Statistical Inference Assignment

Minerva Schools at KGI

CS50: Formal Analyses

Prof. J. Wilkins

December 10, 2020

Statistical Inference

Introduction

Having experience in music composing, I have analyzed melodies according to the music theory principles. Now, Spotify data about numerous characteristics of music pieces allows for studying the correlation between these features and testing it using statistics. In particular, this paper will investigate how well we can predict a song's danceability based on its valence. We will explore these two characteristics using descriptive statistics and distributions, construct a confidence interval, and conduct a statistical and practical significance test to conclude whether valence affects a song's danceability.

Dataset

After processing the data from Kaggle, Minerva Schools at KGI provided us with this dataset about over 30,000 tracks on Spotify analyzed from thirteen audio features: acousticness, danceability, valence, energy, instrumentalness, tempo, loudness, speechiness, liveness, key, mode, duration, and time signature. Each song's audio features were extracted using the Spotify Web API and the Python library (Tamer, 2017). The data can be found here.

We want to analyze these data considering two specific audio features - valence and danceability. According to the Spotify Web API developer guide definitions, valence describes the musical positiveness conveyed by a track; tracks with high valence sound more positive (e.g., happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g., sad, depressed, angry). Danceability characterizes how suitable a track is for dancing based on a combination of musical elements, including tempo, rhythm stability, beat strength, and overall

regularity. So, valence and danceability are the variables to analyze. Valence is a quantitative discrete variable because we can count the value to the nearest thousandth. It is dependent because the positivity or negativity of the sound depends on the key (pitch) the composer used to write the song - C-major will result in a positive valence while e-minor will give a negative valence. However, since we are analyzing only two variables in this context, we may assume valence is independent. Danceability is a quantitative discrete variable as well because we can count the value to the nearest thousandth. We cannot state whether the variable is dependent or independent until we test our null and alternative hypotheses. If we reject our null hypothesis that valence does not affect danceability, then the alternative is likely to be true. Therefore, danceability may be a dependent variable if we see the difference between the valence and the danceability values when changed.[1]

Analysis

Summary Statistics

We read the dataset in Python using the pandas' package to analyze the descriptive statistics. Table 1 shows the summary of these analyses (these are computed in Appendix A). Figures 1 and 2 provide the sample distributions for each variable (these are constructed in Appendix B). To find the right number of bins for the histogram, we will use the Sturges method:

$$n = 3.3 \times lg\, 169,909 + 1 \approx 18.25971$$

Therefore, the optimal number of bins to create a histogram for the sample size of 169,909 is 18.

[1] **#variables** - I accurately identified and classified the variables of the system - valence and danceability. I stated the types of the variables and justified my choice. Also, I analyzed whether they are dependent or independent, and provided evidence for my choices. Though, we need to remember that the classification of "dependent" depends on how we are framing our analysis, which I am considering in the paragraph. I show how these variables are relevant to my hypotheses.

Table 1: Summary statistics for the variables of interest		
	Danceability	Valence
Count	$n = 169,909$	$n = 169,909$
Mean	$\overline{x}_{dnc} = 0.538$	$\overline{x}_{val} = 0.532$
Standard deviation	$s_{dnc} = 0.175$	$s_{val} = 0.262$
Median	0.548	0.544
Mode	0.565	0.961
Range	0.988 - 0 = 0.988	1 - 0 = 1

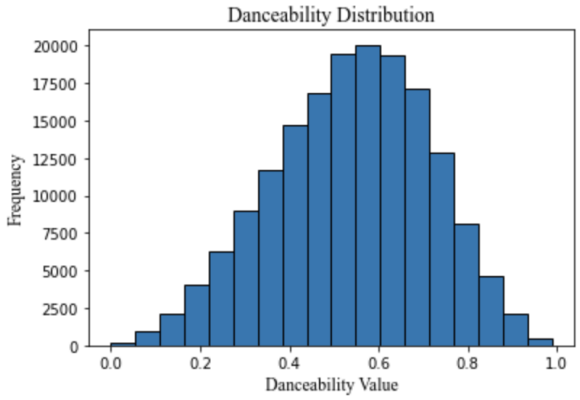


Figure 1. Histogram for Danceability

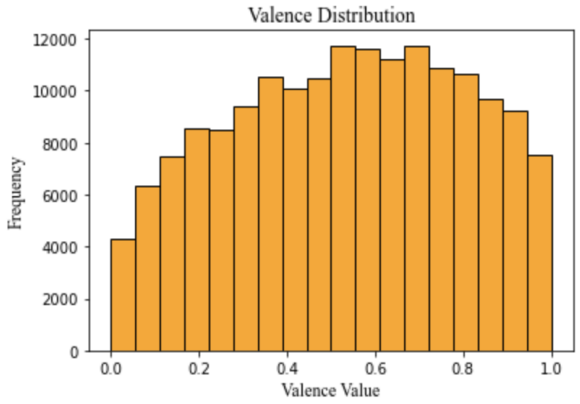


Figure 2. Histogram for Valence[2]

Based on the descriptive statistics and the corresponding histogram for valence and danceability, we compare the values of each quantity. The danceability distribution is approximately normal - it has a nice bell-shaped curve and is symmetrical. Thus, we may infer that the median, mode and mean are approximately equal. The calculations confirm this. The

[2] **#dataviz** - I created understandable histograms of tempo and danceability. I put a title, labeled the x and y-axis, and wrote a description for it. I used bright colors for my bins to make it easier for one to look at and I calculated the number of bins using the Sturges method.

range of the values falls between 0 and 0.988, where a value of 0 is the least danceable and 0.988 is the most danceable.

We can describe the valence distribution as a Johnson distribution of a family S_B. S-bounded means that the values on the left and right are fixed, unlike in normal distribution where they are approaching infinity. The histogram is asymmetrical and unimodal - it has one peak. The median and the mean are approximately equal - 0.544 and 0.532, respectively; however, the mode is different - 0.961. It means that the biggest number of songs have such a high valence. The range falls from 0 to 1, where 0 describes the least positive song, and 1 is the most positive.[3]

Since the danceability distribution is normally distributed, its standard deviation is smaller than the valence's one. The calculations confirm this: $s_{dnc} = 0.175 < s_{val} = 0.262$.[4]

Difference of Means Test

To conduct a hypothesis test, we will use the difference of the means significance test. First, we will state the null and the alternative hypotheses to be clear about them.

1. H_0: valence does not affect danceability. $\mu = 0.532$
2. H_A: valence affects the danceability of a song . $\mu \neq 0.532$

This will be a two-sided test since we are going to reject the null hypothesis if there is a difference in danceability when changing the valence value in any direction. We set the

[3] **#distribution** - I accurately identified appropriate distributions in the given context and justified the choice - a normal and a Johnson distribution. I described the main characteristics of the distributions. Later, I applied the Central Limit Theorem to explain why the normal condition for the sample was met, since our sample size was larger than 30.

[4] **#descriptivestats** - I chose appropriate statistics to analyze the data. Using Python, I accurately calculated the values for the mean, mode, median, standard deviation, and range with a clear algorithm. I provided a well-justified interpretation of the statistics and connected the analysis to a visual representation of the distribution.

significance level to the default: $\alpha = 0.05$. Before conducting the test, we will check whether the conditions are met:

- Because the songs' vary in data from the early twentieth century to the twenty-first century, we can state the sample is random.
- There are 97 million songs in the world, thus our sample size is less than 10% of the population. The independence condition is satisfied.
- Since our sample size is greater than 30, then we can apply the Central Limit Theorem - the distribution of multiple sample means is approximately normal.

Next, using Python, we divide our valence data into two parts: first, where the valence is less than 0.5, second, the valence is more than 0.5 (Appendix C). Thus, we have two independent samples to conduct a difference of the means test. In other words, we filtered out songs into two groups according to their valence and examined the danceability of these data (Table 2).

Table 2: Summary statistics for the danceability for two sample groups with valence more than 0.5 and less than 0.5.

	More than 0.5	Less than 0.5
Count	$n_1 = 94079$	$n_2 = 75640$
Mean	$\overline{x}_1 = 0.61$	$\overline{x}_2 = 0.44$
Standard deviation	$s_1 = 0.14$	$s_1 = 0.17$

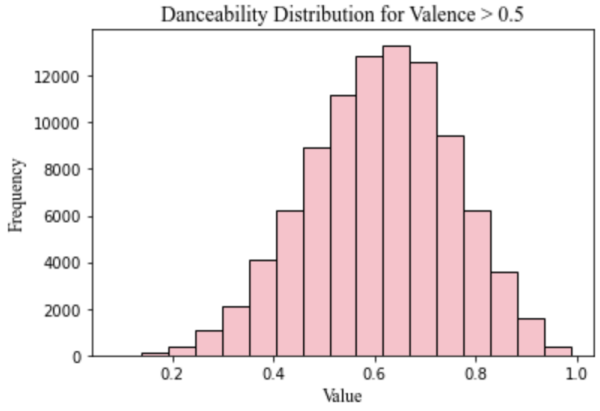


Figure 3. Histogram for Danceability with Valence more than 0.5

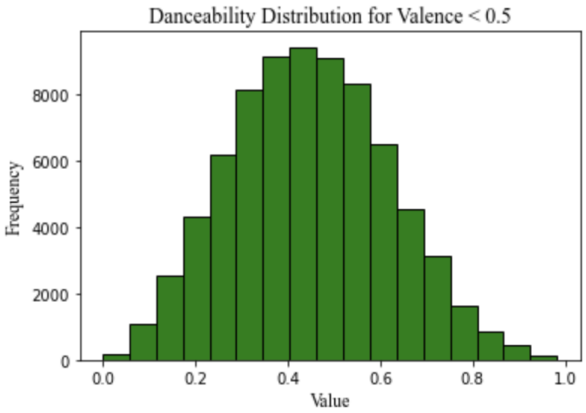


Figure 4. Histogram for Danceability with Valence less than 0.5

Using sample data, we find the standard error, degrees of freedom, test statistic, and the P-value associated with the test statistic. Again, we check the holding of the next conditions: samples are random, independent, the sample size is greater than 30 - the distributions are approximately normal (Figure 3 and 4). Both groups meet these settings.

To assess statistical significance, we compute the P-value . To do so, we first compute the T-score using the usual formula for a difference of means test, $T = \frac{\bar{x}_2 - \bar{x}_1}{SE}$ with $SE = \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$. We take a conservative estimate for the degrees of freedom of the t-distribution as df = 75639. See Appendix D for the calculation in Python. The T-score of 210.88 results in a two-tailed p-value of approximately 0.0. Because the value is so small, Python resulted in displaying just 0.0. The value is much less than the significance level, 0.0 < 0.05. Thus, we reject the null hypothesis in favor of the alternative hypothesis. This p-value is extremely statistically significant, so it is likely not caused by chance for a given statistical significance level.

To assess practical significance, we need to measure an effect size. We calculate Hedge's

g as our measure because it corrects the upward bias of Cohen's d; it pools using n - 1 for each sample instead of n, which provides a better estimate. Though the Hedge's g is also positively biased, we may neglect it since the sample size is very large. Here, the Hedge's g ≈ -1.05. It is very large and, in the context of this dataset, it is practically significant. There is a strong relationship between the two variables: valence and danceability. Thus, we should care about how valence affects danceability.[5]

Confidence Interval

To estimate the average danceability mean, we compute a confidence interval for the difference of the means of danceability values with the valence less and more than 0.5. The confidence interval is useful because we do not have data from the whole population, but it provides a plausible range of values for this quantity. This ability to generalize the findings is beneficial since we want to apply our research to all songs.

To calculate the confidence interval, we use the best estimate of the margin of error, using t score and standard error, and the difference of the means. Because our degrees of freedom value is so large, the t score is approximately equal to z score. Thus, we use the value of t score for a 95% confidence interval as 1.96. Using Python, we calculate the confidence interval which equals to [0.16177179037256378, 0.16480714680779668] (see calculations in Appendix E). Since the range of the confidence interval is very small, our true value of the danceability mean is likely to be within the range.[6]

---

[5] **#significance** - I used the statistical and practical significance tests to interpret the hypotheses. I explained the difference of both given our context. The statistical significance test and a large effect size showed that the valence affects danceability. In conclusion, I acknowledged that there might be a 5% chance for a Type I error since we rejected the null hypothesis.

[6] **#confidenceintervals** - I accurately constructed a confidence interval for the difference of the means with detailed steps using correct formulas and values. I applied a confidence interval for estimating population parameter - the danceability mean - with an explained interpretation. Also, I justified the benefits of using a confidence interval.

Results and Conclusions

We have successfully examined the relations between songs' valence and danceability based on our dataset. First, we examined the data, then divided it into two subgroups to conduct the difference of means test. Since the p-value was less than the significance level, we rejected the null hypothesis in favor of the alternative one. Thus, it is likely that there is a relationship between valence and the danceability of the song, but not that one affects the other. However, we should remember that since the significance level is 0.05, there is a 5% chance to make a Type I error - the null hypothesis might be true, though we rejected it. After assessing the practical significance, we calculated the effect size to see whether we should care about the difference. Using Hedge's g, we proved that the effect size is large and there is a practical significance. Also, we built a confidence interval for the difference of the means which showed that a true danceability mean varies from 0.515 to 0.544.

These results are inductive since statistical proof and a number of instances contribute to logical generalization. The conclusion is strong since the premises are based on a substantial dataset and are verified to be valid by two separate methods: Python and personal calculations. It is reliable since the data relates to a large number of songs.

To sum up, we have shown that summary statistics, confidence interval, and difference of means test led to the reject the null hypothesis in favor of the alternative hypothesis; thus, we may infer that the valence of the song affects its danceability.

References

Tamer, N. (2018). *Top Spotify Tracks of 2017*. Kaggle. https://www.kaggle.com/nadintamer/top-tracks-of-2017

Get Audio Features for a Track (n.d.). Spotify for Developers. https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/

Appendix

Base

The full Jupyter notebook file was submitted in a zipped folder. The data can be accessed here.

Appendix A: Import and Analyze Data

```
#import necessary packages and libraries
from scipy import stats
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#import the data using pandas
spotify_data = pd.read_csv("spotify_data.csv")
#show the first 10 rows of data
spotify_data.head(10)
```

	acousticness	artists	danceability	duration_ms	energy	explicit	id	instrumentalness	key	liveness	loudness	mode	
0	0.995	['Carl Woitschach']	0.708	158648	0.1950	0	6KbQ3uYMLKb5jDxLF7wYDD	0.563	10	0.1510	-12.428	1	Bata
1	0.994	['Robert Schumann', 'Vladimir Horowitz']	0.379	282133	0.0135	0	6KuQTIu1KoTTkLXKrwlLPV	0.901	8	0.0763	-28.454	1	Fan (
2	0.604	['Seweryn Goszczyński']	0.749	104300	0.2200	0	6L63VW0PibdM1HDSBoqnoM	0.000	5	0.1190	-19.924	0	Ch
3	0.995	['Francisco Canaro']	0.781	180760	0.1300	0	6M94FkXd15sOAOQYRnWPN8	0.887	1	0.1110	-14.734	0	(Rer
4	0.990	['Frédéric Chopin', 'Vladimir Horowitz']	0.210	687733	0.2040	0	6N6tiFZ9vLTSOIxkj8qKrd	0.908	11	0.0980	-16.829	1	Fa Fla
5	0.995	['Felix Mendelssohn', 'Vladimir Horowitz']	0.424	352600	0.1200	0	6NxAf7M8DNHOBTmEd3JSO5	0.911	6	0.0915	-19.242	0	
6	0.956	['Franz Liszt', 'Vladimir Horowitz']	0.444	136627	0.1970	0	6O0puPuyrxPjDTHDUgsWI7	0.435	11	0.0744	-17.226	1	V Sha
7	0.988	['Carl Woitschach']	0.555	153967	0.4210	0	6OJjveoYwJdIt76y0Pxpxw	0.836	1	0.1050	-9.878	1	P
8	0.995	['Francisco Canaro', 'Charlo']	0.683	162493	0.2070	0	6OaJ8Bh7IsBeYoBmwmo2nh	0.206	9	0.3370	-9.801	0	Re
9	0.846	['Seweryn Goszczyński']	0.674	111600	0.2050	0	6PrZexNb16cabXR8Q418Xc	0.000	9	0.1700	-20.119	1	C

```
#print descriptive statistics for "danceability" variable
spotify_data['danceability'].describe()
```

```
count    169909.000000
mean          0.538150
std           0.175346
min           0.000000
25%           0.417000
50%           0.548000
75%           0.667000
max           0.988000
Name: danceability, dtype: float64
```

```
#print descriptive statistics for "valeance" variable
spotify_data['valence'].describe()
```

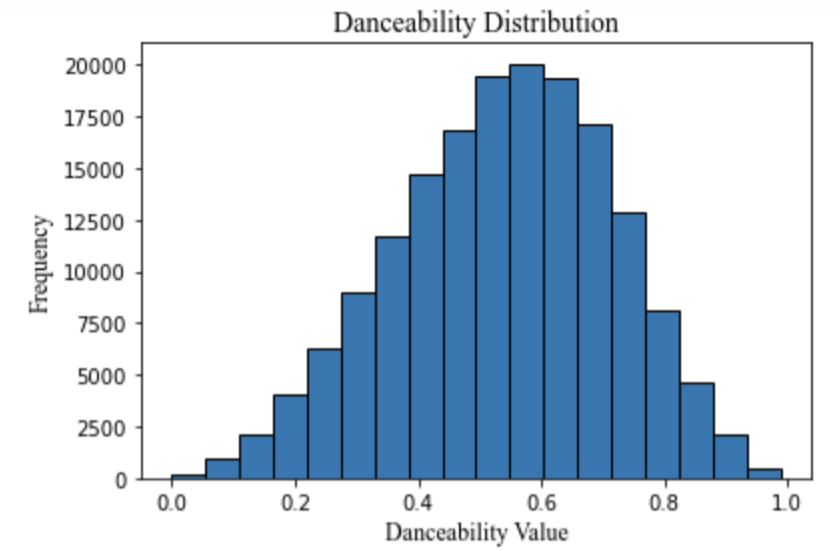
```
count    169909.000000
mean          0.532095
std           0.262408
min           0.000000
25%           0.322000
50%           0.544000
75%           0.749000
max           1.000000
Name: valence, dtype: float64
```

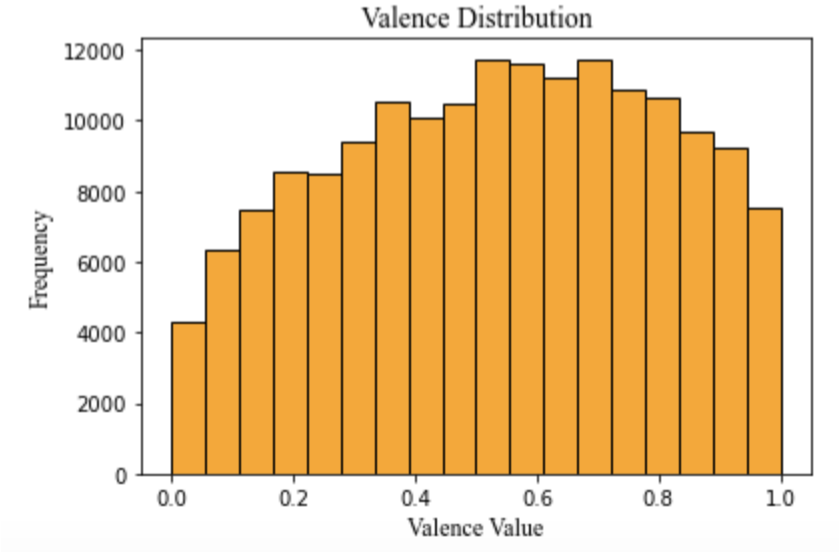
```
#calculate the meadian and the mode using library functions
print("Danceability")
print("- Median", spotify_data['danceability'].median())
print("- Mode", spotify_data['danceability'].mode())

print("Valence")
print("- Median", spotify_data['valence'].median())
print("- Mode", spotify_data['valence'].mode())
```

```
Danceability
- Median 0.5479999999999999
- Mode 0    0.565
dtype: float64
Valence
- Median 0.544
- Mode 0    0.961
dtype: float64
```

Appendix B: Visualize Data

```
#transfer the data into a list
danceability = spotify_data['danceability'].tolist()
valence = spotify_data['valence'].tolist()

#create histograms for each variable
spotify_data['danceability'].plot(kind = 'hist', bins = 18, ec = 'black')
plt.title('Danceability Distribution', fontname = 'times new roman', fontsize = 14)
plt.xlabel('Danceability Value', fontname = 'times new roman', fontsize = 12)
plt.ylabel('Frequency', fontname = 'times new roman', fontsize = 12)
plt.show()
spotify_data['valence'].plot(kind = 'hist', bins = 18, ec = 'black', color = 'orange')
plt.title('Valence Distribution', fontname = 'times new roman', fontsize = 14)
plt.xlabel('Valence Value', fontname = 'times new roman', fontsize = 12)
plt.ylabel('Frequency', fontname = 'times new roman', fontsize = 12)
plt.show()
```



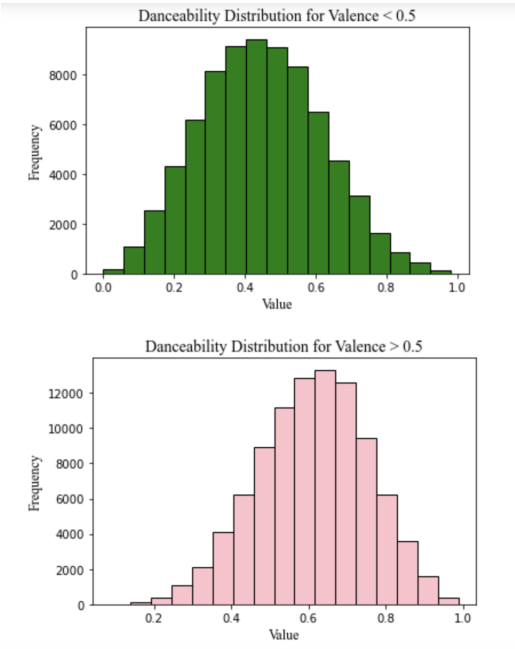


Appendix C: Examine the Subgroups

```
#divide valence data into two parts and finding statistics of danceability for both

less_05 = spotify_data[spotify_data['valence'] < 0.5]
more_05 = spotify_data[spotify_data['valence'] > 0.5]

print("Valence less than 0.5:")
print("-Mean:", less_05['danceability'].mean())
print("-SD:", np.std(less_05['danceability'], ddof = 1))
print("-Sample size:", len(less_05))

print("Valence more than 0.5:")
print("-Mean:", more_05['danceability'].mean())
print("-SD:", np.std(more_05['danceability'], ddof = 1))
print("-Sample size:", len(more_05))
```

```
Valence less than 0.5:
-Mean: 0.447634003172918
-SD: 0.17063028352311016
-Sample size: 75640
Valence more than 0.5:
-Mean: 0.6109234717630982
-SD: 0.1421120289361929
-Sample size: 94079
```

```
less_05['danceability'].plot(kind = 'hist', bins = 17, ec = 'black', color = 'green')
plt.title('Danceability Distribution for Valence < 0.5', fontname = 'times new roman', fontsize = 14)
plt.xlabel('Value', fontname = 'times new roman', fontsize = 12)
plt.ylabel('Frequency', fontname = 'times new roman', fontsize = 12)
plt.show()
more_05['danceability'].plot(kind = 'hist', bins = 17, ec = 'black', color = 'pink')
plt.title('Danceability Distribution for Valence > 0.5', fontname = 'times new roman', fontsize = 14)
plt.xlabel('Value', fontname = 'times new roman', fontsize = 12)
plt.ylabel('Frequency', fontname = 'times new roman', fontsize = 12)
plt.show()
```



Appendix D: Difference of Means Test

```
#calculate the difference of means
n1 = len(less_05)
n2 = len(more_05)

x1 = less_05['danceability'].mean()
x2 = more_05['danceability'].mean()

s1 = np.std(less_05['danceability'], ddof = 1) #apply Bessel's correction
s2 = np.std(more_05['danceability'], ddof = 1)

SE = np.sqrt(s1**2/n1 + s2**2/n2)
tscore = np.abs((x2-x1))/SE
df = min(n1, n2) - 1 #find degrees of freedom

pvalue = 2*stats.t.cdf(-tscore, df) #2 tails

Spooled = np.sqrt(((n1-1)*s1**2+(n2-1)*s2**2)/(n1+n2-2))
g = (x1-x2)/Spooled

print("Standard Error", SE)
print("T-score", tscore)
print("Degrees of freedom:", df)
print("P-value:", pvalue)
print("Hedge's g:", g)
```

```
Standard Error 0.0007743256212328783
T-score 210.87958878358097
Degrees of freedom: 75639
P-value: 0.0
Hedge's g: -1.0502990917576709
```

Appendix E: Confidence Interval

```
#calculate the confidence interval
mean = (x1+x2)/2

ME = tscore*np.sqrt(SE/(n1+n2))

upperbound = mean + ME
lowerbound = mean - ME

print('Confidence interval:', [lowerbound, upperbound])
```

```
Confidence interval: [0.51503476489526, 0.5435227100407563]
```