate the squared difference between that element and the mean value.
        squared_diffs = [(x - mean_value) ** 2 for x in data]
        # Calculate the variance.
        variance_val = sum(squared_diffs) / (n - 1)
        # Calculate the standard deviation from the variance.
        std_dev = variance_val ** 0.5
        return std_dev
    else:
        # If there's no data, return None.
        return None

# Calculate the standard deviation of different datasets.
std_deviation_val = calculate_std_dev(low_absences)
std_deviation_val1 = calculate_std_dev(high_absences)

```

```
31
32 # Display the calculated standard deviation for each dataset.
33 print("Standard Deviation (Low Absences):", std_deviation_val)
34 print("Standard Deviation (High Absences):", std_deviation_val1)
35
```

Run Code Reset to Original Submission

```
Out [57]  Standard Deviation (Low Absences): 23.86981623301624
          Standard Deviation (High Absences): 36.61279156169197
```

Appendix B: Visualize Data

In [8]:

```
import matplotlib.pyplot as plt
import numpy as np

# Extracting relevant columns 'num_class_absences' and 'student_talk_time_average' from the Cornerstone
absences_talk_time = df[['num_class_absences', 'student_talk_time_average']]

# Calculating average talk time for each count of class absences
absence_counts = absences_talk_time['num_class_absences'].unique()
avg_talk_times = [absences_talk_time[absences_talk_time['num_class_absences'] == count]['student_talk_t

# Creating a bar chart to visualize the average student talk time for different counts of class absence
plt.figure(figsize=(10, 6))
plt.bar(absence_counts, avg_talk_times, color='skyblue')
plt.title('Average Student Talk Time based on Class Absences')
plt.xlabel('Number of Class Absences', fontsize=15)
plt.ylabel('Average Student Talk Time (seconds)', fontsize=15)
plt.xticks(absence_counts)  # Ensure all absence counts are displayed on the x-axis
plt.grid(axis='y')  # Adding horizontal gridlines for better readability
plt.tight_layout()

plt.show()
# Displaying the bar chart

# Extracting the 'student_talk_time_average' data from the Cornerstone dataset
student_talk_time = df['student_talk_time_average']

# Creating a histogram to visualize the distribution of average student talk time
plt.figure(figsize=(10, 6))
plt.hist(student_talk_time, bins=20, color='salmon', alpha=0.7)
plt.title('Distribution of Average Student Talk Time')
plt.xlabel('Average Student Talk Time (seconds)', fontsize=15)
plt.ylabel('Frequency', fontsize=15)
plt.grid(axis='y')  # Adding horizontal gridlines for better readability
plt.tight_layout()
```

```
plt.show()

# Extracting the 'num_class_absences' data from the Cornerstone dataset
absence_counts = df['num_class_absences']

# Counting the frequency of each unique count of class absences
absence_frequency = absence_counts.value_counts().sort_index()

# Creating a bar chart to visualize the frequency of different counts of class absences
plt.figure(figsize=(10, 6))
absence_frequency.plot(kind='bar', color='cornflowerblue')
plt.title('Frequency of Class Absences')
plt.xlabel('Number of Class Absences', fontsize=15)
plt.ylabel('Frequency', fontsize=15)
plt.xticks(rotation=0)  # Ensure all absence counts are displayed on the x-axis without rotation
plt.grid(axis='y')  # Adding horizontal gridlines for better readability
plt.tight_layout()

plt.show()
# Displaying the bar chart

# Filter the dataset for low and high amounts of absences
low_absences = df[df['num_class_absences'] <= 2]['student_talk_time_average']
high_absences = df[df['num_class_absences'] > 2]['student_talk_time_average']
import matplotlib.pyplot as plt
import numpy as np

# Filter the dataset for low and high amounts of absences
low_absences = df[df['num_class_absences'] <= 2]['student_talk_time_average']
high_absences = df[df['num_class_absences'] > 2]['student_talk_time_average']

# Create histograms for the two categories
plt.figure(figsize=(10, 6))
```

```
combined_data_low = np.array(low_absences)  # Convert data from 'low_absences' into a NumPy array
combined_data_high = np.array(high_absences)  # Convert data from 'high_absences' into a NumPy array

# Concatenate the data from 'low_absences' and 'high_absences' into a single array and determine bin ed
bin_edges = np.histogram_bin_edges(
    np.concatenate((combined_data_low, combined_data_high)),  # Concatenate both arrays into a single d
    bins=20  # Define the number of bins for the histogram (in this case, 20 bins)
)


plt.subplot(1, 2, 1)
plt.hist(combined_data_low, bins=bin_edges, color='skyblue', alpha=0.7)
plt.title('Student Talk Time - Low Absences')
plt.xlabel('Average Student Talk Time (seconds)', fontsize=15)
plt.ylabel('Frequency', fontsize=15)
plt.grid(axis='y')
plt.grid(False)
plt.ylim(0, 37)

# Calculate mean and median for low absences
mean_low = np.mean(combined_data_low)
median_low = np.median(combined_data_low)

# Plot vertical lines for mean and median on the histogram for low absences
plt.axvline(x=mean_low, color='red', linestyle='--', label=f'Mean: {mean_low:.2f}')
plt.axvline(x=median_low, color='green', linestyle='-.', label=f'Median: {median_low:.2f}')
plt.legend()

plt.tight_layout()

plt.subplot(1, 2, 2)
plt.hist(combined_data_high, bins=bin_edges, color='salmon', alpha=0.7)
plt.title('Student Talk Time - High Absences')
plt.xlabel('Average Student Talk Time (seconds)', fontsize=15)
plt.ylabel('Frequency', fontsize=15)
```

```
plt.grid(axis='y')
plt.tight_layout()
plt.grid(False)
plt.ylim(0, 37)

# Calculate mean and median for high absences
mean_high = np.mean(combined_data_high)
median_high = np.median(combined_data_high)

# Plot vertical lines for mean and median on the histogram for high absences
plt.axvline(x=mean_high, color='red', linestyle='--', label=f'Mean: {mean_high:.2f}')
plt.axvline(x=median_high, color='green', linestyle='-.', label=f'Median: {median_high:.2f}')
plt.legend()

plt.show()


import numpy as np

# Filter the dataset for low and high amounts of absences
low_absences = df[df['num_class_absences'] <= 2]['student_talk_time_average']
high_absences = df[df['num_class_absences'] > 2]['student_talk_time_average']

# Create histograms for the two categories stacked vertically
plt.figure(figsize=(10, 6))

plt.hist(low_absences, bins=20, color='skyblue', alpha=0.7, label='Low Absences')
plt.hist(high_absences, bins=20, color='salmon', alpha=0.7, label='High Absences')
plt.title('Student Talk Time Distribution based on Absences')
plt.xlabel('Average Student Talk Time (seconds)', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.legend()
plt.grid(axis='y')
plt.grid(False)
plt.tight_layout()
```

Appendix C: Difference of Means Test

In [9]:
```
import numpy as np  # Importing NumPy for numerical operations
from scipy import stats  # Importing stats module from SciPy for statistical functions

def difference_of_means(data_group1, data_group2, tails):
    n1 = len(data_group1)  # Calculating the sample size of group 1
    n2 = len(data_group2)  # Calculating the sample size of group 2

    x1 = np.mean(data_group1)  # Calculating the mean of data_group1
    x2 = np.mean(data_group2)  # Calculating the mean of data_group2

    s1 = np.std(data_group1, ddof=1)  # Calculating the sample standard deviation of data_group1
    s2 = np.std(data_group2, ddof=1)  # Calculating the sample standard deviation of data_group2

    # Calculating the standard error of the difference between means
    standard_error = np.sqrt((s1**2 / n1) + (s2**2 / n2))

    # Calculating the t-score for the difference between means
    t_score = np.abs((x2 - x1)) / standard_error

    # Calculating degrees of freedom for the t-distribution
    df = min(n1, n2) - 1

    # Calculating the p-value using the t-distribution
    p_value = tails * stats.t.cdf(-t_score, df)

    # Calculating Cohen's d, a measure of effect size
    sd_pooled = np.sqrt(((s1**2 * (n1 - 1)) + (s2**2 * (n2 - 1))) / (n1 + n2 - 2))
    cohens_d = (x2 - x1) / sd_pooled

    # Printing the calculated statistical values
    print('T-score =', t_score)
    print('p-value =', p_value)
    print('Cohen\'s d =', cohens_d)
    print('Standard Error =', standard_error)
    print('Degrees of Freedom =', df)
    print('Pooled Standard Deviation =', sd_pooled)

# Call the function with your specific subgroup variables and a 2-tailed test
difference_of_means(low_absences, high_absences, 2)
```

```
T-score = 3.236435261180288
p-value = 0.002088619615703515
Cohen's d = 0.6276874341650995
Standard Error = 5.280840735076804
Degrees of Freedom = 53
Pooled Standard Deviation = 27.228678213724375
```

Appendix D: Confidence Interval

In [10]:
```
import numpy as np  # Importing NumPy for numerical operations
from scipy import stats  # Importing stats module from SciPy for statistical functions

def confidence_interval(point_estimate, standard_error, confidence_level, df):
    t = stats.t.ppf(1 - (1 - confidence_level) / 2, df)  # Calculating the critical t-value for the con
    lowbound = point_estimate - t * standard_error  # Calculating the lower bound of the confidence int
    highbound = point_estimate + t * standard_error  # Calculating the upper bound of the confidence in
    return lowbound, highbound  # Returning the confidence interval instead of printing it

print("Confidence Interval for Low Absences")
# Calculating and printing the confidence interval for low absences
low_absences_interval = confidence_interval(
    np.mean(low_absences),  # Point estimate - Mean of low absences
    np.std(low_absences, ddof=1) / np.sqrt(len(low_absences)),  # Standard error
    0.95,  # Confidence level
    len(low_absences) - 1  # Degrees of freedom
)
print("Confidence Interval:", low_absences_interval)

print("\nConfidence Interval for High Absences")
# Calculating and printing the confidence interval for high absences
high_absences_interval = confidence_interval(
    np.mean(high_absences),  # Point estimate - Mean of high absences
    np.std(high_absences, ddof=1) / np.sqrt(len(high_absences)),  # Standard error
    0.95,  # Confidence level
    len(high_absences) - 1  # Degrees of freedom
)
print("Confidence Interval:", high_absences_interval)
```

```
Confidence Interval for Low Absences
Confidence Interval: (106.30779656000861, 113.21370881633548)

Confidence Interval for High Absences
Confidence Interval: (116.85848347742092, 136.84522022628278)
```

Appendix E: Power

In [11]:
```
import statsmodels.stats.power as smp  # Importing statsmodels for power analysis

effect_size = 0.6277  # Effect size - measure of the strength of a phenomenon
alpha = 0.05  # Significance level - probability of rejecting the null hypothesis when it's true
n1 = num_entries_low_absences  # Sample size of group 1
n2 = num_entries_high_absences  # Sample size of group 2
ratio = n2 / n1  # Ratio of sample sizes (if unequal)

# Calculating statistical power for a two-sample t-test
power = smp.TTestIndPower().solve_power(
    effect_size=effect_size,  # Effect size
    nobs1=n1,  # Sample size of group 1
    alpha=alpha,  # Significance level
    ratio=ratio,  # Ratio of sample sizes (if unequal)
    alternative='two-sided'  # Two-sided alternative hypothesis
)

print(f"Statistical Power: {power}")  # Printing the calculated statistical power
```

```
Statistical Power: 0.9814329899487342
```